Identification of Correlation Between 3D Surfaces Using Data Mining Techniques: A Case Study of Predicting Springback in Sheet Metal Forming

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy

by

Subhieh El Salhi

Faculty of Science
Department of Computer Science

October 2014

To my beloved parents for your love and encouragement, especially my dad Dr. "Mohd Faraj" Saad El Din Al Salhi, my greatest inspiration of all.

Abstract

This thesis presents data mining research work undertaken in the context of identifying correlations between 3D surfaces. More specifically, this research is directed at predicting distortions (referred to as springback) in sheet metal forming. The main objective was to identify a mechanism that "best" serves to both capture effectively 3D geometrical information while at the same time allowing for the generation of effective predictors (classifiers). To this end, three distinct 3D surface representation techniques are proposed based on three different concepts. The first technique, the Local Geometry Matrices (LGM) representation, is founded on the idea of Local Binary Patterns (LBPs), as used with respect to image texture analysis, whereby surfaces are defined in terms of *local neighbourhoods* surrounding individual points in a 3D surface. The second technique, the Local Distance Measure (LDM) representation, is influenced by the observation that springback is greater further from edges and corners, consequently surfaces are defined in terms of *distance* to the nearest edge or corner. The third technique, the Point Series (PS) representation, is founded on the idea of using a spatial "linearisation" with which to represent surfaces in terms of *point series curves*. The thesis describes and discusses each of these in detail including, in each case, the theoretical underpinning supporting each representation. A full evaluation of each of the representations is also presented. As will become apparent, the PS technique was found to be the most effective. The presented evaluation was directed at predicting springback, in the context of the Asymmetric Incremental Sheet Forming (AISF) manufacturing process, in such a way that an enhanced version of the desired 3D surface can be proposed intended to minimise the effect of springback. For the evaluation two flat-topped, square-based, pyramid shapes were used. Each pyramid had been manufactured twice using Steel and twice using Titanium. In addition this thesis presents some idea on how the springback prediction mechanism can be incorporated into an "intelligent process model". The evaluation of this model, by manufacturing corrected shapes, established that a sound prediction framework, incorporating the 3D surface representation techniques espoused in this thesis coupled with a compatible classification technique, had been established.

Acknowledgements

This thesis is the end of my three year journey to get my Ph.D degree in Computer Science. First and foremost, all praise be to Allah, the Most Gracious and Most Merciful, for giving me the power, the strength and the patience to overcome all tests that came my way and for blessing me with those wonderful people who have believed that my efforts will be fruitful by the end of my Ph.D journey.

I am deeply indebted to my Supervisor, Professor Frans Coenen, who has offered me the chance to work under his supervision and who has supported me with his invaluable assistance, care and guidance throughout my Ph.D research. He has been always there to listen, discuss and suggest research ideas. Through his extraordinary experience, he has taught me not only to be a good student but also a good researcher and an intellectual person. He has enlightened my way by his inspiration and endless efforts on how to explain and present academic work simply and clearly. He has been always not only a great mentor but also a real friend. He was the most perfect resource which inspired me and enriched my experience making me the person that I am today. He was the best supervisor any one could hope for, and more. It has been a pleasure and an honour to have been supervised by him.

My deepest appreciation and gratitude to my second supervisor Dr. Clare Dixon for her assistance, constructive suggestions, insightful comments, feedback and research ideas. Besides my supervisors, I would also like to extend my gratitude to my assessor committee; Professor Trevor J.M. Bench-Capon, Dr. Boris Konev and Dr. Muhammad Khan for their evaluation, constructive comments and feedbacks. I would also like to thank staff at Tecnalia (Spain) and IBF (Germany) for their support in providing test data and the manufacturing of corrected shapes, in particular: Dr. Mariluz Penalva Oscoz and Dr. Asun Rastrero from Tecnalia, and David Baily from IBF.

Beside the facilities that have been provided by the Department of Computer Science, the kindness of people that characterises the department to be the perfect place to work in, I am also grateful to the staff at the department for helping me in numerous ways. Thanks also go to my fellow PhDs who stood by my side, provided me with incredible support and have been always there for me when I needed them. In particular, Puteri N. E. Nohuddin, Matias Fernando Garcia, Kwankamon Dittakan, Wen Yu, Latifa Al-Abdulkarim, Maduka Attamah, Jeffery Raphael and Eric Schneider.

It is very difficult for me to find the right words to express my gratitude to my parents, Dr. "Moh'd Faraj" Al Salhi and Basema Abu Zer. They not only have raised

me with endless care, love and support, but also they have constantly encouraged me to be an independent and self confident person and to contribute positively to society. Without their support and encouragement, this work would not have been started. I would like also to thank my sisters and brothers; Mayssa, Amer, Shaima, Fatima, Alaa El Deen, Alaa and Leena for their faith in me and my ability to accomplish my Ph.D tasks professionally and also for their continuous love and respect.

My literary skills are powerless to express my respect, love and appreciation to my husband, Moneef Al Qaisi for the patience, support and persistent confidence he provided me with. He has been always my best friend my confidant and my role model. I am blessed to share my life with him. His love and kindness are foundation stones for all that I am and currently have.

My sincere acknowledgements would not be complete without true thanks to my angels Khaled, Tala and Ahmad. I am blessed to have such wonderful kids in my life. I could not have coped with the pressures of life without their love and hugs. I owe them so much; they have always inspired me to work harder, smarter and to utilise my time effectively. They have always been the source of my strength to keep reaching for the best. I hope that they will be proud of me when they realize what my Ph.D years have entailed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Data mining is the process of extracting useful information from data. The discipline has emerged as a response to the global increase in the amount of data that is available for analysis, facilitated by corresponding technical advances in the ways that we are able to collect and store data. The term *data mining* has been popularly used as a synonym for *Knowledge Discovery in Databases* (KDD) by some researchers such as [74, 191], while others view data mining as a sub-process within KDD such as [62, 63, 93, 94]. In this latter case KDD is viewed a multi stage process that includes elements of data pre-processing and post processing of results (as well as the central data mining process). In this thesis the latter view has been adopted. Currently data mining encompasses a variety of different application oriented tasks within which we can include classification, prediction, pattern recognition and clustering. The technology used to realise these tasks is in part borrowed from the related fields of machine learning and statistics, and in part is unique to the discipline of data mining. The work described in this thesis is directed at classification (although the work also encompasses elements of pattern recognition). Classification is what is known as a "supervised learning" technique in that it requires the availability of pre-labelled training data with which to build a classifier that can then be used to label "unseen" data.

Data mining was originally concerned with tabular data [3], but since its conception has been applied to increasingly complex forms of data such as text, images, video, graphs and so on. In many cases it is not the data mining techniques that are of concern, rather the way in which that data can be organised (represented) so as to permit data mining. The work described in this thesis is concerned with the mining of three dimensional (3D) surfaces. There are a number of applications where this is applicable, for example geological analysis for terrains (so called "Terrain Classification") as presented in [130, 146] and image texture analysis such as that presented in [22]. (Images can be viewed as 3D surfaces where the third dimension represents a "grey scale" value.) There is also related work concerned with 3D object classification (mostly used in the medical

field in the context of Magnetic Resonance Imaging and Optical Coherence Tomography data) such as that found in [5, 203]. The work presented in this thesis is directed at 3D surfaces describing fabricated components produced using sheet metal forming processes. More specifically, the work is directed at using classification techniques to predict distortions in such components that occur as a result of the application of the sheet forming processes used to produce them. To the best knowledge of the author there is no reported research on classification techniques directed at 3D surfaces with the aim of identifying (predicting) distortions associated with such surfaces.

The commercial motivation and the application focus for the work described in this thesis, as noted above, is sheet metal forming. There is an increasing demand for accurate and well formed sheet metal components in a variety of industries (such as the automotive and aircraft manufacturing industries). To this end there are a number of manufacturing process that can be adopted. One such process, and that used for evaluation purposes with respect to the work described in this thesis, is Asymmetric Incremental Sheet Forming (AISF). In AISF the metal sheet from which the desired component is to be manufactured is clamped into a "blankholder", a forming tool then follows a predefined tool path to "push out" a desired shape [115]. The main advantage of AISF, over alternative sheet metal forming processes, is that of cost reduction [86, 190]. However, a major limitation of techniques such as AISF is that as a result of applying the process deformations, called *Springback* are introduced whereby the produced shape is not the same as the desired shape. Springback is defined as the elastic deformation that occurs in a produced shape, as a result of the application of a sheet metal forming process, that become apparent when the manufactured piece is unclamped. In other words the produced shape differs from the desired shape. Springback is a complex physical phenomenon that normally occurs because of the elastic properties of the material being worked. The motivation for the work described in this thesis is considered further in Section 1.2 below.

The work described in this thesis is thus directed at representing 3D surfaces in such a way that classification techniques can be employed so as to effectively predict springback (so that some mitigation can be applied). The main challenge of the work is how "best" to describe the 3D surfaces, that define the sheet metal components, so that the springback phenomena can be effectively predicted. An alternative way of viewing the work described is that it is directed at uncovering correlations between two 3D surfaces, T and T', so as to generate classifiers that can be used to predict the correlation associated with "unseen" shapes. More generally the thesis addresses a number of issues concerned with the representation of 3D surfaces and the employment of classification techniques with respect to such surfaces (these issues are discussed further in Section 1.3 below). The thesis proposes several different approaches to address these issues. As will become apparent later in this thesis, very positive feedback was obtained from the industrial partners who provided support for the work described (IBF[1] and

[1]The Institute of Metal Forming (IBF) is a research institute belonging to the Faculty of Georesources and Materials Engineering of RWTH Aachen University. See [180] for more details.

Technalia[1]) and manufactured a number of "corrected shapes" produced using one of the proposed springback prediction techniques. Perhaps the most significant novelty of the work described in this thesis is in the context of the practical application of data mining to support a real commercial application, namely sheet metal forming.

The rest of this introductory chapter is organised as follows. A more detailed description of the motivations of the work described in this thesis is presented in Section 1.2. Section 1.3 presents the main research question and its related issues. The main research contributions are listed in Section 1.4. The research methodology adopted is presented in Section 1.5. A review of the published work to date resulting from the research described in this thesis is presented in 1.6. Section 1.7 represents the organisation of the rest of this thesis. Finally, this chapter is concluded with a summary presented in Section 1.8.

1.2 Motivation

From the foregoing the motivation for the research described in this thesis is to provide a solution to a real world problem encountered in the sheet metal forming industry, more specifically, to identify techniques for predicting the springback phenomena. Accurate prediction of springback is considered to be of great significance with respect to sheet metal manufacturing processes in general. Generally speaking, springback is induced as a result of a variety of factors such as: (i) the material itself (ambient temperature, thickness, type of raw material and so on), (ii) the manufacturing tools used (such as the dimension and the shape of the tool head in the case of AISF) and (iii) the product geometry [13, 18, 86, 114, 115]. Previous work conducted to minimise the springback impact may be categorised into two main groups. The first group describes work directed at reducing springback by modifying the manufacturing parameters such as the force used to form the shape, this approach has been found to be expensive and has very limited application in real life [141, 194]. The second group describes work intended to predict springback and then to compensate for it by modifying the input shape description. The work described in this thesis falls into this second category. In this second category there is a substantial body of work based on using the Finite Element Method (FEM) [38, 136, 178], however the application of FEM requires significant resource because of the large number of parameters that need to be considered. In order to overcome this limitation, and since springback is a "local" phenomena [12] due to the local forming nature of AISF and distributed unequally across a desired shape, the approach taken in this thesis is founded on a "local geometry" based approach. In summary the work presented in this thesis is motivated by the following.

1. Commercial needs with respect to the sheet metal forming industry (especially the automotive and airplane manufacturing industries).

[1]TECNALIA is a technological corporation, located in northern Spain, involved in sheet metal forming research. See [111] and [112] for more details.

2. The need for techniques to represent 3D surfaces in terms of their local geometry, as springback is a local phenomena related to local geometries, in order to be able to conduct further processing so as to effectively predict springback.

3. The desire to be able to generate effective classifiers to predict springback (based on 3D surface representation techniques).

4. The desire to supply manufacturers with a "corrected" input description that would serve to mitigate against the springback effect and consequently improve the quality of shapes intended to be produced in the context of sheet metal forming.

5. The opportunity to research an aspect of data mining that has not previously received attention with respect to the available literature (to the best knowledge of the author).

The motivation for selecting the AISF sheet metal forming process to be the exemplar application was because of ongoing collaborative research carried out within the Computer Science department in the University of Liverpool with respect to the EU funded *Innovative MAnufacturing of complex Ti sheet components* (INMA) project (See [112] for more details on this project). This has meant that suitable "training data" was readily available together with access to domain experts in industry.

1.3 Research Question and Related Issues

Given the motivations presented in the previous section, the main research question is *"How best can 3D surfaces be represented to reflect local geometrical information according to certain feature(s) of interest so that classification techniques can be applied effectively?"*. As noted at the start of this chapter, classification requires the provision of "training data". This thesis assumes that this training data will be available in the form of a grid where the grid centre points are defined in terms of a x-y-z coordinate system. Each grid centre point will also have a "class label" associated with it; the value which we wish our classifier to eventually be able to predict. The main challenge of the work described in this thesis is thus how best to represent this data so that effective classifiers can be built.

In the context of sheet metal forming, information concerning the before and after surfaces is presented in the form of "point clouds", sets of points $P = \{p_1, p_2, p_3, \ldots, p_n\}$ where each $p_i \in P$ is defined in terms of x-y-z coordinates [192]. In the context of the AISF application domain the training data was generated using two 3D surface descriptions, input C_{in} and output C_{out}, where C_{in} is a "point cloud" describing the *desired* shape T while C_{out} is a point cloud describing the *produced* shape T'.

To address the research question, as formulated above, the following related research issues were identified:

1. *Training Set Generation.* How best to generate the required training data, in the desired grid format, given two correlated input clouds. In other words how best to identify a correspondence between C_{in} and C_{out}, so that appropriate (grid) training data can be produced.

2. *3D Surface Representation.* The need to determine a most appropriate 3D surface representation technique. The representation technique that is best able to describe the nature of 3D surfaces so as to allow the effective application of classification algorithms.

3. *Best Classification Technique.* Given a particular grid encapsulation there are a number of different classification techniques that can be applied. Different surface representations may require different classification algorithms to be employed. There is a need to thus identify the most appropriate classification algorithm/approach compatible with a particular representation.

4. *Corrected Input Generation.* With respect to unseen data, once appropriate class labels have been identified, these need to be applied in some way. How these are applied is application dependent but in the context of sheet metal forming, and especially AISF, a "corrected" cloud must be produced. How such a corrected cloud can best be produced was thus an issue to be addressed.

It should also be noted that, although the above is directed at AISF, any proposed solution to the identified research issues will also have more general applicability; however, this is not considered further in this thesis.

Given the above the following main objectives for the work described in this thesis were identified:

1. To research and identify a framework to compare T and T' (C_{in} and C_{out}) shapes in order to generate the required training data in a simple, easy and effective manner.

2. To research, identify and evaluate a variety of 3D surface representation techniques appropriate to classifier generation.

3. To research and investigate the suitability of a variety of classification techniques with respect to the proposed representation techniques (individual representation techniques may be best suited to different classification algorithms).

4. To research and investigate mechanisms to generate corrected clouds in order to utilise the predictions produced by classifiers.

5. To investigate the possibility of defining an "Intelligent Process Model" with general applicability within the sheet metal forming industry.

1.4 Contributions

The main contributions of this thesis are the proposed representation techniques for modelling 3D surfaces in such a way that the main geometrical characteristics of such 3D surfaces are captured. Most previously considered 3D surface representation techniques have been directed at the application domain of visualisation inspection [19, 106, 205]; however, as noted at the start of this chapter, little work has been done related to the representation of 3D surface to support both feature extraction and classification. Therefore, based on the aforementioned objectives of this thesis, the main contributions of the research (with respect to both computer science and industry) can be itemised as follows:

1. **A 3D Surface Correlation Technique**: In the context of sheet metal forming a novel and effective framework to identify the correlations between the C_{in} and C_{out} point clouds describing a given surface in order to effectively identify the features of interest (springback) so that a comprehensive description for the desired shape in terms of local geometry and anticipated springback can be produced.

2. **The Representation And Springback Prediction (RASP) Framework**: In the context of sheet metal forming, an effective framework for processing point cloud data and generating springback classifiers for future use.

3. **The Local Geometry Matrix (LGM) representation**: A novel technique to represent 3D surfaces in terms of local neighbourhoods based on ideas taken from the field of image texture analysis (especially the use of Local Binary Patterns [88]).

4. **The Local Distance Measure (LDM) representation**: A mechanism to represent 3D surfaces in terms of the distance to a nearest edge or corner (there appears to be a correlation between springback and proximity of edges).

5. **The combined LGM and LDM representation**: An effective 3D surface representation technique that combines the proposed LGM and LDM representations.

6. **The Point Series (PS) representation**: An novel and effective technique to represent 3D surfaces in terms of a linearisation of space, a series of points, that can be incorporated into a KNN classification style technique.

7. **Best Classification Techniques**: The identification of individual classification techniques best suited to each of the proposed representations.

8. **A Corrected Point Cloud Generation Technique**: In terms of sheet metal forming an effective mechanism with which to generate corrected point clouds based on springback predictions.

9. **Intelligent Process Models (IPMs)**: Two mechanisms for combining the springback prediction and corrected point cloud generation mechanisms.

1.5 Research Methodology

In order to provide an answer to the above research question and acknowledging, in the context of sheet metal forming, that springback is primarily affected by the nature of local geometries, the proposed research methodology was centred on possible representations of 3D surfaces. In other words the broad research methodology was to investigate a sequence of different techniques to represent 3D surfaces. As indicated in Section 1.4 three different styles of representation were considered: (i) a Local Binary Pattern (LBP) based technique, (ii) a distance from nearest edge based technique and (iii) a linearisation of space based technique.

As noted at the beginning of this chapter classification requires the provision of training data. To this end "real provenance" data sets, describing 3D surfaces that have been manufactured, were obtained from industry. More specifically The Tecnalia Corporation (Spain) and IBF institute of metal forming (Germany) with whom the Department of Computer Science at The University of Liverpool has contact within the context of the INnovative MAnufacturing (INMA) Framework 7 European project. The data sets described two flat topped pyramid shapes referred to as the Gonzalo and Modified pyramids (more details of these data sets will be presented later in Chapter 3). In total eight data sets were obtained each comprising before and after point clouds.

To evaluate the proposed representations each was applied to the provided data and the result used to generate classifiers. These classifiers were then evaluated using standard approaches used with the field of data mining (such as the use of Ten-fold Cross Validation (TCV) [73], and the analysis of metrics such as the Area Under the receiver operating Curve [26, 97]). As will become apparent later in this thesis some excellent results were produced. To determine whether the results were also statistically significant Friedman and Nemenyi tests were used [50, 66, 76].

In the context of the sheet metal forming application used as a focus for the work described in this thesis it was also considered desirable to identify mechanisms whereby springback predictions could be used to generate corrected clouds. This required an investigation into the generation of corrected clouds. So that the correction mechanism could be effectively applied it was decided to combine the mechanism with the classification (springback prediction) process into a single Intelligent Process Model (IPM). Two IPMs were considered: (i) a single pass IPM and (ii) an iterative IPM. The distinction being that the second incorporated an iterative prediction-correction loop rather than a straightforward application of the prediction result.

To evaluate the IPM concept a number of corrected shapes were manufactured (by IBF in Germany) so that a full evaluation of the research work described in this thesis could be undertaken. The organisation of these tests required a significant lead time and thus it was only possible to test the LGM representation, coupled with the single pass IPM, in this manner. However, again as will become apparent later in this thesis, the results obtained were very encouraging.

1.6 Published Work

In this section an annotated list of publications to date that have arisen from the work described in this thesis is presented:

- **Conference papers:**

  (a) *M. Khan, F. Coenen, C. Dixon, and S. El-Salhi. "Finding Correlations Between 3D Surfaces: A study in Asymmetric Incremental Sheet Forming". In Machine Learning and Data Mining in Pattern Recognition (Proceedings Conference MLDM 2012), Springer LNAI 7376, pages 336-379, 2012.* This paper presented the first 3D surface technique, the LGM technique, proposed in this thesis. In the paper several variations of the LGM technique combined with different classification techniques were considered and evaluation. The main finding was that there is no significant difference between them. It should also be noted that the reported evaluation was conducted using two alternative data sets, referred to as the small and large pyramids for obvious reasons, which are not referred to with respect to the work described in this thesis (because the new data sets are more *reliable* specially in terms of materiel type.). This paper is related to the grid representation and the LGM techniques described in Chapters 3 and 4 respectively.

  (b) *S. El-Salhi, F. Coenen, C. Dixon, and M. S. Khan. "Identification of Correlations Between 3D Surfaces Using Data Mining Techniques: Predicting Springback in Sheet Metal Forming". In SGAI International Conference on Artificial Intelligence, pages 391-404, 2012.* This paper followed on from the work described in (a) and introduced the second proposed 3D surface representation technique, the LDM technique. Evaluation was conducted by comparing the operation of the LDM technique with the LGM technique proposed earlier. A combination of both LGM and LDM was also proposed. The main finding was that the combination of the LGM and LDM techniques achieved much better performance than when each technique was used in isolation. Again the reported evaluation used the small and large pyramid data sets used in the early stages of the work described in this thesis before the acquisition of the Gonzalo and Modified data sets. This paper is related to the LGM and LDM techniques described in Chapters 4 and 5 respectively.

  (c) *S. El-Salhi, F. Coenen, C. Dixon, and M. S. Khan. "Predicting Features in Complex 3D Surfaces Using a Point Series Representation: A Case Study in Sheet Metal Forming". In Advanced Data Mining and Applications, volume 8346 of Lecture Notes in Computer Science. 2013.* This paper introduced the third 3D representation, the Point Series representation technique. The paper describes how a "bank" of point series patterns (curves) can be established to describe a given 3D surface and how a k-nearest neighbour technique can

applied for prediction purposes. The technique was evaluated, in the context of sheet metal forming, using the Gonzalo and Modified pyramids (as opposed to the small and large pyramid data sets used in earlier work). This paper is related to the PS techniques described in Chapter 6.

- **Journal paper:**

  (d) *S. El-Salhi, F. Coenen, C. Dixon, and M. S. Khan. "On Predicting Springback Using 3D Surface Representation Techniques: A Case Study in Sheet Metal Forming". In Expert Systems with Applications (2015), Vol. 42, pages 79 - 93.* This paper is a combination and extension of the work described in (a), (b) and (c), presented in such a way that a more substantial comparisons between the operations of the three representation techniques, LGM, LDM and PS, was included along with a statistical significance comparison. The statistical comparisons showed a significant difference between the proposed techniques. The Gonzalo and Modified pyramids were used for the evaluation. This paper is related to the LGM, LDM and PS techniques described in Chapters 4, 5 and 6 respectively and the statistical study presented in Chapter 7.

  (e) *M. S. Khan, D. Bailly, F. Coenen, C. Dixon, S. El-Salhi, M. Penalva, A. Rivero, "An Intelligent Process Model: Predicting Springback in Single Point Incremental Forming". In The International Journal of Advanced Manufacturing Technology (2014), Vol. 74, pages 1-12.* This paper presented the Intelligent Process Model (IPM) as a process to generate "corrected" versions of the input clouds. The LGM technique presented in papers (a) and (b) was used to translate the 3D surface into a suitable format whereby a classifier could be used to generate the springback prediction effectively. Comparisons between the shapes produced using the initial input clouds and the shapes produced using the corrected clouds are presented, demonstrating that the IPM can be successfully used to improve the quality of the produced shapes. This paper is related to the work presented in Chapters 8 and 9.

The work described in this thesis has also lead to a number of "spin off" investigations, not reported on in this thesis, which in turn has resulted in further publications (to which the author has made some contribution). For, completeness these publications are summarised below.

- **Conference papers:**

  (f) *W. Yu, F. Coenen, M. Zito, and S. El-Salhi. Minimal Vertex Unique Labelled Subgraph Mining. In DaWaK, pages 317-326, 2013.* This paper proposed the concept of Vertex Unique Labelled Subgraph (VULS) mining, a variation of graph mining. The significance with respect to the work reported in this

thesis was that the VULS concept can be used as another mechanism for representing 3D surfaces. The reported evaluation was founded on the Gonzalo data set also used with respect to the work described in this thesis.,

(g) *W. Yu, F. Coenen, M. Zito, and S. El-Salhi. Vertex Unique Labelled Subgraph Mining. In SGAI Conf., pages 21-37, 2013.* This paper is an extended version of (f) and proposed minimal VULS mining (a variation of VULS mining). The significance of this paper in the context of the work described in this thesis is that evaluation was again conducted in the context of AISF springback prediction using the Gonzalo data set. The reported experimental results indicate that the proposed minimal VULS algorithm can successfully identify all minimal VULS in reasonable time and with (in some cases) excellent coverage (an important requirement in the context of the AISF sheet metal forming application used as a focus for the work).

(h) *W. Yu, F. Coenen, M. Zito, and S. El-Salhi. Vertex unique labelled sub-graph mining for vertex label classification. In Advanced Data Mining and Applications, volume 8346 of Lecture Notes in Computer Science. 2013.* This paper is a further extension of (f) and (g), again using the AISF application domain for evaluation purposes. The main contribution of the paper was a Match-Voting algorithm for springback prediction. The results again indicated that the minimal VULS concept could be successfully applied in the context of sheet metal forming.

With respect to the above three publications it should be noted that the work on VULS is on going and has yet to reach full fruition. It is outside the scope of this thesis and thus not considered further in the following chapters.

1.7 Thesis Organisation

In this section the overall organisation of the remainder of this thesis is presented. Chapter 2 presents the necessary background to the work described together with a review of related work. An overview of the sheet metal forming application domain and the data sets used with respect to the evaluation reported in this thesis are presented in Chapter 3. The detail of the three proposed 3D surface representation techniques are presented in Chapters 4, 5 and 6 respectively. Chapter 7 presents a statistical significance comparison of these three techniques to determine whether there is a statistically significant difference in their operation. Chapter 8 then presents the proposed IPMs. In Chapter 9, some results obtained when real parts are manufactured using one of the proposed IPMs are presented and discussed. This thesis is concluded with a summary of the main findings, and some suggested future work, in Chapter 10.

1.8 Summary

This chapter has defined the main motivations, main research question and related issue associated with the work described in this thesis. The work is broadly focused on three surface representation techniques: (i) The Local Geometrical Matrix (LGM) technique, (ii) the Local Distance Measure (LDM) technique and (iii) the Point Series (PS) technique. The main objective of the work described is to provide an answer to the research question: "*how best can 3D surfaces be represented to reflect local geometrical information according to certain feature(s) of interest so that classification techniques can be applied effectively?*". The main contributions and research methodology were also presented. In the following chapter (Chapter 2) previous work and background to the research presented in this thesis is reviewed.

Chapter 2

Background, Related Work and The Application Domain

2.1 Introduction

This chapter presents a review of the background, related work and the application domain central to the work described in this thesis. The related work is mainly founded on three areas of research study (as shown in Figure 2.1): (i) sheet metal forming and Asynchronous Sheet Metal Forming (AISF) in particular, (ii) Data Mining (especially classification) and (iii) 3D surface representation. Each of these is considered in this chapter. We commence the discussion in Section 2.2 with a review of the application domain. We then go on to discuss the concept of data mining in Section 2.3, and 3D surface representation in Section 2.4. For evaluation purposes, as reported later in this thesis, a number of classification techniques were employed, a review of these techniques is included in Section 2.3. A review of the evaluation metrics, and their derivation, used with respect to the work described in this thesis, is then presented in Section 2.5. Finally, this chapter is concluded with a summary in Section 2.6.

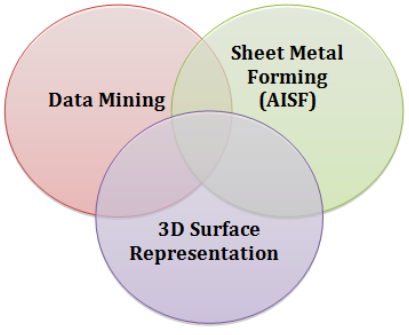


FIGURE 2.1: The main themes of the Thesis: Sheet Metal Forming (AISF in particular), Data Mining (classification in particular) and 3D Surface Representation.

2.2 Application Domain: Asymmetric Incremental Sheet Forming (AISF)

As noted in the introduction to this thesis the exemplar domain at which the work described in this thesis is directed is sheet metal forming, more specifically Asymmetric Incremental Sheet Forming (AISF). AISF as a process was patented in 1967 by Lezak [132], however academic work related to AISF was not undertaken until much later. The first publication, to the best knowledge of the author, was in 2005 [115]. AISF is defined *"as a sheet metal process that: (i) has a solid, small sized forming tool; (ii) does not have large, dedicated dies; (iii) has a forming tool which is in continuous contact with sheet metal; (iv) has a tool that moves under control, in three dimensional space (CNC)*[1] *and (v) can produce asymmetric sheet metal shapes"* [115]. Of note is that the sheet thickness of the part being manufactured reduces (in an irregular manner) as the shape is pushed out; taken to extreme AISF will result in undesired fractures being introduced into the part. The basic component of an AISF machine are the sheet metal blank, the blank holder where the sheet metal is fastened and the forming tool used to form the desired shape according to a prescribed path. The process is illustrated in Figure 2.2. The tool spins at w angular speed and moves horizontally and vertically at v speed to form the shape as shown in the figure. Single Point Incremental Forming (SPIF) or Two Point Incremental Forming (TPIF) are two alternatives to AISF. The distinction is that in TPIF a "die" is used to support the sheet metal over which the tool head is passing. SPIF is sometimes referred to as dieless AISF. With respect to the work described in this thesis; experiments were conducted using only SPIF; hence wherever the term AISF is used this refers to SPIF or dieless AISF.

AISF is mainly used for small production runs and prototyping. AISF supports a wide range of applications in industrial fields such as the aerospace industry, where the trend is to manufacture small numbers of parts. Interestingly the AISF concept has also been used in the medical contexts to produce artificial limbs and dental crowns. For further discussion on the applications of AISF the interested reader is referred to [114, 115].

The most significant issues associated with the AISF sheet metal forming processes are: (i) production time (it is a lengthy process, several hours to produce a single part); (ii) it is time consuming and requires trial and error experiments to determine the *best* parameters values for the process and (iii) deformation (geometric inaccuracies) resulting from the "springback" that is introduced as part of the process [86]. It is the latter that is of concern with respect to the work presented in this thesis. Springback is the geometric "elastic deformation" that manifests itself on completion of the AISF process (when the part is unclamped); as a result the produced shapes is not identical to the intended shape. An additional issue is that springback is unequally distributed and non-linear [36, 40, 209]. Springback tends to be induced by several factors: (i) the shape and dimensions

[1] CNC is an acronym for computer controlled cutting.

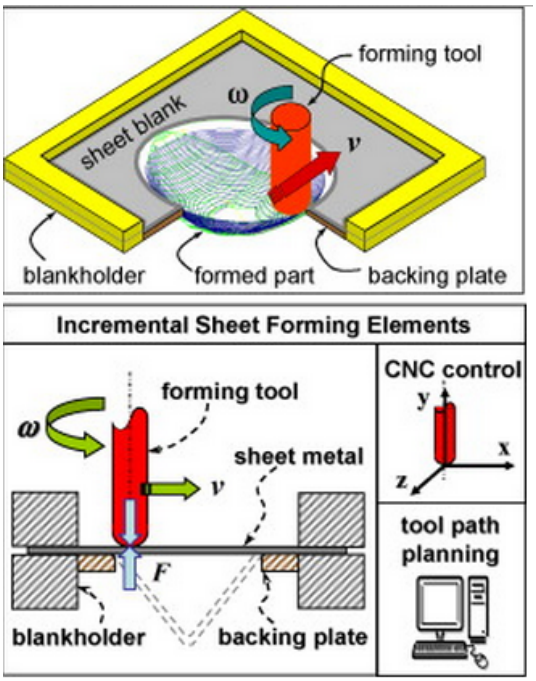


FIGURE 2.2: The AISF process. The basic elements of AISF are the metal forming force F, the tool speed v and the spin angular speed w. [114].

of the manufacturing tool (the tool head), (ii) the properties of the material from which the part is being manufactured and (iii) the geometry of the shape to be manufactured [13, 18, 65, 86, 114, 115, 140, 154]. It is generally acknowledged within the industry that the geometric shape of the part to be manufactured is the most significant factor. Hence the springback phenomena encountered in AISF is good exemplar application domain for the work presented in this thesis. Some further discussion regarding springback is thus presented in the following section.

2.2.1 Overview of Springback

There has been a substantial amount of reported work on springback analysis, prediction and compensation. This previously conducted work can be categorised into four main approaches:

1. **Experimental Approaches**: Work to identify the optimal parameters for a certain (simplified) forming processes and to explain the relationship between these parameters through a series of "try-out" experiments.

2. **Analytical approaches**: Typically concerned with describing the changes in the "before" and "after" geometrical shape of a given part using, mathematically based, theoretical analysis techniques. Generally adopted in the context of simple shapes.

3. **Modelling Approaches**: Used for investigations with respect to more complicated 3D shapes. The idea is to develop models of the manufacturing process with

respect to different parameters. Normally, the Finite Element Method (FEM) or Artificial Neural Network (ANN) methods are employed to generate a fully operational simulation environment whereby springback predictions can be made (and in some cases compensated for).

4. **Regression Approaches**: Work that has adopted a regression based approach to attempt to characterise springback. This approach incorporates the most recent work on springback analysis.

Note that other categorisations have also been proposed (see for example [154]). Some further detail concerning the above four approaches is given below. In each case appropriate examples are given.

Using experimental approaches springback is typically analysed and characterised with respect to particular forming process parameters or material properties [34, 134, 169, 218]. Experimental approaches are typically used to identify the relationship of springback with respect to other parameters such: as sheet thickness, tool radius, tool head size and the different properties of the material used. Two processes, called "V-bending" and "U-bending", are the most popular techniques used with respect to the experimental approach since as a result of the "bending" the induced springback is large and can thus be easily detected and analysed. Examples of this approach can be found in [47] where V-bending was used and Zhang et al. [230] where U-bending was adopted. Despite the simplicity of the approach it does not sufficiently take into consideration actual manufacturing process conditions [34]; it is also time-consuming and expensive [134].

In the case of the analytical approaches springback prediction is founded on a theoretical analysis that entails certain assumptions and simplifications, namely a simplified tool description is assumed and the nature of the manufacturing conditions are simplified. Some work that has adopted the analytical approach has simplified the process even further by considering only simple 2D shapes (and without considering any realistic forming process), see for example [149, 169, 230]. Examples of work using the analytical approach where the forming process is taken into consideration can be found in [14, 230]. It is generally acknowledged that the analytical approach is informative even though the process is over simplified. It has also been suggested that the analytical approach can be used to provide support for modelling approaches so as to provide some understanding of the manufacturing process conditions and parameters that have an impact on the nature of springback.

Modelling approaches, as the name suggests, are directed at building a simulation environment. The most frequently used modelling approaches are based on FEM, see for example [36, 137, 153, 222]. Some researchers, such as Karafillis et al. [119], have proposed iterative FEM models where springback calculations are used to optimise the tool path. Other researchers have used FEM not only to predict but also to compensate for springback error (see for example [38, 150]). There has been some work

where the outcomes from FEM simulations have been analysed using Rough Set Theory (RST). Examples include [185] and [219]. In [185] RST was applied to FEM U-bending simulation results in order to identify optimal forming process parameters. In [219] a framework was proposed, based on the Knowledge Discovery in Data (KDD) process, whereby FEM simulation results could be collected and analysed using RST. Despite the sometimes successful application of FEM it requires significant resource to undertake. The limitation and drawbacks of FEM, as presented in [43, 99, 134, 150, 204], may be summarised as follows.

1. Identifying the different alternatives for each parameter required by FEM is challenging and time consuming.

2. Given that FEM supports different alternatives for each parameter and that these parameters may be related, choosing a particular parameter setting may impact on the options for other parameters.

3. To generate an accurate simulation environment an extensive set of "try-out" experiments must first be undertaken with respect to the alternatives for each parameter. This is a resource intensive process. Hadoush et al. [90] estimate that the time required may vary between 16 hours to a few days.

4. To achieve a highly accurate level of prediction requires accurate identification of the forming parameters which in turn requires a degree of expertise (which may be informed using analytical work).

5. FEM cannot respond interactively when a new condition arises that was not considered during the FEM design stage [56].

6. The choice of alternatives for the various parameters impact upon the simulation process with respect to: (i) the integration algorithm used (implicit or explicit), (ii) the number of integration points and (iii) the type of shape used in the simulation (shell, solid or 2D plane) [134].

7. FEM entails mathematical approximations (such as numerical integration) which in turn are a source for potential inaccuracies (sensitive to numerical tolerances). In other words, the reliability and accuracy provided by FEM, in many cases, do not satisfy industrial requirements [150].

8. Limited applications in real life as prediction using FEM is very time consuming [91, 134].

Despite the significant work that has been conducted to develop and improve FEM simulation/modelling, the accuracy of the resulting springback prediction remains insufficient with respect to industrial requirements [34, 157]. Therefore Artificial Neural Networks (ANNs) are often quoted as being a good alternative modelling approach to

FEM. Moreover, ANN approaches are suited to interactive simulation environments such as those described in [56, 126, 148]. The forming process parameters, such as tool geometry and material properties, are input into the ANN for training purpose. Typically an ANN comprises several layers where each layer consistis of several "neurons". The structure of an ANN is determined after several iterations of training [110]. The work presented in [92] proposed a hybrid simulation environment based on both FEM and ANN simulation to predict springback. However, the main drawback of the use of ANNs is the computational cost (time and memory) required. In addition, when using ANN techniques it is difficult to understand the internal operation due to the black box nature of ANNs. ANN techniques can be argued to be a data mining technique. However, there has been a very little reported work that considered the problem of springback prediction in the context of classification (as in the case of the work described in this thesis).

The final category of previous work on springback analysis considered in this review is the work based on regression analysis. The significance of this approach is that it is the most recent. A good example of the use of this approach, in the context of springback prediction, can be found in [17] where a *Multivariate Adaptive Regression Splines* (MARS) technique to support springback prediction is proposed. MARS incorporates a non-parametric[1] statistical regression test [75]. Initially, two files are generated; one describing the desired shape and one the obtained shape (in other words T and T') in terms of a small number of "features" (types of geometries, such as flat planes), along with a description of their locations, orientations and normals. A comparative analysis between the description files (before and after manufacturing) was used to generate an accuracy file that contains details about each individual point located in the T and T' point clouds along with their springback values in the x, y, and z directions. Features of a particular type (such as flat plane features) were identified and then linked with this accuracy file. Then for each identified individual feature a separate text file was produced that contains information about all the points (vertices) within the feature with respect to certain parameters (some parameters are related to the part to be manufactured such as its geometry while other parameters are related to the manufacturing process such as tool diameter). Series of "try-out" experiments were conducted to determine the main parameters of each relevant feature. After that, the text files were used to generate the "MARS" models which could then be employed to predict springback in the context of new shapes that contain the identified feature types. Although an improved result was reported, the usage of MARS is limited because it typically covers only a limited range of features. Another disadvantage is that the technique requires complex structures to compare features with corresponding parameters; this complexity increases as the number of features is increased and/or when the features are specified to a greater level of detail.

[1] There is no assumption about the relationship between the variables.

The approach presented in this thesis is founded on the idea of classification. The conjecture is that sufficiently robust classifiers can be learnt without the need for: (i) a detailed understanding of the effect that specific parameters have on the springback phenomena (as in the case of the experimental and analytical approaches) or (ii) a deep understanding of the process (as in the case of the modelling approaches) or (iii) restriction to a limited set of "features" (as in the case of the regression approach). To the best knowledge of the author there has been no reported work on the use of classification techniques to predict springback in the context of AISF (or sheet metal forming in general).

It should also be noted that successful springback prediction (however this is done) is only "half of the story". For such predictions to be useful they need to be applied. The application of springback prediction, in order to produce high quality shapes, is also a research issue [10]. For example the first version of the the Displacement Adjustment (DA) algorithm, suggested by [137], was enhanced to produce the Smooth Displacement Adjustment (SDA) algorithm so as to aid the application of springback predictions. Further discussion on the application of springback, in the context of the work presented in this thesis, is described in Chapter 8.

2.3 Knowledge Representation and Data Mining

Knowledge Discovery in Databases (KDD) is the process of knowledge extraction from raw data. The field of KDD came into being in the early 1990s [74]. According to Fayyad *et al.* [62, 63] the KDD process is defined as the *"non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data"*. Despite the different definitions that have been proposed for KDD, such as those presented in [21, 62, 94, 96], the KDD process is generally acknowledged to comprises the following steps [62–64]:

- **Data cleaning and pre-processing**: The detecting, and correcting or removing, of errors caused by the presence of "outliers" or noise and irrelevant data; and the filling in of missing values where appropriate.
- **Data integration**: In the case where data comes from multiple sources the removal of inconsistencies.
- **Data selection**: The identification of a subset of the available variables according to the nature of the knowledge discovery task to be performed.
- **Data transformation**: Translation of the data into an appropriate format with respect to the knowledge discovery technique to be applied.
- **Data mining**: The actual knowledge discovery process where the patterns of interest are identified.

- **Pattern evaluation**: Analysis of the discovered patterns typically using metrics of some form.
- **Usage**: The application of the extracted knowledge in the context of some application domain.

From the foregoing the data mining step is the most significant. A variety of definitions for the phrase "data mining" can be found in the literature [21, 62, 94, 96]. For example Han describes data mining as *"the process of discovering interesting patterns and knowledge from large amounts of data"* [94]. What these definitions have in common is that the ultimate goal of data mining is the extraction of hidden knowledge from data. Thus, KDD can be viewed as a multi-stage process where the data mining stage is the most significant within the overall process [62, 63, 167].

Data mining encompasses a number of different kinds of pattern identification. That of interest with respect to the work described in this thesis is *prediction* (also referred to as *classification* or *categorisation*). The challenge is how best to "learn" the predictor. This is actually done in a supervised manner where we have pre-labelled training data. A variety of algorithms are available for the generation of classifiers using pre-labelled training data [2, 24, 37, 94, 195]. Five are used with respect to the work described in this thesis: (i) Decision Trees (in particular, C4.5 technique), (ii) Rule-Based Classifiers (in particular, JRIP and PART techniques), (iii) Naïve Bayes, (iv) a neural network classifier and (v) k-Nearest Neighbour (k-NN). It should be noted that for the five classification techniques we used the implementation available in Weka [215]. Each is described in more detail in the following five sections.

2.3.1 Decision Tree

Decision Trees (DTs) are a widely used classification technique due to their simplicity, ease of understanding, explanation generation capability and interpretability. In addition, if desired, they can easily be converted into a rule format [216]. In the context of classification a decision tree is a tree structure where the root and body nodes represent alternatives while the leaf nodes represent individual classifications. More specifically each root/body node represents an attribute and the connections to child nodes potential individual attribute values or groups of values. Therefore, a DT can be said to be a tree based classifier. In a binary DT there can only be two alternatives at each root/-body node; in other forms of DT there may be many alternatives emanating from root and body nodes. When constructing a decision tree the challenge is in selecting which attribute is to be represented by which node and how to split the range of potential values that an attribute might have. Once a DT is constructed it becomes very easy and straightforward to classify a new data item starting from the root and finding a route through the DT until one of the leaves (classes) is reached. Typically DT construction is top down following a "greedy" search process, with no backtracking, based on a "divide and conquer" strategy where the training set is partitioned recursively

into subsets according to some splitting criterion. Various splitting criteria have been proposed. Popular measures include Information Gain, Gini Index and Gain Ratio (for more information see [55, 94]). A variety of decision tree generation algorithms have also been proposed.

With respect to the work described in this thesis the C4.5 algorithm [173] was adopted as it has been considered to be a benchmark DT classifier throughout the data mining community. C4.5 uses *Information Gain* (IG) as the splitting criteria whereby the attribute with the highest information gain is selected to be used in the current node. IG is calculated using Equation 2.1:

$$IG(D, X) = Entropy(D) - Entropy(D, X) \tag{2.1}$$

where $IG(D, X)$ is the information gain for the data set D with respect to attribute X. Entropy for the data set D is calculated using Equation 2.2.

$$Entropy(D) = \sum_{i=1}^{i=|c|} -p_i \ \log \ p_i \tag{2.2}$$

where p_i is the probability of class $i \in c$. Normally, $p_i = \frac{|c_{i,D}|}{|D|}$ where $|c_{i,D}|$ is the number of records corresponding to class i with respect to the entire data set D. Intuitively, $0 \leq Entropy(D) \leq 1$. Entropy is a measure of the homogeneity of a given data set. If $Entropy(D) = 0$, then all the records belongs to the same class and therefore the outcome is *certain*. On the other hand, if $Entropy(D) = 1$ this would mean that the data set is totally homogeneous and all classes are equally likely.

IG is thus a measure of the expected reduction in the entropy for a given attribute. In other words IG indicates the "importance" of a given attribute with respect to the DT construction process. In the context of Equation 2.1 the importance of an attribute is determined by identifying the entropy value of the attribute *before* and *after* splitting. The same calculation is made for the complete set of attributes and the attribute that maximises information gain selected for the DT node in question.

Example: The following example illustrate the process of constructing the DT for a given data set (D) where the IG measurement is adopted as the splitting criteria. Table 2.1 describes a small hypothetical data set (D) comprised of 14 records describing bank customers, each record consists of five attributes: (i) *Status* describing whether the customer is a new or existing customer (True or False), (ii) *Age* (Young, Midde or Senior), (iii) *Gender* (Male or Female), (iv) *Income* (Low, Medium or High) and (v) *Loan* (Yes or No). The *Loan* attribute is the class attribute. The steps required to induce the DT for the given data set (D) are described below.

1. For the root node, the IG values for each attribute is calculated. For the *Age* attribute, the entropy values for the data set D before and after splitting are

calculated. Recall that the dataset D is a collection of 14 records with 9 Yes and 5 No records. So, the entropy before splitting is:

$$\begin{aligned} Entropy(D) &= -9/14 \log 9/14 - 5/14 \log 5/14 \\ &= 0.94 \end{aligned}$$

After splitting and with respect to age attribute, the data set (D) is divided into three subsets according to the attribute values as shown in Figure 2.3 and therefore the entropy after splitting $Entropy(D, Age)$ is calculated as follows:

$$\begin{aligned} Entropy(D, Age) = & \sum_{\substack{v \in \{young, \\ middle, \\ senior\}}} \frac{|S_v|}{|D|} \; Entropy(Age_v) \\ & = 5/14 \; Entropy \; (Age_{young}) + 4/14 \; Entropy \; (Age_{middle}) \\ & + 5/14 \; Entropy \; (Age_{senior}) \end{aligned}$$

Where $|S_v|$ is the number of records where S has the label v and S_v/D is the proportion of records that exist in the the set S_v that features the label v ($S_v \subset D$). $Entropy(Age_{young})$, $Entropy(Age_{middle})$ and $Entropy(Age_{young})$ are calculated as follows.

$$\begin{aligned} Entropy(Age_{young}) &= -\tfrac{3}{5} \log 3/5 - \tfrac{2}{5} \log 2/5 = 0.97 \\ Entropy(Age_{middle}) &= -\tfrac{4}{4} \log 4/4 - \tfrac{0}{4} \log 0/4 = 0.0 \\ Entropy(Age_{senior}) &= -\tfrac{3}{5} \log 3/5 - \tfrac{2}{5} \log 2/5 = 0.97 \end{aligned}$$

Now, the value of $Entropy(D, Age)$ is calculated as:

$$\begin{aligned} Entropy(D, Age) &= 5/14 \; (0.97) + 4/14 \; (0) + 5/14 \; (0.97) \\ &= 0.69 \end{aligned}$$

2. The value of the $IG(D, Age)$ is calculated as follows.

$$\begin{aligned} IG(D, Age) &= Entropy(D) - Entropy(D, Age) \\ &= 0.94 - 0.69 \\ &= 0.26 \end{aligned}$$

3. Similarly, the IG value for the rest of attributes are calculated in the same manner and thus $IG(D, Income) = 0.03$, $IG(D, Gender) = 0.15$ and $IG(D, Status) = 0.05$.

4. Based on the obtained IG values, the attribute associated with the highest IG value is selected to represent the current node. In this case, the age attribute is selected to represent the root node.

5. Steps 1, 2, 3 and 4 are repeated recursively for each node in the tree as it is constructed until no more records and/or attributes remain. The Final DT is shown in Figure 2.4 is obtained.

Status (new Customer)	Age	Gender	Income	Loan
True	Young	Female	High	No
False	Young	Female	High	No
True	Middle	Female	High	Yes
True	Senior	Female	Medium	Yes
True	Senior	Male	Low	Yes
False	Senior	Male	Low	No
False	Middle	Male	Low	Yes
True	Young	Female	Medium	No
True	Young	Male	Low	Yes
True	Senior	Male	Medium	Yes
False	Young	Male	Medium	Yes
False	Middle	Female	Medium	Yes
True	Middle	Male	High	Yes
False	Senior	Female	Medium	No

TABLE 2.1: A labeled training data set consists of 14 records. Four attributes are used to describe the data set while the Loan attribute used to label the record with either *Yes* label (coloured in red) or *No* label (coloured in green).

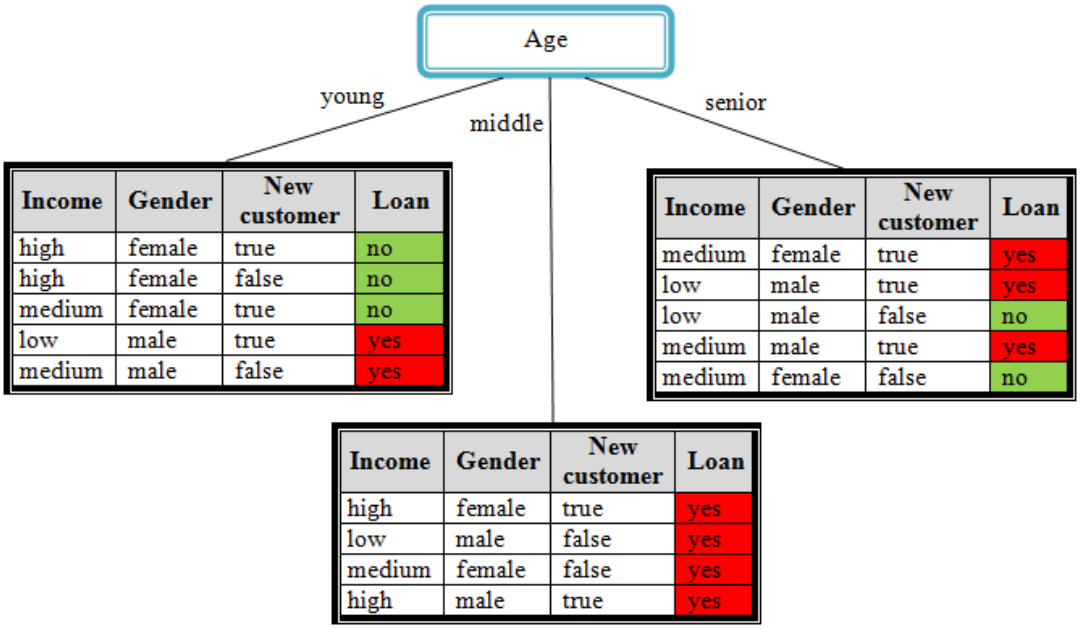


FIGURE 2.3: The three subsets of the data set D obtained after splitting with respect to the different attribute values: *young*, *middle* and *senior*.

2.3.2 Rule-Based Classifiers

Rule-Based Classifiers comprise sets of rules $R = \{r_1, r_2, \cdots, r_n\}$. In this case, R is referred to as a *Rule Base*. The rules have an $X \Rightarrow Y$ format where X is a condition

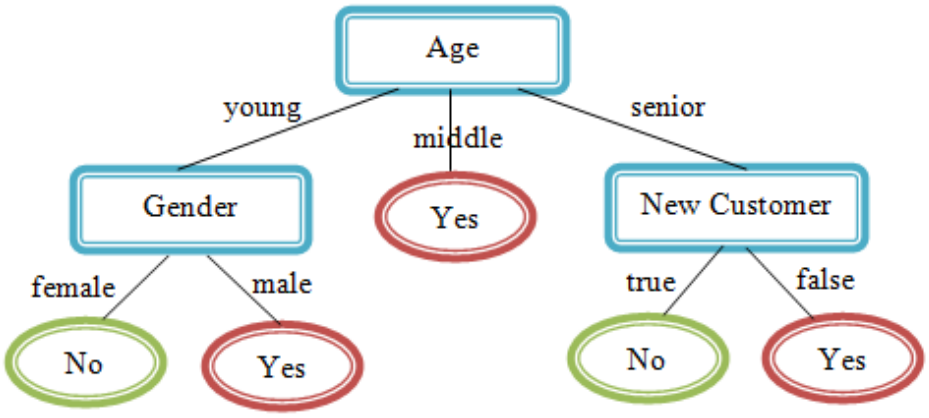


FIGURE 2.4: The DT classifier for the data set D presented earlier in Table 2.1.

(antecedent) and Y is a class label (consequence). The condition can be either a single attribute or a conjunction of attributes. If the attributes of a given new record satisfy the condition X of a rule r then it is said that r covers the record and the label from Y is used to classify the record.

Example
The set of rules described below $\{R1, R2, R3, R4, R5\}$ is an example of a rule based classifier for the data set D presented earlier in Table 2.1:

R1: *if Age=middle* $\rightarrow$ *loan= Yes*
R2: *if Age=young and Gender= male* $\rightarrow$ *loan= Yes*
R3: *if Age=young and Gender= female* $\rightarrow$ *loan= No*
R4: *if Age=senior and New customer = true* $\rightarrow$ *loan= No*
R5: *if Age=senior and New customer = false* $\rightarrow$ *loan= Yes*

Typically, the rules are learned one at a time and added repeatedly to the rule set. During the generation process the records covered by each rule are removed from the data set prior to the next iteration, otherwise the next rule would be identical to previous one. This process is repeated until the entire training data set is covered or the number of the records covered by a new rule is less than some predefined threshold. Further details regarding the construction of rule based classifiers, and the different issues related to the construction process, are beyond the scope of this thesis, for more details see [72, 93, 94, 195].

In the context of the work described in this thesis the RIPPER (Repeated Incremental Pruning to Produce Error Reduction) algorithm (called JRIP in Weka) is used [41]. Using JRIP rules are generated *directly* from the dataset, starting with an empty set of rules which is iteratively added to (more details can be found in [93, 94, 195]). The main

advantages of Rule-Based Classification are: (i) simplicity, (ii) ease of interpretation and generation and (iii) understandability. Note that the set of generated rules are ranked (*ordered*), a new record is assigned the class label of the highest matching ranked rule [195]. Otherwise, a default case is applied.

An alternative method of generating a rule based classifier is to extract rules, of the above form, from a previously generated DT. Different approaches have been proposed to extract and generate rules from decision trees, for more details see [195]. The PART classifier [72] has been adopted with respect to the work described in this thesis as a rule based classifier generated from a partial decision tree in such a way that the "best" leaf obtained in each iteration is used to generate a rule. This type of rule generation is referred to as an *indirect* rule generation because rules are generated as a "by product" of some other process (DT generation).

2.3.3 Bayesian Classifiers

The third type of classifier used for evaluation purposes was a Bayesian classifier. Bayesian classifiers are statistical classifiers based on the well known Bayes probability theorem:

$$P(H|A) = \frac{P(A|H)P(H)}{P(A)} \tag{2.3}$$

where H is a "hypothesis" and A is an "evidence". The probability of the hypothesis H holding given the evidence A is denoted by the conditional probability $P(H|A)$ and is called the *posterior probability*. In the context of classification, A is a vector of n features whereby $A = \{a_1, a_2, \cdots, a_n\}$. Similarly, the conditional probability $P(A|H)$ is the *posterior probability* of A given H. $P(H)$ and $P(A)$ are called the *prior probability* of the hypothesis H and the feature vector A respectively. The prior probability is the distribution probability without considering any event, while the posterior probability considers the event H. Bayes theorem assumes that all the attributes in a feature vector are *independent*, this is why it is sometimes referred to as *Naïve Bayes*; however, the assumption simplifies the calculation. Given a training set of n records $T = \{t_1, t_2, \cdots, t_n\}$ where each t_i comprises of l attributes $t_i = \{a_1, a_2, \cdots, a_l\}$. Suppose that there are m classes where t_i belongs to class k if and only if the posterior probability $P(c_k|t_i)$ has the highest probability value amongst other classes:

$$P(c_k|t_i) > P(c_j|t_i) \qquad 1 \leq j \leq m \quad and \quad j \neq k$$

Recall that $P(c_k|t_i)$ is calculated using Equation 2.3. A Bayesian classifier is constructed as follow [94].

1. Estimate the prior probability $P(c_k)$ of each class, $P(c_k) = \frac{|c_{k,T}|}{n}$, where $|c_{k,T}|$ is the number of tuples in T that belongs to class c_k and n is the total number of tuples in the training set. The prior probability $P(H)$ is constant for all classes.

2. Maximise the posterior probability $P(t_i|c_k)$ by finding the total product of the posterior probability of each attribute in t_i individually as follows[1].

$$P(t_i|c_k) = \prod_{i=1}^{l} P(a_i|c_k) = P(a_1|c_k) \times P(a_2|c_k) \times P(a_3|c_k) \times \cdots \times P(a_l|c_k)$$

3. Finally, calculate $P(t_i|c_k)P(c_k)$ for all m classes. The class c_k that has the highest $P(t_i|c_k)P(c_k)$ value associated with it is selected as the predicted class.

The main limitation of Bayesian classifiers is the Naïve Bayes assumption. However, their main advantages are their simplicity and computational efficiency; they only require a single scan of the training data and provide a fast classification of new unlabelled cases [24, 94].

2.3.4 Artificial Neural Network

The fourth type of classifier generator used for evaluation purposes with respect to the work described in this thesis is an Artificial Neural Network (ANN) classifier. An ANN is a mathematical model inspired by the conjectured operation of the human biological neural system. ANNs have been used with respect to a wide range of real world applications, especially in the context of environments that are continuously changing. Examples include: (i) banking systems for loan decision making [33], (ii) investment decision making [89] and (iii) monitoring and control manufacturing processes [56, 126]. More examples can be founded in [214] which provides a survey of such applications. Typically, an ANN comprises a set of layers: (i) the input layer, (ii) the hidden layer(s), and (iii) the output layer. Each layer consists of nodes ("neurons") and weighted links. The simplest structure (topology) is the input-output layer where there are no hidden layers. Note that although the complexity of an ANN structure increases as the number of hidden layer(s) increases, the effectiveness frequently also tends to increase.

The idea is to use training data to train an ANN so that the link weightings can be learnt starting with initial weights. ANNs thus fall into the supervised learning category where weights are iteratively adjusted in order to minimise the error between the desired output and the predicted output through using a sufficient number of training examples [100]. There are various algorithms that can be used to apply the necessary weight adjustment during training. The *Back-Propagation* (BP) algorithm is a widely used algorithm for learning weights, that has been extensively employed with respect to many applications domains due to its simplicity. Note that if the accuracy of a generated ANN is not acceptable then the ANN can be trained again using either a different structure or new initial weights [177].

[1] In case of continuous attribute values, the Gaussian distribution is assumed along with a mean μ and standard deviation σ to be used for the calculation of probability. $P(x|c_k) = f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$. See [94] for more details.

Figure 2.5 presents a simple example of an ANN (a perceptron). The perceptron, popularised in the 1960s, is an early example of an ANN [94, 177]. The n inputs are given by $X = \{x_1, x_2, \ldots, x_n\}$ and each x_i is connected to the neuron by a weighted link w_i. Typically, the neuron consists of a summation function along with an activation function (sometimes referred to as the "threshold function"). In a simple case, the output y will be activated ($y = 1$) if and only if the summation of x_0 and the weighted inputs exceed a threshold value t as shown in Equations 2.4 and 2.5 and also in Figure 2.5.

$$x_0 + \sum_{i=1}^{n} x_i w_i > t \tag{2.4}$$

$$y = f(x_0 + \sum_{i=1}^{n} x_i w_i) \tag{2.5}$$

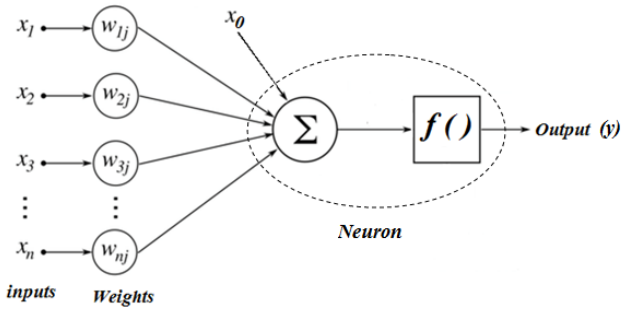


FIGURE 2.5: A typical Perceptron Example.

With reference to Figure 2.5 x_0 is an additional *fixed* input called the *bias* neuron which can exist in more than one layer. x_0 is connected to all neurons in the next layer (but not the previous one). x_0 can be set to any value in the activation function for some specific output. The main goal of x_0 is to provide more flexibility and control for the ANN [94, 177].

Beside the simple perceptron, there are many different types of ANN that have been suggested and proposed. A commonly used type of ANN is the Multilayer Feed Forward Neural Network (FNN) where the flow of information is in only one direction (forward). FNN can be seen as an extension of the perceptron with hidden layers and sometimes it is known as a Multi-Layer Perceptron (MLP) [179].

Despite the accurate prediction that can be obtained using supervised ANN, the main limitation of NN is the large amount of training data required to effectively train them. Moreover, the time complexity for training NN can be significant which makes them unsuited to mining very large data sets [44]. A further criticism sometimes levelled at ANN is that it is a "black box" technique, which means that explanation generation is difficult if not impossible. Nevertheless, ANNs have been included in the five classifier

generation algorithms considered in this thesis for the purpose of evaluating the proposed 3D surface representations.

2.3.5 *k*-Nearest Neighbour

The last classification technique used to evaluate the 3D surface representations proposed later in this theses was the k-NN classifier. k-NN is one of the simplest and most popular non-parametric[1] classification techniques. k-NN was originally proposed by Fix and Hodges in 1951 [67][2] and because of its expensive computation requirements (in terms of both time and storage space) it did not gain popularity until the 1960s when available computing power reached a level whereby the use of k-NN became a realistic option. k-NN is now considered to be one of the most powerful classification techniques available [216]. k-NN operates by finding the most similar k previously labelled records to a new record and using the knowledge of these pre-existing labels to label the new record. The main challenges are: (i) the *similarity* measure to be used, (ii) how to address the situation where the nearest k records have different labels associated with them and (iii) what the "best" value for k is. In the case of the work described in this thesis, as will become apparent later in Chapter 6, $k = 1$ was used thus obviating the need to resolve challenges (ii) and (iii). Records are usually presented using a feature vector representation, in which case similarity between two records can be determined using a simple distance measurements such as the standard Euclidean distance measure (Equation 2.6) or the Manhattan Euclidean distance measure (Equation 2.7). The first is typically used where the distribution is Gaussian, the second where it is Exponential [225].

$$D(x,y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \tag{2.6}$$

$$D(x,y) = \sum_{i=1}^{N}|x_i - y_i| \tag{2.7}$$

The above distance measures require two equal length feature vectors (so that a one-to one matching can be achieved). Thus these distance measures are not suited to all types of data and data distributions [224, 225]. Simple distance measures are also not suited data representations other than feature vector representations. As noted earlier one of the representations proposed in this thesis is the PS representation, this uses a point series (curve) representation. Thus in the context of the work described later in this thesis the k-NN algorithm was used in combination with *Dynamic Time Warping* (DTW) as this allowed for the effective measurement of the similarity between point series [39, 156, 223]. DTW is therefore discussed in more detail in the following section.

[1] The term non-parametric in this context indicates the data or population is assumed to not have any particular distribution.

[2] Later it was republished in [187].

2.3.6 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) is an established technique used to identify the similarity between two point series or sequences that can be represented as a "curve". Originally DTW was used with respect to speech recognition systems where the data was recorded in the form of time series (hence its name) [156, 182], but it can equally well be applied to any point series representation. It also has the advantage that the two series to be compared do not have to be of equal length. Other than hand writing recognition [57] DTW has been successfully applied with respect to a variety of time series applications [164] as well as motion tracking [152]. Further examples of the application of DTW can be found in [20, 77, 122, 124, 182, 211].

In the context of the work described in this thesis, as noted above, DTW is used to compare PS representations of 3D surfaces. Thus DTW is described in detail below. The description highlights three key constraints concerning the operation of DTW:

1. The alignment process of two series (using the concept of a *cost* matrix).
2. The conditions of the alignments process that must be satisfied.
3. The identification of the *optimal* "warping path" (using an *accumulative* cost matrix based on *Dynamic programming*[1]).

Starting with the alignment process, given two series $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ of length $n \in \mathbb{N}$ and $\mathcal{B} = \{b_1, b_2, \ldots, b_m\}$ of length $m \in \mathbb{N}$. Let the feature space be denoted by $\mathcal{F}$, then $a_i, b_j \in \mathcal{F}$, $\forall i \in [1:n]$ and $\forall j \in [1:m]$. Similarity is normally described in terms of some *cost* measure, a *distance* measure $c(a, b)$. Typically a and b are more similar as the cost value becomes smaller. Thus, a cost matrix $\mathcal{M}$ of size $n \times m$, created by *aligning* $\mathcal{A}$ and $\mathcal{B}$, contains the local distance (cost) for each corresponding points in $\mathcal{A}$ and $\mathcal{B}$. We use $\mathcal{M}(i, j)$ to denote the local distance between $a_i \in \mathcal{A}$ and $b_j \in \mathcal{B}$. Equation 2.8 is used to calculate $c(a_i, b_j)$, the *local* distance (cost) to be stored in the elements of $\mathcal{M}$, in terms of the simple Euclidean distance between two points.

$$\mathcal{M}(i, j) = c(a_i, b_j) = \sqrt{(a_i - b_j)^2} \tag{2.8}$$

A warping path $\mathcal{P}$ is a sequence of points $\{p_1, p_2, \ldots, p_s, \ldots, p_k\}$ where $p_s = \mathcal{M}(is, js) \in [1:n] \times [1:m]$, $s \in [1:k]$ and $\mathcal{P}$ is typically subject to several constraints [123, 152]:

1. *Boundary condition*: The warping path starts at the bottom left element $p_1 = \mathcal{M}(1, 1)$ and ends at the top right element $p_k = \mathcal{M}(n, m)$.

[1] Dynamic Programming is a strategy used to solve complex problems by breaking them down into a collection of simpler sub-problems where each sub-problem is solved separately, then the solutions are combined together to generate an overall solution (see [83, 188] for more details).

2. *Monotonicity condition*: $\forall i, \forall j \in \mathcal{P} : 1 \leq 2 \leq \cdots \leq k$. This indicates that both indices i and j are forced to be increased along the warping path $\mathcal{P}$ and this guarantees that $\mathcal{P}$ will not turn back.

3. *Step size condition*: $p_{i+1} - p_i \in \{(1,0), (0,1), (1,1)\}$ for $i \in [1 : k-1]$.

The first condition (Boundary) guarantees that the entire series $\mathcal{A}$ and $\mathcal{B}$ are considered by forcing the warping path to be started at $p_1 = (1,1)$ (lower left corner) and ended diagonally at $p_k = (n,m)$ (upper right corner). The second condition ensures that the warping path would never be in the backward direction which means that the possibility to repeat any feature is zero and this is consistent with the *time* warping path definition. The last condition (Step size) ensures the *continuity* of the generated path $\mathcal{P}$ in terms of the elements in $\mathcal{A}$ or $\mathcal{B}$; in other words that the path is comprised of adjacent cells (including diagonally adjacent cells). Figures 2.6, 2.7, 2.8 and 2.9 presents some example warping paths.

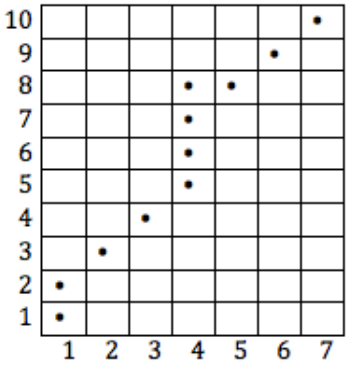


FIGURE 2.6: A warping path, $\mathcal{P}$, that satisfies the (i) *Boundary*, (ii) *Monotonicity* and (iii) *Step size* conditions.

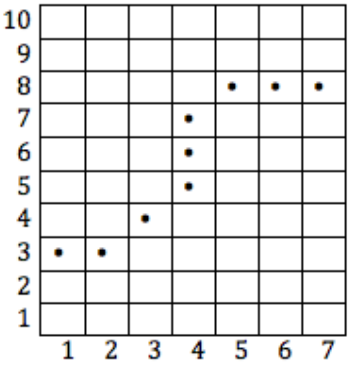


FIGURE 2.7: A warping path, $\mathcal{P}$ where the *boundary* condition is violated.

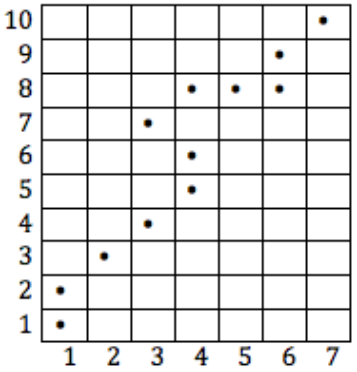


FIGURE 2.8: A warping path, $\mathcal{P}$ where the *monotonicity* condition is violated.

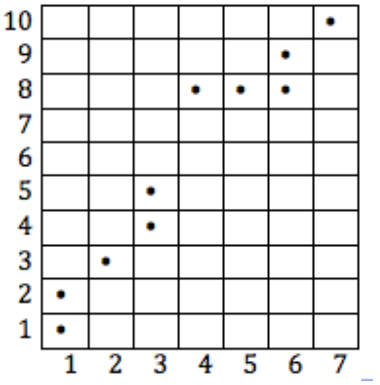


FIGURE 2.9: A warping path, $\mathcal{P}$ where the *step size* condition is violated.

Many different warping paths can be generated between $\mathcal{A}$ and $\mathcal{B}$ (Figure 2.10) where the total cost of a warping path, $c_p(A, B)$, is calculated using Equation 2.9. However, we

are interested only in the *optimal* (shortest) warping path which minimise the *warping cost*[1] as shown in Figure 2.10 where the optimal path is highlighted in red. Note that if $\mathcal{A}$ and $\mathcal{B}$ are identical the warping path through the matrix would be along the diagonal. We indicate the minimal warping cost using the notation $c_p^*(A, B)$. It should be noted that if $\mathcal{A} = \mathcal{B}$ are identical, then $c_p^*(A, B) = 0$.

$$c_p(A, B) = \sum_{s=1}^{k} c(a_s, b_s), \qquad \forall (a_s, b_s) \in \mathcal{P} \tag{2.9}$$

Determining *all* the warping paths between $\mathcal{A}$ and $\mathcal{B}$ is an exhaustive process and may require *exponential* computational cost. Instead we generate an optimal path dynamically (hence "dynamic" time warping) by stepping through the matrix in such a way as to minimise the cost.

Recall that the value of $\mathcal{M}(i, j)$ as defined earlier is represented in terms of Euclidean distance, where $\mathcal{M}(i, j) = c(a_i, b_j)$. However, the value of $c(a_i, b_j)$ is actually comprised of: (i) the local cost (distance) between the corresponding points of the series and (ii) the *minimum* cost of the adjacent neighbours. The local cost of $\mathcal{M}(i, j)$ is typically defined as the distance between the corresponding elements of the two series $\mathcal{A}$ and $\mathcal{B}$; for example, the *local* cost (distance) between a_i and $b_j$is $\mathcal{M}(i, j) = c(a_i, b_j) = |a_i - b_j|$. However, the *minimum* cost of the adjacent neighbours of $\mathcal{M}(i, j)$ is calculated with respect to the "positions" represented by the elements within $\mathcal{M}$. The minimum distance is defined recursively amongst the adjacent elements using a dynamic programming approach. Given $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ of length $n \in \mathbb{N}$ and $\mathcal{B} = \{b_1, b_2, \ldots, b_m\}$ of length $m \in \mathbb{N}$, then the generation process of $\mathcal{M}$ is as follows:

1. Firstly, the value for $\mathcal{M}(1, 1)$ is identified by calculating the distance between a_1 and b_1. Thus, $\mathcal{M}(1, 1) = c(a_1, b_1) = |a_1 - b_1|$.

2. The values of the $\mathcal{M}(1, j)$ of the first *row* are calculated using the local cost $c(a_1, b_j) = |a_1 - b_j|$ and the value of previous element $\mathcal{M}(1, j - 1)$ as shown below:

$$M(1, j) = |a_1 - b_j| + \mathcal{M}(1, j - 1)$$

3. The values of the $\mathcal{M}(i, 1))$ of the first *column* are calculated using the local cost $c(a_i, b_1) = |a_i - b_1|$ and the value of previous element $\mathcal{M}(a_{i-1}, b_1)$ as shown below:

[1] In some special cases, the optimal path is not *unique*. For more details and examples see [152].

$$\mathcal{M}(i,1) = |a_i - b_1| + \mathcal{M}(i-1,1)$$

4. The cost for each remaining element $\mathcal{M}(i,j),\ \forall i \in (1:n]$ and $\forall j \in (1:m]$, is calculated using the local cost $c(a_i, b_j)$ and the minimum cost of adjacent neighbours $\mathcal{M}(i-1,j), \mathcal{M}(i,j-1), \mathcal{M}(i-1,j-1)$ as shown below:

$$\mathcal{M}(i,j) = |a_i - b_j| + min\{\mathcal{M}(i-1,j), \mathcal{M}(i,j-1), \mathcal{M}(i-1,j-1)\}^1$$

   where *min* indicates the minimum value of the three adjacent cells: $\mathcal{M}(i-1,j)$, $\mathcal{M}(i,j-1)$ and $\mathcal{M}(i-1,j-1)$. An example of these adjacent cells are clearly shown in Figure 2.10 coloured in green in the shaded block.

Finally, the value of the *optimal* warping path, $\mathcal{P}$, with respect to $\mathcal{A}$ and $\mathcal{B}$ is located at $\mathcal{M}(n,m)$. Thus, $DTW(A,B) = c_p^* = \mathcal{M}(n,m)$.

By employing dynamic programming to calculate the minimum warping path costs the time complexity of DTW is $O(n \times m)$. Nevertheless, further efficiency improvements can be made to enhance the performance of DTW by considering *global* constraints besides the local constraints (described earlier). The idea is to consider only a portion of the $\mathcal{M}$, using a *windowing concept*, because we know that the optimum warping path will be located along the diagonal. The idea was first proposed to prevent undesirable behaviour, such as when a single point in one series matches with many points on the other series (a one-to-many mapping) resulting in a piece of horizontal or vertical warping path directed away from the diagonal [45, 152]. However, the use of windows also provides for efficiency gains. The *Sakoe-Chiba band* (S-C band)[2] is one of the most frequently used windowing methods. The idea is based on the observation that $\mathcal{P}$ typically runs close to the diagonal of $\mathcal{M}$ by w units, where $|i-j| \leq w$ and $w > 0$ [181]. Therefore, the diagonal will be bounded by $j = i + w$ (upper limit) and $j = i - w$ (lower limit) which means that only part of the cost matrix $\mathcal{M}$ needs to be completed. Figure 2.11 shows an example of *Sakoe-Chiba band* windowing. DTW can utilise the windowing concept to improve the run time complexity to $O((n) \times w)$ [108, 147, 184]. Further details on different efficiency enhancements for DTW and their implications can be found in [122–124, 152]. With respect to the work described in this thesis the Sakoe-Chiba band expedient was adopted (see Chapter 6).

[1] *Lexicographic* ordering is adopted if *min* is not unique.

[2] It has been found that the Sakoe-Chiba band works well when $n \approx m$ [84, 122].

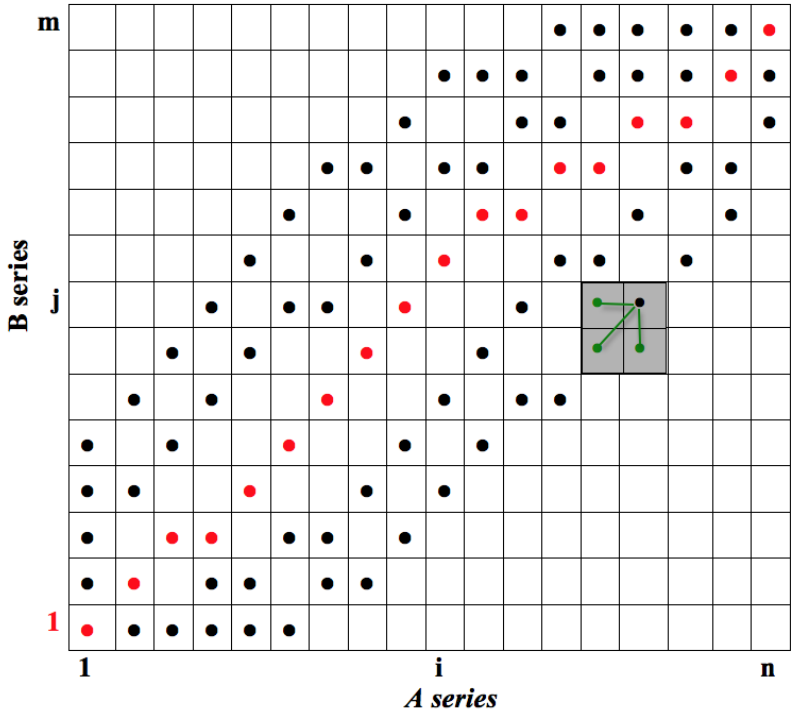


FIGURE 2.10: Different warping paths can be defined between two point series $\mathcal{A}$ and $\mathcal{B}$. The *optimal* warping path is in red colour. The distance values stored in the adjacent cells $\{M(i-1,j), M(i,j-1), M(i-1,j-1)\}$ (shaded block) are used to identify the value of $M(i,j)$ elements (black and red points).

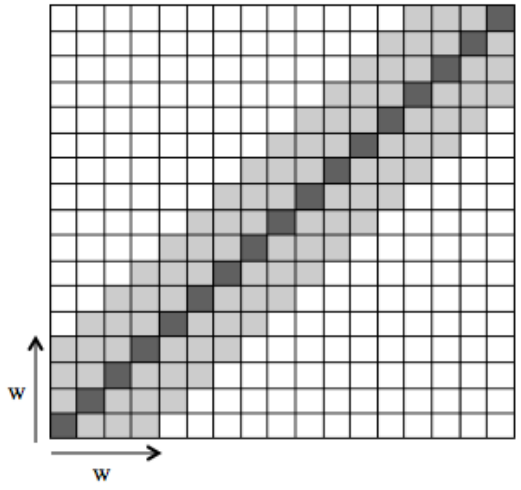


FIGURE 2.11: An example of the *Sakoe-Chiba band* windowing concept with $w = 3$. Only the shaded elements are considered when determining the optimum warping path.

2.4 3D Surface Representation Techniques

Techniques for representing 3D surfaces have been widely used in various application domains such as: medicine and bioinformatics [48, 175, 201, 221], manufacturing [98, 107] and especially computer graphics with respect to animation and video gaming [101, 106].

The most straight forward representation for a given 3D surface (or object) is a *point cloud* [127]. A point cloud is simply a collection of unordered and unorganised data points, $P = \{p_i = (x_i, y_i, z_i) \mid 1 \leq i \leq n\}$. A variety of techniques are available to generate such point clouds, these include: (i) laser and optical scanners, (ii) 3D digitizers (coordinate measuring machines), (iii) some Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) systems and (iv) range data converters[1] such as in [5, 203]. In the case of the work described in this thesis the C_{in} input clouds were generated using a CAD system and the C_{out} output clouds using an optical measuring system[2].

Although point cloud representations are simple there is often a need, depending on a particular application domain, for a higher level representation. A wide range of 3D representation techniques have been proposed. An overview of these different representations is presented in this section. These are categorised as follows: (i) Mathematical representations (Parametric and Implicit representation), (ii) Mesh representations and (iii) Other representations such as CSG, B-Rep and Voxel representations. Each is discussed in further detail below including their advantages and limitations. Note that in the context of the work described in this thesis we are interested in 3D representations that are not only able to capture the geometrical information contained in a given surface but also support the application of classification techniques.

2.4.1 Mathematical Representation

Mathematical 3D surface representations play a significant role in the context of industrial applications where they are typically used, in connection with CAD software, to build 3D models of some object. Mathematical representations also often provide the foundation for other representations. With respect to the work described in this thesis mathematical representations are utilised in Chapters 3 and 5. In Chapter 3 it was adopted for the purpose of springback calculation (given a before and an after surface), whilst in Chapter 5 it was adopted for the purpose of identifying "critical features" (corners and edges) within a given 3D surface. More detail concerning the particular mathematical representation process used is presented in Section 2.4.4. Mathematical 3D representations can be further divided into two categories: (i) Parametric and (ii) Implicit. Each is discussed in the following two subsections.

[1] Range data is 2D data where each "pixel" value describes the distance between the points in a 3D scene (object) to a specific point such as a camera. It is sometimes considered to be a special form of 3D data and referred to as "2.5D" data

[2] The actual system used was the GOM (Gesellschaft fur Optische Messtechnik).

Parametric Representations

A parametric representation is a mapping from $\mathbb{R}^2(u,v) \rightarrow \mathbb{R}^3(x,y,z)$ where a vector valued function $f(u,v)$ comprised of two variables u and v is applied as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x(u,v) \\ f_y(u,v) \\ f_z(u,v) \end{bmatrix} \tag{2.10}$$

where $u_{min} \leq u \leq u_{max}$ and $v_{min} \leq v \leq v_{max}$ are surface parameters and x, y, z are the coordinates of a point located on the surface as shown in Figure 2.12. The domain can be defined more generally when u and v are normalised as $(0 \leq u, v \leq 1)$ [30].

The parametric surface representations tend to be simple representations. In [176] it is noted that "parametric surfaces are generally easier [than implicit surfaces] to draw, tessellate, subdivide, and bound, or to perform any operation on that requires a knowledge of *where* on the surface". In [133] their simplicity with respect to the modification of object shapes is noted. Due to this simplicity parametric representations have been widely utilised: (i) as a surface construction technique [113] where by curve fitting and smoothing processes are applied to support *visualisation* [99] and (ii) for inferring new ranges of data points for regions that are either sparsely represented or do not contain any data points.

Given the normal vector $\vec{n} = \langle a, b, c \rangle$ of a point $p_0 = (x_0, y_0, z_0)$, then the line along the normal n can be defined using a parametric equation which can then be used for further processing (as in the case of the work described in Chapters 3 and 5). However, it is usually not feasible to describe an entire 3D object using parametric mathematical representations because [60, 133]: (i) it is difficult to determine whether the position of a given point is inside or outside a given volume, and (ii) it is difficult to combine multiple "patches" to form a complex shape due to smoothness problems that may arise at patch boundaries. So we can say that the parametric representations are best suited to representing interesting geometrical features (such as lines) but not an entire 3D surface.

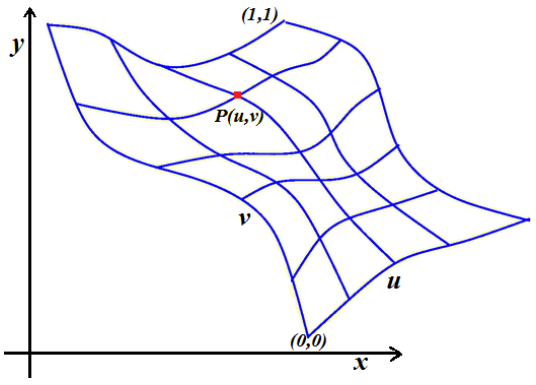


FIGURE 2.12: A parametric surface.

Implicit Representations

An implicit 3D surface representation is defined as a mapping $\mathbb{R}^3(x, y, z) \rightarrow \mathbb{R}$ given by the following function of three variables:

$$f(x, y, z) = 0 \tag{2.11}$$

Given a point (x_0, y_0, z_0), with a normal $\vec{n} =< a \;\; b \;\; c >$, located on a plane, the implicit representation of the plane is given by:

$f(x, y, z) = 0$
$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0$
$ax - ax_0 + by - by_0 + cz - cz_0 = 0$
$ax + by + cz = ax_0 + by_0 + cz_0$

Now, let $d = ax_0 + by_0 + cz_0$ and thus we have:

$ax + by + cz = d$
$ax + by + cz - d = 0$ (2.12)

Equation 2.12 should be satisfied by each point (x, y, z) located on this plane. The coefficients of the implicit surface (a,b,c,d) indicate that the value of function f for the point (x, y, z) gets close to zero. These coefficients can be estimated [198]. The main advantage of implicit surface representations is the ease with which they can be used to indicate the location of a given point with respect to a 3D surface as shown in Equation 2.13. More specifically they are well suited to describing topological information (information about connected components). This property is used, with respect to the work described in this thesis, to determine springback magnitude (as described in Chapter 3).

$$\begin{cases} f(x, y, z) \geq 0 & \text{if the point is on/outside the surface.} \\ f(x, y, z) < 0 & \text{if the point is inside the surface.} \end{cases} \tag{2.13}$$

However, implicit mathematical 3D representations are not well suited to identifying points on a surface, although they are well suited to identifying whether particular locations are "inside" (below) or "outside" (above) the surface (this is why the representation is referred to as being implicit). Implicit representations are thus well suited to *raytracing* (for more details about raytracing see [71]).

One of the most popular examples of an implicit mathematical representation is the Isosurface representation [143]. An Isosurface (*isos* in Greek translates to "the same") is defined as the surface that contains all points that feature the same scalar value in the context of volumetric 3D data [200] (thus a contour in 2D). The Isosurface representation

is employed, for example, in the medical field to represent a region (object) of interest in medical images.

The main disadvantage of implicit mathematical representations is that there is a "trade off" between efficiency (speed) and effectiveness (quality); this is particularly significant in the context of visualisation. The solution is to transform it into some other format that does support visualisation (for example a *mesh* representation) as described in the following section. In the context of the work described in this thesis an implicit representation is used to identify the point where the extension of a normal $\vec{n}$ to a given point $p_0 = (x_0, y_0, z_0)$ on an input surface intersects the output surface as will become clear in Chapter 3.

2.4.2 Mesh (Polygonisation) Representations

A Mesh representation, of a given 3D object, is defined in terms of a collection of 0-dimensional cells (vertices), 1-dimensional cells (edges) and 2-dimensional cells (facets) in R^3. Thus, an object is defined by a pair of ordered lists, $\langle \mathbb{P}, \mathbb{V} \rangle$, where: (i) $\mathbb{V} = \{v_1, v_2, \ldots, v_n\}$ is a set of n vertices defined in terms of a 3D coordinate system where $v_i = (x_i, y_i, z_i)$ and (ii) $\mathbb{P} = \{p_1, p_2, \ldots, p_m\}$ is a set of polygons described in such a way that $p_i = v_{i1}, v_{i2}, \ldots, v_{ij}$ is the polygon comprised of j vertices (thus if $j = 3$ we would have a triangular polygon [30]). The $\mathbb{P}$ is normally used to describe the topological information (information about the connectivity between $\mathbb{V}$). In other words, the mesh can be seen as *unstructured grid*.

Constructing a mesh representation from a point cloud is a well known problem in the field of 3D geometrical modelling referred to as "surface reconstruction". Surface reconstruction has been widely investigated in the context of various domains such as reverse engineering (reconstructing a representation given a visualisation of a 3D object) and face recognition. Alternatively the polygons that frequently make up mesh representations can be generated from parametric mathematical representations [129] and from implicit mathematical representations [158]. Recall that the representation techniques with respect to this thesis should facilitate surface comparison (as well as classification). Mesh representations inherently support such comparisons. They also support fast processing, especially in the context of current graphics system that support high quality visualisation. The popularity of mesh representations comes from their simplicity and high manageability in 3D design (easy to define and modify). From this point, and with respect to the work described in this thesis, the advantages of the mesh representation are incorporated into a uniform equal grid representation where the centre points of these grids represent the set $\mathbb{V}$. Using a regular grid the connectivity information can be easily attained, therefore the regular grid (mesh) representation adopted in this thesis is considered to be a special case of the mesh representation without the set $\mathbb{P}$. More specifically the grid representation was adopted because:

1. It was compatible with the CAD/CAM systems used to generate desired shape descriptions.

2. It was simple to generate and process.

3. When combined with other representation forms it readily supports edge detection (as will be demonstrated later in this thesis).

4. It facilitates the capture of local geometries (again as will also be demonstrated later in this thesis).

5. It provides a useful mechanism whereby geometrical information can be stored (and subsequently used or reused).

6. It provides flexibility (it supports translation into other forms).

As noted above details of the grid (mesh) representation used throughout this thesis will be presented in Chapter 3. However, grid representations have been used elsewhere, for example with respect to *texture analysis* in the context of satellite image processing [85, 120] and *terrain analysis* in Geographical Information System (GIS) [145]. In the field of texture analysis Local Binary Patterns (LBPs) [159] are a popular grid based technique that has been used to describe effectively the local structure of a given image. Though originally proposed in the context of texture analysis LBPs have been adopted more widely, such as for remote sensing [53] and for face recognition [4]. With respect to the work described in this thesis the LBP idea was adopted with respect to the Local Geometry Matrix (LGM) representation technique described later in Chapter 4. In the field of GIS (systems concerned with the integration of geographically referenced information) map information is typically included in either a Vector or Raster format. The Vector format is essentially a parametric mathematical representation. However, the Raster format is a uniform cell-based method which is essentially a (fine grained) grid representation.

Alternative mesh representations, to that proposed and utilised in this thesis, have been adopted elsewhere with respect to research related to AISF, such as the work presented in [137, 150]. The effectiveness of mesh representations can be influenced by surface complexity. However, in the case of the 3D objects considered in this thesis, objects that can be produced using AISF, this is not an issue.

One last observation in the context of mesh representations, and by extension grid representations, is that they can be conceptualised as graphs. Although outside the scope of the work described in this thesis there is some current work, in which the author has participated, looking at the application of graph mining techniques to facilitate springback prediction. More specifically the concept of Vertex Unique Labelled Subgraphs as described in [226–228].

2.4.3 Other 3D Surface Representation Techniques

There are other 3D Surface Representation Techniques that have not been included in the above categorisation that have been extensively used for *solid* modelling. These include:

Constructive Solid Geometry (CSG) [213], Boundary Representation (B-Rep) [174] and Voxel Representation [85, 120]. CSG and B-Rep are typically used to represent 3D solids in terms of their outer "shell", whilst Voxel representations are used to represent both the "interior" and "exterior" of a given 3D solid. Voxel representation is mostly applied to volumetric data especially in the medical field, for more information see [6, 25, 120]. However, all of these types of representation are typically concerned with aspects of visualisation such as: (i) object recognition and (ii) the detection of deformations with respect to shape reconstruction processes. These types of representations are therefore not well suited to surface representation for springback prediction. As volumetric data representation is beyond the scope of this thesis, our review of this sub-area terminates here.

2.4.4 Overview of Critical Feature Techniques

This section presents a general overview of the techniques that have been proposed, and extensively employed, to extract the "critical" features for a given 3D surface with respect to different applications. The critical features (sometimes called "sharp" feature) are the corners and edges within a given 3D surface. In the context of grid representations we refer to "critical points", grid centre points that represent edges or corners. The Local Distance Measure (LDM) representation presented later in this thesis uses the concept of critical points (see Chapter 4). Point cloud representation and mesh representation are two examples of different 3D representations. Examples of detection techniques used with each approach are discussed in further detail below.

Generally speaking, critical feature detection and extraction from 3D surfaces is an area of research interest with respect to different applications such as: (i) 3D imaging and modelling [105], (ii) movement tracking [28], (iii) image matching [80] and (iv) image recognition [8, 46, 118, 139, 160]. It also finds application with respect to domains such as the automotive part manufacturing industry where it is used for fault detection in assembly line processes [165, 220]. From the listed applications, we can argue that the accurate detection of critical features is of primary interest for most of these applications. From the above we have seen that there exists a wide variety of 3D representation techniques. There is a similar range of critical feature detection techniques many dependent on the nature of the adopted 3D representation. In the following the various critical feature detection methods have been grouped according to whether they are: (i) cloud based, (ii) mesh based or (iii) normal analysis based.

In the context of point cloud representations, the lack of information concerning connectivity and neighbourhoods in the point cloud makes direct detection of critical features a challenging task [32]. However, there is some reported research directed at extracting critical feature points from point clouds, examples include [87, 127, 210, 231]. In Gumhold et al. [87] the authors generated a local neighbourhood graph for each point in the point cloud, then identified critical points using *eigenvector* and *eigenvalue* analysis. The technique attempted not only to detect the critical points but also to

classify effectively the point according to whether they were: edges, crease lines, corners or flat surfaces. This work was extended and improved in [165] by the inclusion of better surface smoothing and noise reduction techniques. In Zhao et al. [231] a technique to extract edges (and corners) from point clouds was proposed whereby the point cloud is first converted into a 2D image format where the z values are used as the image pixel value. Well known 2D image edge extraction techniques can then be applied, such as: Canny edge detection [31], the Sobel operator [171] and the Prewitt operator [170]. However the approach is expensive in terms of computation time and storage requirements [142]. Moreover, using the image processing detection techniques with point cloud representations has the disadvantages that: (i) it requires the user to specify certain parameters, (ii) it is difficult to know the real location of points belonging to edges and (iii) it tends to be very sensitive to the presence of "noise".

With respect to mesh representations, as noted above, these are popular in the context of CAD/CAM systems. Many mesh based critical feature detection techniques have been proposed such as [70, 103, 161, 196]. Most of these mesh based techniques are founded on *curvature* analysis. Curvature describes the *shape of a local region* within a given 3D surface and thus it is often used as a local shape descriptor [9]. A number of mechanisms have been proposed for determining curvature. For example there is reported work where the mesh representation is converted into a mathematical surface representation (an "implicit representation") and *differential* geometry employed to calculate the curvatures such as in the case of [172]. Further discussion regarding different types of curvature analysis, and their computation and estimation, can be found in [7, 163]. Eigen analysis has been extensively used in combination with curvature analysis to determine the *direction of bending* for a given 3D surface, see for example [162, 193]. Despite the accurate detection of critical features offered by the combination of curvature analysis and the eigen analysis, the major limitation is the complex nature of the required mathematical calculation founded on the second derivative of a surface; this is difficult to calculate in the context of 3D surface representation other than where specific mathematical representations are used [163].

The Identification of critical features based on the calculation and comparison of normals with respect to "neighbourhood areas" has been reported by a number of authors [151, 229, 232]. The basic idea is the consider the angle between the normals of adjacent points, if this angle is above some threshold a critical feature can be said to exist. The ease and simplicity of the approach are its main advantages. Some authors (for example [49]) recommend this type of critical feature detection. The nature of the generic grid representation proposed with respect to the work described in this thesis is will suited to critical feature identification using normal analysis because:

- The surrounding neighbourhood of a given grid point can be clearly and easily identified.
- By ordering the neighbours of a given grid point in (say) a clockwise direction, $\{N, NE, E, SE, S, SW, W, NW\}$ this facilitates: (i) normal calculation using the

cross product approach, and (i) identification of the direction of the calculated normals using the "right hand rule".

The normal analysis approach to critical feature selection was thus adopted. Further detail concerning the adopted normal analysis critical feature detection method is presented in Chapter 5 and thus will not be considered further here.

2.5 Evaluation Criteria

The main aim of the work presented in this thesis was to identify the most appropriate 3D representation technique with which to model 3D surfaces in the context of spring-back prediction with respect to sheet metal forming processes such as AISF. To identify this representation the proposed representations were evaluated individually and comparatively. More specifically the conducted evaluation was as follows:

- **Individually** for each technique (discussed separately in each relevant chapter) using *accuracy* and *Area Under ROC Curve*(AUC) as the performance measures.
- **Comparatively**, first by comparing collated accuracy and AUC values, and then statistically by applying the Friedman and the Nemenyi tests to demonstrate whether there was a statistically significant difference between the operation of the proposed techniques.

This section presents an overview of the evaluation measures used with respect to the individual evaluations (accuracy and AUC). Details concerning the adopted statistical evaluation will be presented later in Chapter 8.

The most fundamental mechanism for analysing classifier performance within the data mining community is the confusion matrix where each instance can be classified as belonging to class X or $\neg X$[1] as shown in Figure 2.13. With reference to Figure 2.13 the True Positives (TP) are the number of instances that are correctly classified as belonging to class X, the False Negatives (FN) are the number of instances belonging to class X that are erroneously predicted as belonging to class $\neg X$ class, the True Negatives (TN) are the number of instances that are correctly classified as belonging to class $\neg X$ and the False Positives (FP) are the number of instances belonging to class $\neg X$ that are erroneously predicted as belonging to class X. Frequently used measures that may be derived from a confusion matrix are accuracy, sensitivity and specificity. These metrics are defined in Equations 2.14, 2.15 and 2.16 below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{2.15}$$

[1] A one against all binary classifier has been adopted with respect to the work described in this thesis.

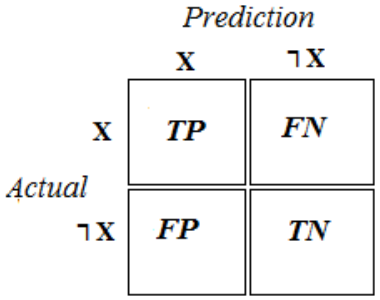


FIGURE 2.13: Confusion matrix.

$$specificity = \frac{TN}{FP + TN} \tag{2.16}$$

Accuracy is an overall indicator of the quality of a classifier although it does not take into consideration the distribution of the classes ("class priors"). Sensitivity is an indicator of the ability of the classifier to identify the positive instances (X), while specificity reflects the ability of the classifier to identify the negative instances ($\neg X$). The Area Under a Receiver Operating Characteristic (ROC) Curve (AUC) [26, 97] is used extensively in this thesis for evaluation purposes. Broadly, the ROC curve concept was originally used in signal detection theory to depict the trade-off between hit rates and false alarm rates [58]. The "hit rate" is called the True Positive Rate (TPR), benefit or sensitivity; while the "false alarm rate" is called the False Positive Rate (FPR), or cost. Both are expressed in the form of a real number ranging from between 0.0 and 1.0. TPR and FPR are calculated as shown in Equation 2.17 and 2.18 respectively. Spackman [61] illustrated how the ROC curve can be used to evaluate the performance of a binary classifier. A ROC curve is generated by plotting the FPR against the TPR (with the FPR plotted along the X-axis and the TPR along the Y-axis). In the ROC space, the best classification performance exists in the upper left corner (FPR=0 and TPR=1) while the diagonal represents random classification (guessing). Therefore, a "good" ROC curve is one that reaches the upper left corner. Figure 2.14 shows four different ROC curves, each curve representing the operation of a classifier. From the figure, it can be seen that curve A is the best curve as it has the highest TPR over the other curves. Curves B, C and D are all below curve A. Curve C represents a classifier that operates in a completely random manner. The Area Under a ROC curve (AUC) is a single value frequently used to measure classifier performance ($0 \leq AUC \leq 1$). In other words AUC is an indicator of the probability that a classifier will correctly classify instances [15, 69, 109, 135]. Note that an AUC value of 0.5 indicates a random classifier (guessing).

$$TPR = \frac{TP}{TP + FN} = sensitivity \tag{2.17}$$

$$FPR = \frac{FP}{FP + TN} = 1 - specificity \tag{2.18}$$

For example, consider a 2-class problem where class 1 has 990 instances and class 2 has 10 instances, then the accuracy of the generated model would be $\frac{990}{1000} = 99\%$ as long as each new instance will be labelled with the majority class (class 1, in this case). However, a classifier that does this is clearly not a good classifier. The main advantage of AUC is its ability to deal with unbalanced data sets since it considers the distribution of classes (TPR and FPR values) [61]. Therefore, AUC was chosen to be one of the performance evaluation measures with respect to the proposed mechanisms presented in this thesis because of the uneven error (springback) distributions within the evaluation datasets. The Mann-Whitney-Wilcoxon (MWW) statistical method, which employs a *ranking* concept based on the signal detection theory proposed by [95], was used with respect to the work described in this thesis to calculate AUC values[1]. A full example on how to calculate the AUC value based on the MWW statistic, is presented in Appendix C.

Ten Cross Validation (TCV) [189] was also adopted with respect to the conducted evaluation in order to reduce the *overfitting* problem and to ascertain the validity of the generated classifiers [73]. Overfitting mainly occurs when a generated classifier (model) fits the data set exactly and in a perfect manner. TCV is used in order to limit the implication of overfitting [23]. TCV is a well established technique for evaluating the performance of supervised learners whereby the data is divided into ten parts so that class labels are distributed equally (stratified). Using the TCV technique the learner is applied ten times, each time to a different $9/10$ of the data set, and tested using the remaining $1/10$. On completion, the recorded results of the ten iterations are used to compute an averaged set of results.

2.6 Summary

This chapter has presented the background to the work presented in this thesis. The chapter covers three main areas: (i) sheet metal forming processes, (ii) data mining (classification techniques) and (iii) 3D surface representation. Recall that springback is the major cause of deformation in AISF that affects the final geometry of the shape produced and that springback prediction was the main motivation of the work described in this thesis. Therefore, the chapter commenced with a general description for the springback phenomena within the context of the AISF process. Then an overview of the KDD process, and data mining in particular, was presented including reviews of the classification techniques used for evaluation purposes with respect to the work described in this thesis. The main 3D representation techniques that are the foundation of the thesis work were discussed next. In the context of this thesis the proposed 3D surface

[1]The AUC/ROC calculation conducted using Weka is also done using the Mann Whitney statistic [215].

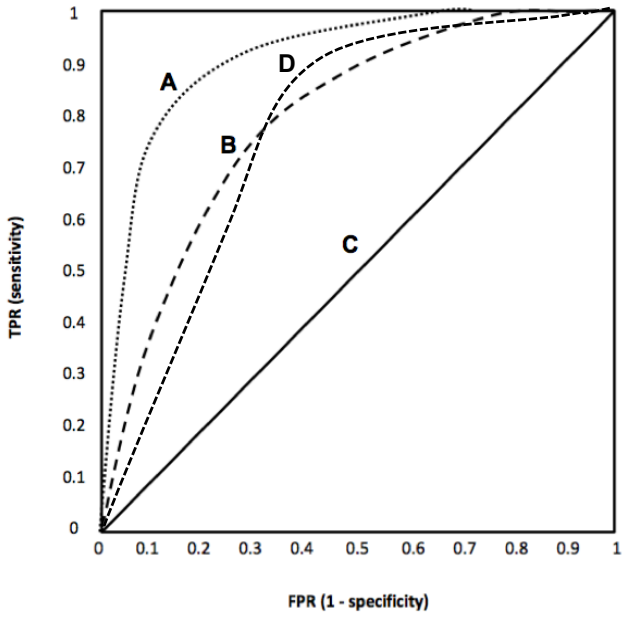


FIGURE 2.14: Four different example ROC curves (A, B, C and D). Curve C is the curve produced as a result of simply guessing. Curve A is said to dominate B, C and D since A is above and to the left of B, C and D. However, B and D do not dominate each other therefore the AUC is a convenient way to compare their performance [135].

representations were not only required to capture effectively geometrical information, but also to facilitate the classification task. Finally the criteria used to evaluate the operation of the proposed techniques was presented. It was noted that two types of evaluation were conducted: (i) *individual* evaluation for each proposed technique using accuracy and AUC measurements and (ii) *overall* performance evaluation using statistical approaches. Only the first was considered in this chapter, the latter will be presented separately in Chapter 7. The following chapter describes the necessary data preprocessing that needs to be applied to the AISF data sets used for evaluation purposes.

Chapter 3

The Grid Representation, Error Calculation Mechanism and the RASP Framework

3.1 Introduction

This chapter describes the generic Representation And Springback Prediction (RASP) framework used to generate springback classifiers with respect to the different proposed techniques described later in this thesis. The chapter also presents the grid representation and error calculation mechanism required to support the different proposed surface representation techniques, the labelling process and the data sets used for evaluation purposes. The grid representation provides a standard input format with respect to the different surface representation techniques. As will be seen, the grid representation also facilitates error (springback) calculation. The error calculation mechanism comprises three steps: (i) normal calculation, (ii) intersection point calculation and (iii) error calculation where both the magnitude and the direction of the error is determined. The Representation And Springback Prediction (RASP) framework on which the proposed surface representation techniques are founded is comprised of three processes: (i) data preprocessing and error calculation (as described in this chapter), (ii) surface representation and (iii) classifier generation.

Given the above the rest of this chapter is organised as follows. Section 3.2 presents a detailed overview of the proposed RASP framework. Then the grid representation generation process is described comprehensively in Section 3.3. A full description of the error calculation mechanism is then given in Section 3.4. Section 3.5 presents the process of labelling for the attributes and the class (springback) attribute which is an essential pre-processing step for the proposed 3D surface representation techniques. A full description of the evaluation data sets is presented in Section 3.6. Finally, the chapter is concluded with a summary in Section 3.7.

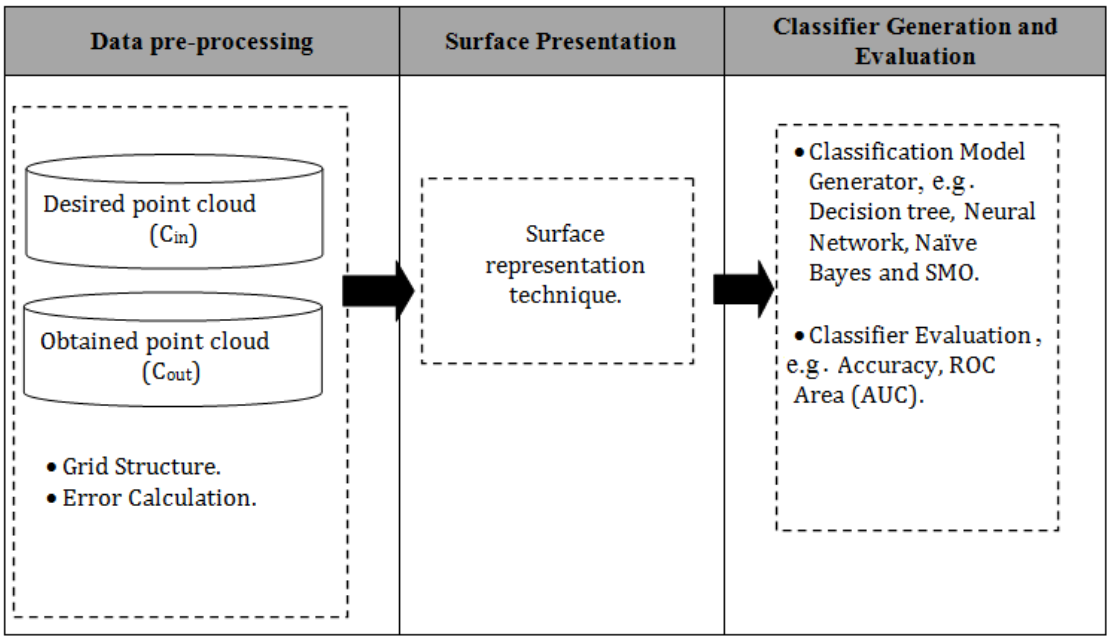


FIGURE 3.1: Schematic describing the Representation And Springback Prediction (RASP) Framework.

3.2 The Representation and Springback Prediction (RASP) Framework

The research methodology adopted with respect to this thesis was to investigate a number of techniques in order to find the *best* surface representation method with respect to "surface" error prediction (classification). Each of the proposed techniques, at least at an abstract level, was founded on the same set of high level processes. In other words, the proposed techniques subscribe to a general framework called the Representation And Springback Prediction (RASP) Framework. A schematic for the framework is presented in Figure 3.1. In the case of our sheet metal forming application the input to RASP will be the C_{in} and C_{out} point clouds as indicated in the figure, however for other prediction applications other forms of input may be used. From Figure 3.1 it can be seen that the proposed framework comprises three main phases: (i) data pre-processing, (ii) representation and (iii) classification. The first phase is concerned with generating the advocated grid representation with associated error values. During the second data representation phase the grid representation is translated into one of the proposed surface representation techniques described later in this thesis. The third phase comprises classifier generation and evaluation. Once an appropriate classifier has been generated it can be used to predict grid labels associated with unseen grids. In the case of sheet metal forming applications these will be the springback errors associated with new shapes to be manufacturer (provided that the same material and process as that used to build the model is used). Later in this thesis two Intelligent Process Models (IPMs) will be proposed whereby this can be achieved.

3.3 Grid Representation

A grid is a simple organisational structure used to represent a 2D space by subdividing the space into equally spaced horizontal and vertical bands. Grid representations can be equally well applied to 2D and 3D space. The main reasons for using a grid representation are as follows [113, 144, 168]:

1. To minimise the density of the point clouds (the required input to RASP).
2. To minimise the computation time (as a result of 1).
3. To create a representative sample of a given shape.
4. To perform dimensionality reduction (2D instead of 3D).
5. To represent the shape using fewer parameters than when using all available data.
6. To permit straightforward further processing.
7. To obtain an integrated and unified framework for both C_{in} and C_{out}.
8. To provide a simple referencing system between corresponding grid squares representing C_{in} and C_{out}.
9. To support the efficient definition of local geometries (important in the context of springback prediction).

The grid representation generation process commences with "before" (C_{in}) and "after" (C_{out}) point clouds referenced in terms of a Euclidean coordinate system. C_{in} is the point cloud for the desired shape (T), while C_{out} is the point cloud for the actual shape (T') produced as a result of (say) the application of some sheet metal forming process such as AISF. Two grid representations are initially generated, G_{in} for the C_{in} cloud and G_{out} for the C_{out} cloud. These are then used to produce a final grid representation G for input to the RASP framework.

The size of the grids is defined in terms of a value (d) which represents the size of a grid square as illustrated in Figure 3.2. Each grid square is represented by a central representative point, P, described in terms of a pair of (x, y) coordinates. The z value associated with each grid square is obtained by averaging the z coordinates for all the 3D points located in that grid square. Thus each grid represents a mesh describing a 3D surface. The $x - y$ coordinates for a specific grid square in G_{in} will be the same as the corresponding grid square in G_{out}; however, the z values may differ. Consequently, each grid square is referenced by a pair of (x, y) coordinates and has a z value associated with it. Note that, prior to grid generation, both coordinate clouds, C_{in} and C_{out}, are registered to the same reference origin and direction.

The size, in terms of grid squares, of the overall grid structure is equivalent to $R \times C$ where R is the number of grid rows, and C is the number of grid columns; R and C

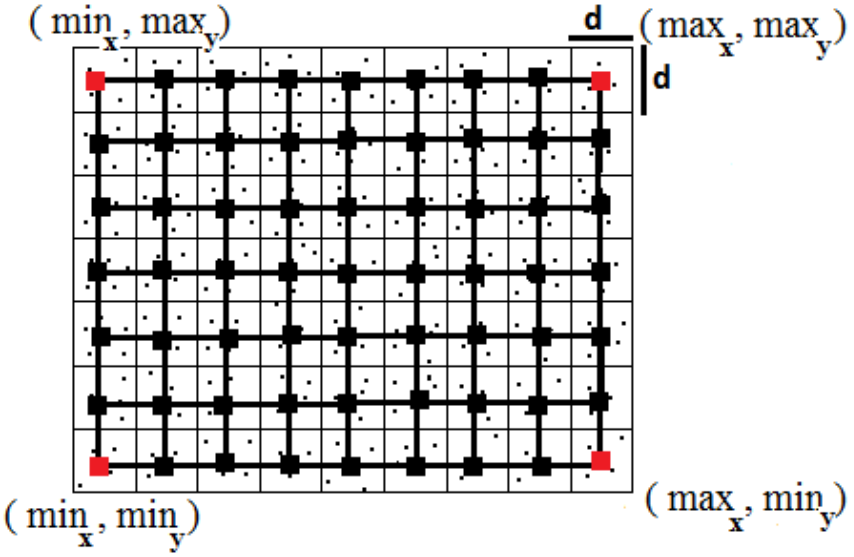


FIGURE 3.2: Typical grid structure for a point cloud (red grid centres indicate the corner grid squares).

are calculated according to the size of the input clouds[1] as described in equations 3.1 and 3.2 respectively where d is the size of the desired grid square in mm. Appropriate adjustments are made where d does not fit exactly into $max_y - min_y$ or $max_x - min_x$.

$$R = \lceil \frac{max_y - min_y}{d} \rceil \tag{3.1}$$

$$C = \lceil \frac{max_x - min_x}{d} \rceil \tag{3.2}$$

with respect to the above min_x, max_x, min_y and max_y are as indicated in Figure 3.2.

To support the generation of classifiers each grid square in the final grid G has to have an error (*"springback"*) value associated with it. How this is calculated is considered in the next section.

3.4 Springback Calculation Mechanism

The mechanism for the calculation of the *"Springback"* (error) between a pair of correlated grid squares is described in this section. Broadly, the separation between two correlated grid squares is the distance along the normal originating from the centre point P of a grid square in G_{in} to where the normal cuts the surface represented by G_{out}. The error calculation mechanism comprises three steps: (i) normal calculation, (ii) G_{out} intersection point calculation and (iii) the final error calculation. Each is discussed in more detail in each of the following subsections.

[1]The point clouds may have different sizes, therefore the shared area (intersection) for both clouds is considered.

3.4.1 Normal Calculation

In order to find the degree of the deviation (distance and direction) between each grid centre point P_i located on the 3D surface described by G_{in} and its corresponding point located on the surface described by G_{out}; the normal for each G_{in} centre point is first calculated. If the before and after surfaces are horizontally parallel, then the G_{in} and the G_{out} centre points will be directly positioned over one another, in which case the error calculation is straightforward (the z difference), however, usually this is not the case. Instead *vector* arithmetic is used. A vector $\vec{v}$ has a magnitude and a direction and is usually represented using the format $\langle x, y, z\rangle$ or $xi + yj + zk$ where i, j and k are the unit vectors in the positive directions of the x, y and z axes respectively and generally referred to as the *standard basis* [11, 59, 131, 208]. A vector from a point $P_0(x_0, y_0, z_0)$ to a point $P_1(x_1, y_1, z_1)$ is defined as $\vec{v} = \langle x_1 - x_0, y_1 - y_0, z_1 - z_0\rangle$. Given two vectors $\vec{v} = \langle v_1, v_2, v_3\rangle$ and $\vec{u} = \langle u_1, u_2, u_3\rangle$ at right angles to each other, the normal $\vec{n}$ is defined by the cross product between $\vec{v}$ and $\vec{u}$ ($\vec{n} = \vec{v} \times \vec{u}$, the result of the cross product of two vectors is a vector that is perpendicular to both vectors $\vec{v}$ and $\vec{u}$). The direction of the normal can be ascertained using the "right hand rule". The cross product is calculated using Equation 3.3 [11, 131, 206] :

$$\vec{n} = \vec{v} \times \vec{u} = \langle v_2u_3 - v_3u_2, v_3u_1 - v_1u_3, v_1u_2 - v_2u_1\rangle. \tag{3.3}$$

The calculation of the normal is thus founded on the concept of the vectors' cross product [59, 183, 208]. With respect to the work described in this thesis there are a number of $\vec{v}$ and $\vec{u}$ vector configurations that can be used to calculate the normal of a grid point. For internal points of the grid, there are four $\vec{v}$ and $\vec{u}$ combinations that can be used. There are three at the edges and two at the corners as illustrated in Figure 3.3. Thus, in each case the separation between the corresponding grid squares in the before and after clouds can be calculated as follows:

- Internal grid point. The four normals for the four neighbouring points (N, E, S and W) are calculated, then the normal for the grid square is calculated by averaging these four normals.
- Edge grid point. The normal is calculated by averaging the two normals that can be obtained.
- Corner grid point. Straight forward calculation of a single normal.

Note that a clockwise direction is used to identify the order of selecting the vectors that are used in the cross product to calculate normals (as illustrated in Figure 3.3) so that the surface normals will all be pointing in the same direction.

3.4.2 Intersection Point Calculation

After the normals have been calculated for all grid centre points in G_{in}; the vector describing each normal is translated into a polynomial equation describing the line (L)

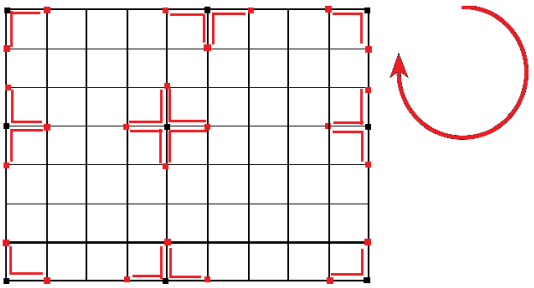


FIGURE 3.3: $\vec{v}$ and $\vec{u}$ vector configurations, indicated in red, that maybe used for normal calculation. Note that a clockwise direction is used so that all normals point in the same direction.

passing through the relevant centre point in G_{in} and intersecting G_{out}. The desired straight line equation is defined using a set of parametric equations.

Fact 3.1. The line in 3-D space that passes through a point $P_0(x_0, y_0, z_0)$, and is parallel to the non zero vector $\vec{v} = \langle a, b, c\rangle = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$, has parametric equations $x = x_0 + at$, $y = y_0 + bt$ and $z = z_0 + ct$.

Thus for each point in a grid P_i $(x_i,\ y_i,\ z_i)$ with its associated normal (calculated as described above) $\vec{n} = \langle a, b, c\rangle$; and, based on Fact 3.1, the parametric equations (Equations 3.4) can be used to determine the straight line equation passing through P_i.

$$x = x_i + at, \qquad y = y_i + bt, \qquad z = z_i + ct. \tag{3.4}$$

With reference to Figure 3.4, the steps required to calculate the t value and the associated intersection point P_{int} are presented below where the following identities are used.

(a) $P_i(x_i, y_i, z_i)$: is a point located on G_{in}.

(b) $P_j(x_j, y_j, z_j)$: is the corresponding point located on G_{out} and determined in the same manner as for P_i.

(c) $P_{int}(x_{int}, y_{int}, z_{int})$: is the intersection point on G_{out} where the normal from P_i cuts G_{out}.

(d) $\vec{n}\langle a, b, c\rangle$: is the normal for the point P_i.

(e) $\vec{N}\langle A, B, C\rangle$: is the normal for the point P_j.

(f) $\vec{L}$: is the line passing through P_i and P_{int}.

The process is thus as follows:

1. Identify the *line* $(\vec{L})$ passing through P_i and P_{int}. The line $(\vec{L})$ is parallel to the normal $(\vec{n})$ to the point P_i.

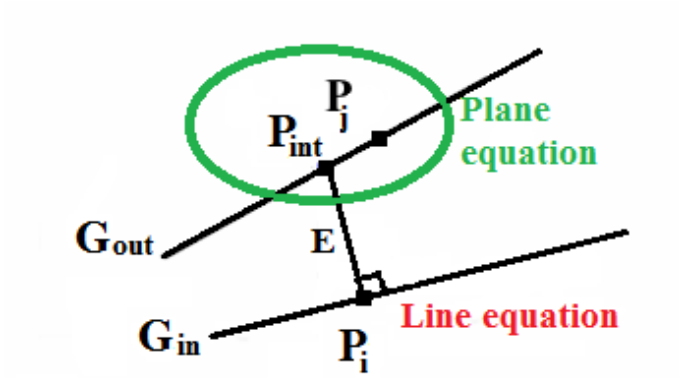


FIGURE 3.4: Error (springback) calculation (E) between G_{in} and G_{out} defined as the distance between the point P_i on G_{in} to where the normal of P_i cuts G_{out} at P_{int}.

$$
\begin{aligned}
&P_{int} - P_i = t \cdot \langle a, b, c \rangle \\
&(x_{int}, y_{int}, z_{int}) - (x_i, y_i, z_i) = t \cdot \langle a, b, c \rangle \\
&x_{int} - x_i = a \cdot t \\
&y_{int} - y_i = b \cdot t \\
&z_{int} - z_i = c \cdot t
\end{aligned}
$$

$$
x_{int} = x_i + a \cdot t \qquad y_{int} = y_i + b \cdot t \qquad z_{int} = z_i + c \cdot t \tag{3.5}
$$

2. Identify the *plane* equation at the point P_j on G_{out}[1]

$$
\begin{aligned}
&\vec{N} \cdot \vec{P_j P} = 0 \\
&A \cdot (x - x_j) + B \cdot (y - y_j) + C \cdot (z - z_j) = 0 \\
&A \cdot x - A \cdot x_j + B \cdot y - B \cdot y_j + C \cdot z - C \cdot z_j = 0 \\
&A \cdot x + B \cdot y + C \cdot z = A \cdot x_j + B \cdot y_j + C \cdot z_j \\
&A \cdot x + B \cdot y + C \cdot z = D
\end{aligned} \tag{3.6}
$$

Note that the intersection point P_{int} should satisfy both the parametric equations described in 3.5 and the plane equation given in (3.6). Thus:

[1] Any point P on G_{out} that satisfy equation 3.6 means it is located on this plane. In our case, P is the intersection point P_{int} on which it satisfies the plane equation of P_j.

$$
\begin{aligned}
&A \cdot (x_i + a \cdot t) + B \cdot (y_i + b \cdot t) + C \cdot (z_i + c \cdot t) = D \\
&A \cdot x_i + A \cdot a \cdot t + B \cdot y_i + B \cdot b \cdot t + C \cdot z_i + C \cdot c \cdot t = D \\
&A \cdot a \cdot t + B \cdot b \cdot t + C \cdot c \cdot t = D - A \cdot x_i - B \cdot y_i - C \cdot z_i \\
&t = \frac{D - A \cdot x_i - B \cdot y_i - C \cdot z_i}{A \cdot a + B \cdot b + C \cdot c}
\end{aligned} \tag{3.7}
$$

Now, by substituting the parametric equations in the plane equation we can find the t parameter for the parametric equation and consequently the coordinate for the intersection point P_{int} can be obtained using (3.5).

3.4.3 Error (Springback) Calculation

After the coordinates for the intersection point have been calculated, the error (springback) to be associated with each G_{in} grid square may be obtained. This is the distance between the centre grid point P_i located on G_{in} and the intersection point P_{int} located on G_{out}. The distance between two points in 3-D space is calculated using equation 3.8.

$$
E = \frac{|a(x_{int} - x_i) + b(y_{int} - y_i) + c(z_{int} - z_i)|}{\sqrt{(a^2 + b^2 + c^2)}} \tag{3.8}
$$

On completion of the error calculation, a vector $\overrightarrow{P_i P_{int}}$ is obtained with a magnitude equal to error (E) and a direction. The error E is assigned a positive sign if the direction for $\overrightarrow{P_i P_{int}}$ and the normal ($\vec{n}$) is the same, and a negative sign otherwise. To determine whether the directions for vectors $\overrightarrow{P_i P_{int}}$ and $\vec{n}$ are the same or not the angle θ between them is calculated using the dot product rule presented in Definition 3.1 above. Clearly, θ has only two options (with respect to our case). The first option is when $\theta = 0°$ indicating that both vectors run parallel in the same direction. The second option is if $\theta = 180°$ which means that both vectors run parallel but in opposite directions (Figure 3.5).

Definition 3.1. Given any two vectors a and b in R^n, the "Dot Product" can be defined as: $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos \angle \mathbf{a}, \mathbf{b}$

where $\|\mathbf{a}\|$ denotes the length of $\mathbf{a}$, and $\angle \mathbf{a}, \mathbf{b}$ is the angle between $\mathbf{a}$ and $\mathbf{b}$, taken to be between 0 and π.

Fact 3.2. Let $\mathbf{u}$ be a vector in the real vector space R^n. Then $\mathbf{u} \cdot \mathbf{u} = \|\mathbf{u}\|^2$, where $\|\mathbf{u}\|$ is the length of $\mathbf{u}$.

Example:
The following example illustrates the Springback Calculation process. Let point P_i and its normal $\vec{n_i}$ located on C_{in}, and point P_j and its normal $\vec{n_j}$ located on C_{out} have the following coordinates:
To define a line $\vec{L}$ passing through P_i and P_{int} and parallel to $\vec{n_i}$:

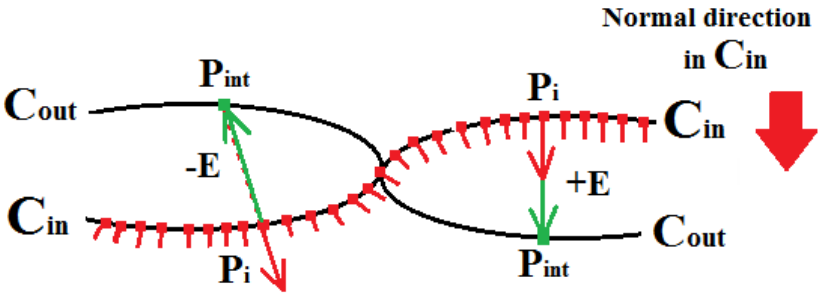


FIGURE 3.5: The Error direction illustrated by two examples. The left hand example shows that both the normal $\vec{n}$ and $\overrightarrow{P_iP_{int}}$ have opposite direction as the angle between them is ($\theta = 180°$). Therefore, the error E, in this case, is assigned a negative ($-$) sign. However, the angle between the normal $\vec{n}$ and $\overrightarrow{P_iP_{int}}$ on the right hand example is $\theta = 0°$ which means that both vectors run parallel in the same direction and as a result the error is assigned a + sign.

$$\begin{aligned} P_i &= (77.50, 38.04, 0.05) \\ \vec{n_i} &= \langle -0.90, 0.00, 18.75 \rangle \text{ The normal of } P_i \\ P_j &= (77.50, 38.04, -0.65) \\ \vec{n_j} &= \langle -1.82, 0.23, 18.75 \rangle \text{ The normal of } P_j \end{aligned}$$

$$P_{int} - P_i = t \cdot \vec{n}$$
$$x_{int} - 77.50 = -0.90 \cdot t$$
$$y_{int} - 38.04 = 0.00 \cdot t$$
$$z_{int} - 0.05 = 18.75 \cdot t$$

$$x_{int} = 77.50 - 0.90 \cdot t \tag{3.9}$$
$$y_{int} = 38.04 + 0.00 \cdot t \tag{3.10}$$
$$z_{int} = 0.05 + 18.75 \cdot t \tag{3.11}$$

The plane equation at P_j is then identified as follows:

$$\vec{n_j} \cdot \overrightarrow{P_jP} = 0$$
$$A \cdot (x - x_j) + B \cdot (y - y_j) + C \cdot (z - z_j) = 0$$
$$-1.82 \cdot (x - 77.50) + 0.23 \cdot (y - 38.04) + 18.75 \cdot (z + 0.65) = 0$$
$$(-1.82 \cdot x) + (0.23 \cdot y) + (18.75 \cdot z) = -144.35 \tag{3.12}$$

Now, Equations 3.9, 3.10 and 3.11 are substituted in Equation 3.12 to obtain the t value:

$$t = \frac{-144.35 - (-140.89) - (8.67) - (0.86)}{(-1.82 \cdot -0.90) + (0.23 \cdot 0.00) + (18.75 \cdot 18.75)}$$

$$t = \frac{-144.35 - (-140.89) - (8.67) - (0.86)}{(1.64) + (0.00) + (351.56)}$$

$$t = -12.99/353.21$$

$$t = -0.04$$

Next, the coordinates of the intersection point P_{int} are obtained by re-evaluating equations 3.9, 3.10 and 3.11:

$$x_{int} = 77.50 - (0.90 \times -0.04) = 77.53$$
$$y_{int} = 38.04 + (0.00 \times -0.04) = 38.04$$
$$z_{int} = 0.05 + (18.75 \times -0.04) = -0.64$$

Finally, the error is calculated. This is the distance between P_i on C_{in} and the intersection point P_{int} on C_{out}, therefore the error (Springback) associated with the point $P_i = (77.50, 38.04, 0.05)$ is:

$$E = \frac{|a(x_{int} - x_i) + b(y_{int} - y_i) + c(z_{int} - z_i)|}{\sqrt{(a^2 + b^2 + c^2)}}$$
$$E = \frac{|(-0.90 \times 0.03) + (0.00 \times 0.00) + (18.75 \times -0.69)|}{\sqrt{(-0.90)^2 + (0.00)^2 + (18.75)^2}}$$
$$E = +0.69$$

E is assigned a positive sign + because the angle between vector $\overrightarrow{P_iP_{int}}$ and the normal $\vec{n_i}$ is $\theta \approx 0$ as shown below:

$$|\vec{n_i}|^2 = (-0.90)^2 + (0.00)^2 + (18.75)^2$$
$$= 358.38$$
$$\overrightarrow{P_iP_{int}}\langle P_{intx} - P_{ix}, P_{inty} - P_{iy}, P_{intx} - P_{ix}\rangle$$
$$\overrightarrow{P_iP_{int}}\langle 0.03, 0.00, -0.69\rangle$$
$$|\overrightarrow{P_iP_{int}}|^2 = (0.03)^2 + (0.00)^2 + (-0.69)^2$$
$$= 0.48$$
$$\vec{n} \cdot \overrightarrow{P_iP_{int}} = n_{ix} \cdot p_x + n_{iy} \cdot p_y + n_{iz} \cdot p_z$$

$$
\begin{aligned}
&= (-0.03) + 0.00 + (-12.96) \\
&= -12.96 \\
\cos\theta &= (\vec{n_i} \cdot \overrightarrow{P_i P_{int}}) / \sqrt{|\vec{n_i}|^2 + |\overrightarrow{P_i P_{int}}|^2} \\
&= -12.96 / \sqrt{358.38 + 0.48} \\
&= 0.69 \\
\theta &= \arccos(-0.69) \\
\theta &= 2.33^\circ \\
\theta &\approx 0^\circ
\end{aligned}
$$

This means that both $\overrightarrow{P_i P_{int}}$ and $\vec{n_i}$ run parallel in the same direction.

3.5 Discretising process

After the error values (the set E) have been calculated and assigned to the P_i's in the G_{in}, a *discretising* (*binning*) process is applied where each springback error value is given a label value. The inherent nature of the classification techniques used later in this thesis require discrete attribute values, the suggestion being that more effective classifiers are produced [128].

One of the main objectives of discretisation is to enhance computational effectiveness (the number of discretised values, labels, are fewer than the number of all possible continuous values) [35, 54]. Therefore, discretisation was adopted to transform the continues real values into nominal values by dividing the range of values into sub-ranges where each sub-range is replaced by an integer value (label).

There are two types of discretisation [35]:

1. *Equal width* discretisation: where the range of values are divided into k equal length intervals (thus the number of instances in each bin is different) such that bin "size" is $((max - min)/k)$. Typically, the first and the last bins are extended to include values outside the data range.

2. *Equal frequency* discretisation: where the range of values are divided into k bins such that each bin holds an equal number of instances (n/k) where n is the total number of instances. Different variations of equal frequency binning have been proposed (more details can be founded in [54]).

Equal width and equal frequency discretisation are both considered to be simple unsupervised methods where no attempt is made to take class labels into consideration. The equal frequency and equal width approaches considered are also global discretisation approaches as they both operate using all available values. The main limitation of the

equal width discretisation is that it may result in many instances belonging to one range while few instances belong to others (a skewed distribution). Thus, with respect to the work reported in this thesis, equal frequency binning was adopted and applied because of the unequal distribution of the springback values over 3D surfaces with respect to the AISF process. However, it should be noted that with respect to some of the published work, produced prior to the work described in this thesis, equal width discretisation was used by the author. An overview of different discretisation algorithms can be found in [138].

Algorithm 3.1 presents an overview of the discretisation process adopted with respect to the work described in this thesis. The input is a data set M comprised of m records characterised by a set of attributes (features) $\{t_0, t_1, \cdots, t_n\}$ and a class attribute c such that each attribute has a set of values associated with it. The algorithm commences by sorting the records for each attribute t_i according to value (ascending order) and associating a *rank* with each (a sequential numbering from 1 to m). Then, a Discretisation Table is generated that contains information about label IDs and the rank of each attribute using Algorithm 3.2. Note that label ID's are numbered sequentially. Given a data set M comprised of 90 records, an attribute set $A = \{att_0, att_1, att_2, att_3\}$ where each attribute has a set of ranked values between 1 and 90 and L is a set of labels of size 3. The Discretisation Table will be as shown in Table 3.1.

TABLE 3.1: *Discretisation Table* for a given example.

	att_0	att_1	att_2	att_3
labels ID (max rank)	**1** (30) **2** (60) **3** (90)	**4** (30) **5** (60) **6** (90)	**7** (30) **8** (60) **9** (90)	**10** (30) **11** (60) **12** (90)

Algorithm 3.3 is then applied to discretised the attributes based on the label information gathered in the Discretisation Table. Thus, for a given instance, if the rank values of att_3 is 67, then the label is 12. Finally, the class attribute is discretised using Algorithm 3.4, where M is sorted in ascending order where the first $\lceil m/c \rceil$ instances have the same label, then the next $\lceil m/c \rceil$ and so on. By the end, all the instances will have been discretised with respect to the prescribed set of attributes and class labels.

3.6 Evaluation Data Sets

To evaluate the different 3-D surface representation methods proposed, and the associated classification techniques, eight *" real world "* data sets were used:

- Gonzalo Steel Version 1 (GSV1).
- Gonzalo Steel Version 2 (GSV2).
- Gonzalo Titanium Version 1 (GTV1).

Algorithm 3.1: Equal Frequency Discretisation

Input: Dataset M with m records, set of attributes $A = \{att_0, att_1, \cdots, att_n\}$, number of attribute labels $|L|$, number of class label $|c|$)
Output: *Discretised* dataset M

```
rankList ← empty 2D array of size m × n;
for j ← 0 to n do
    for i ← 0 to m do
        rankList [i][j] ← value of t_j in record i;
    end
    rankList ← sorted rankList with respect to t_j values ;
end
T ← GenerateDiscretisationTable(rankList, m, n, |L|) (Algorithm 3.2);
M ← DiscretiseAttributes (T, M, m, n, |L|) (Algorithm 3.3);
M ← DiscretiseClassAttribute (M, m, n, |c|)) (Algorithm 3.4);
```

Algorithm 3.2: GenerateDiscretisationTable

Input: rankList, number of records m, number of attributes n, number of attribute labels $|L|$
Output: Discretisation Table T

```
T ← empty 3D array measuring n × |L| × 2;
k = 1;
for i ← 0 to n do
    for j ← 0 to |L| − 1 do
        index ← ⌈ m × (j + 1) / |L| ⌉;
        T[i][j][0] ← index;
        T[i][j][1] ← k;
        k + +;
    end
end
return T
```

- Gonzalo Titanium Version 2 (GTV2).
- Modified Steel Version 1 (MSV1).
- Modified Steel Version 2 (MSV2).
- Modified Titanium Version 1 (MTV1).
- Modified Titanium Version 2 (MTV2).

These were manufactured by IBF[1] who provided support for the work described in this thesis. The first four describe a flat topped pyramid shape referred to as the "Gonzalo" pyramid[2], the last four a "modified" version of this shape. The Gonzalo pyramid is

[1]The Institut für Bildsame Formgebung (Institute of Metal Forming) at the Rheinisch-Westfaelische Technische Hochschule Aachen, Germany

[2]The name "Gonzalo" is derived from the name of the person at IBF who designed and manufactured the shape.

Algorithm 3.3: DiscretiseAttributes

Input: Discretisation Table T, Dataset M, number of records m, number of attributes n, number of attributes labels $|L|$
Output: Dataset M with discretised attribute values

```
1  label ← |L| − 1;                          // Default label is the max label ID
2  for i ← 0 to m do
3      for j ← 0 to n do
4          for k ← 0 to |L| do
5              if M[i][j].rank < T[j][k][0] then
6                  label ← T[j][k][1];
7                  break;
8              end
9          end
10         M[i][j] ← label;
11     end
12 end
13 return M
```

asymmetric in shape, a bulge exists in one of its side sections. The Modified pyramid is an enhanced shape with rounded corners. The differences between the Gonzalo and Modified shapes can be observed from Figures 3.6 and 3.7.

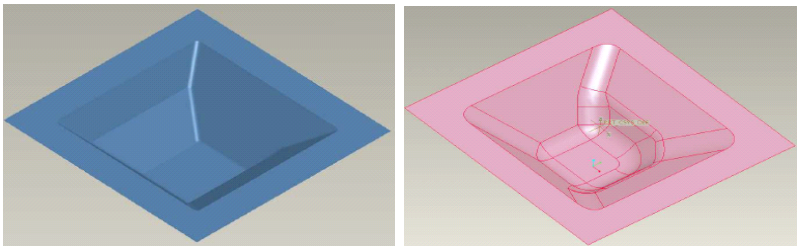


FIGURE 3.6: Gonzalo (*left*) and Modified (*right*) Pyramids.

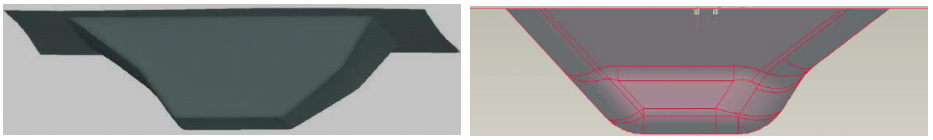


FIGURE 3.7: Side sections of Gonzalo (*left*) and Modified (*right*) Pyramids.

Note also that four of the data sets were made of steel and four of titanium. Each had a C_{in} before and C_{out} after cloud associated with it (thus sixteen clouds in total). In each case the C_{in} point cloud, representing the desired shape T for each pyramid shape, was generated using a CAD system and thus this cloud is sometimes referred to as the CAD cloud (representing the CAD shape). The C_{out} output point cloud, representing the obtained shape T', was obtained using a GOM (Gesellschaft f'ur Optische Messtechnik) optical measuring tool applied on completion of an AISF process, and thus this output cloud is sometimes referred to as the GOM cloud (representing the GOM shape). Flat topped pyramid shapes were used because this is a commonly used shape with respect to

Algorithm 3.4: Discretise Class Attribute

Input: M, m, n, $|c|$
Output: Dataset M with discretised class attribute values

Sort M based on class attribute values in ascending order;
rankList $\leftarrow$ array of length m;
for $i \leftarrow 0$ **to** m **do**
    rankList [i] $\leftarrow$ *rank* springback error value;
end
$T \leftarrow$ empty 2D array measuring $|c| \times 2$;
$k = 1$; `// Start class labelling from 1.`
for $j \leftarrow 0$ **to** $|c| - 1$ **do**
    index $\leftarrow m \times \dfrac{(j+1)}{|c|}$;
    $T[j][0] \leftarrow rankList[index]$;
    $T[j][1] \leftarrow k$;
    $k++$;
end
for $i \leftarrow 0$ **to** m **do**
    label $\leftarrow |c| - 1$;
    for $j \leftarrow 0$ **to** $|c| - 1$ **do**
        if $M[i][n+1].rank < T[j][0]$ **then**
            label $\leftarrow T[j][1]$;
            break;
        end
    end
    $M[i][n+1] \leftarrow$ label;
end
return M

sheet metal forming related research. Tables 3.2 and 3.3 give some statistics regarding the evaluation data sets in the context of their C_{in} and C_{out} clouds.

TABLE 3.2: Statistics concerning the width (W) (mm), length (L) (mm), height (H) (mm), area (A) (mm^2), number of points (N) and density with respect to the C_{in} point clouds for each of the evaluation data sets.

	GSV1	GSV2	GTV1	GTV2	MSV1	MSV2	MTV1	MTV2
Width (W)	195	195	195	195	190	190	190	190
Length (L)	195	195	195	195	190	190	190	190
Height (H)	43	43	43	43	42	42	42	42
Area ($A = W \times L$)	38025	38025	38025	38025	36100	36100	36100	36100
Num. points (N)	250847	250847	250847	250847	565817	565817	565817	565817
Density (N/A)	7	7	7	7	16	16	16	16

The pairs of C_{in} and C_{out} clouds were processed as described above, to provide the desired grid representations required by phase two of the RASP framework. For experimental purposes a range of d values were used $\{2.5, 5, 10, 15, 20\}$ mm. Thus forty (8×5) data sets were generated in total, The number of records associated with each

TABLE 3.3: Statistics concerning the width (W) (mm), length (L) (mm), height (H) (mm), area (A) (mm^2), number of points (N) and density with respect to the C_{out} point clouds for each of the evaluation data sets.

	GSV1	GSV2	GTV1	GTV2	MSV1	MSV2	MTV1	MTV2
Width (W)	194	194	199	195	196	196	195	195
Length (L)	194	194	189	194	195	195	195	194
Height (H)	45	44	46	46	45	44	47	46
Area ($A = W \times L$)	37636	37636	37611	37830	38220	38220	38025	37830
Num. points (N)	421214	233480	430900	185526	257436	269031	394895	401186
Density (N/A)	11	6	11	5	7	7	10	11

is presented in Table 3.4. From the table it can be seen, as would be expected, that as d increases the number of records decreases. Note also that the shapes were not all of exactly the same size, hence at lower values of d the number of records is no longer consistent across the eight surfaces. With respect to the conducted evaluations, reported in detail later in this thesis, it should be noted that given a pair of shapes (surfaces) the generated classifiers were training and testing on the same shape, or by training on one shape and testing on another. The $GSV1$ and $MSV1$ shapes are depicted in Figures 3.8 and 3.9 respectively using $d = 1$ ($d = 1$ was used so that the differences between the generated shapes can be easily noticed and detected).

TABLE 3.4: Number of records generated for the Gonzalo and Modified pyramids using different values of d

d	GSV1	GSV2	GTV1	GTV2	MSV1	MSV2	MTV1	MTV2
2.5	6086	6086	5928	6086	5853	5853	5853	5853
5	1523	1523	1483	1523	1483	1483	1483	1483
10	402	402	381	402	381	381	381	381
15	171	171	171	171	171	171	171	171
20	102	102	102	102	102	102	102	102

3.7 Summary

This chapter presented an overview of the necessary preprocessing required to translate sheet metal forming input point cloud data into the desired grid representation (with associated springback error values). This has been incorporated into the RASP framework which comprises three phases. During phase 1 the point clouds are translated in to a grid representation that provides a simple, manageable and uniformed structured for both the C_{in} and C_{out} point clouds. Error values were then associated with the G_{in} grid centres to produce the grid representation G required for Phase 2 of the RASP framework. The error calculation mechanism was founded on the concepts of line and plane equations, and 3-D vector representation and manipulation. The pre-processing stages would not be completed without the labelling process, therefore a full description for the

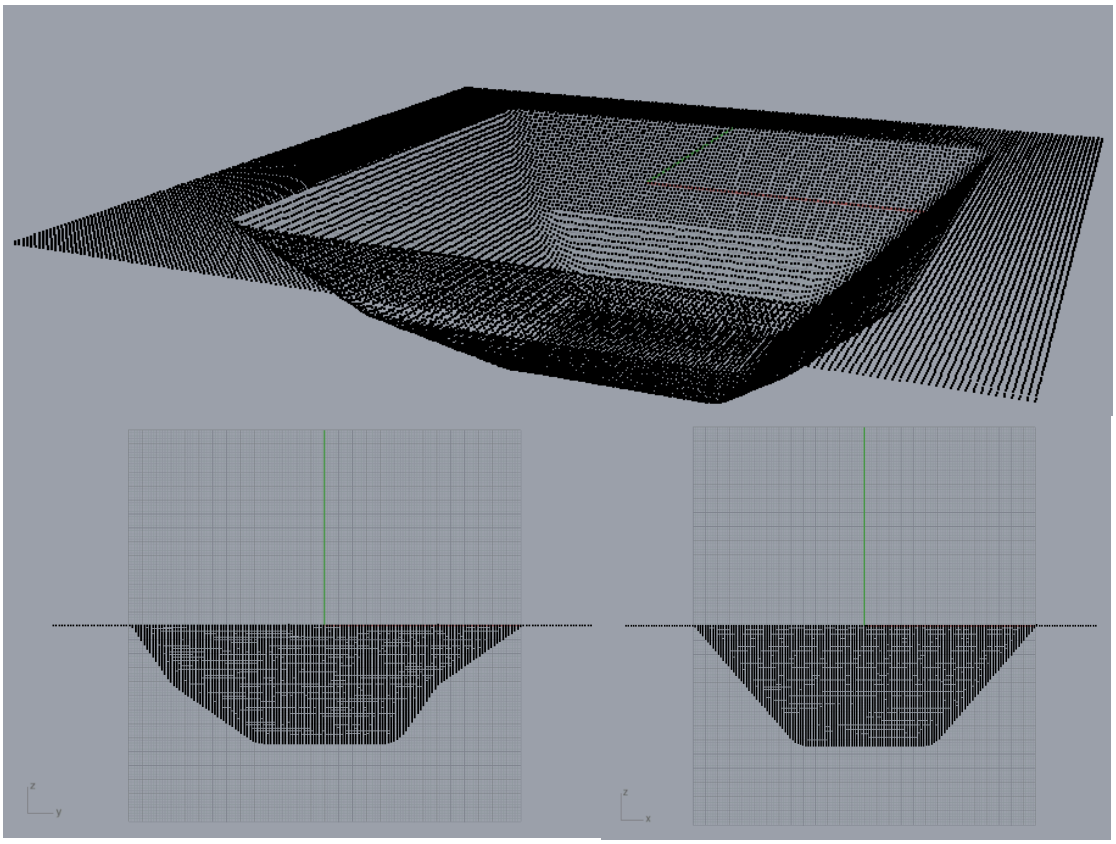


FIGURE 3.8: Different views for the GSV1 G_{in} point cloud using grid size $d = 1mm$.

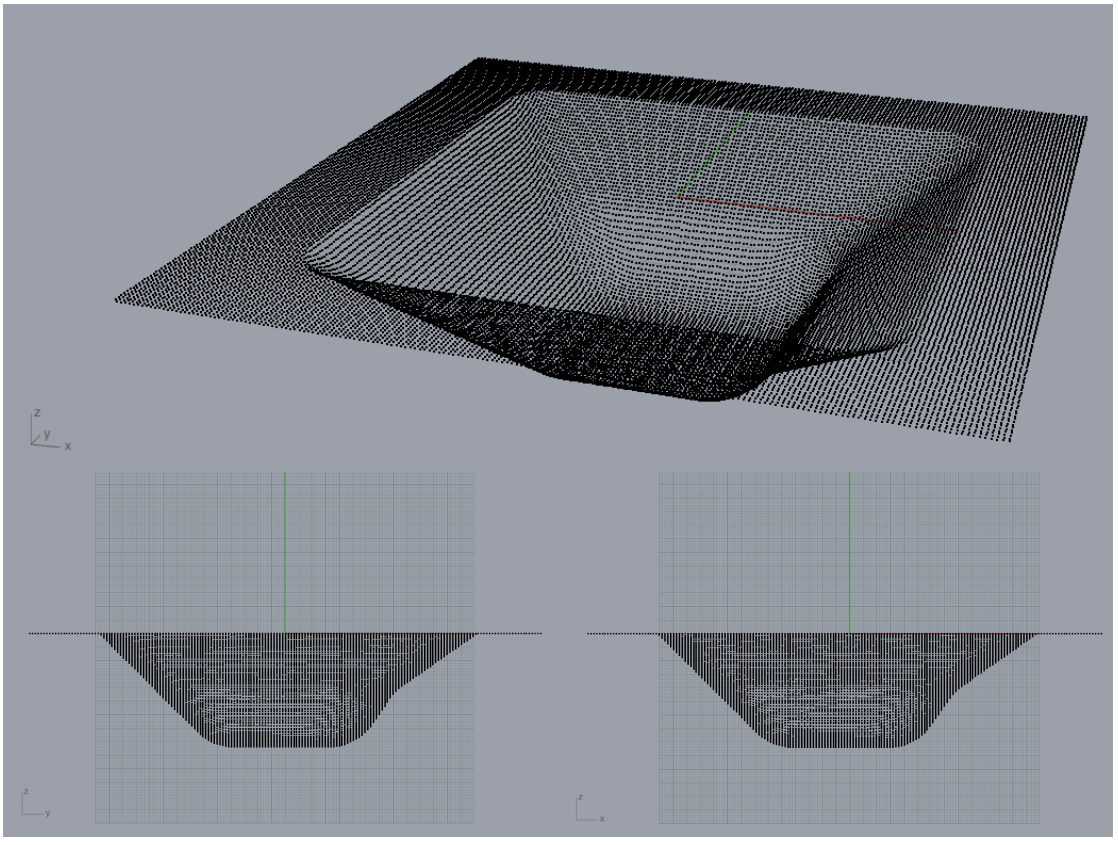


FIGURE 3.9: Different views for the MSV1 G_{in} point cloud using grid size $d = 1mm$.

process was also presented. Finally a description of the data sets, used for evaluation purposes later in this thesis, was given. In the next chapter, the first proposed surface representation techniques is considered.

Chapter 4

Local Geometry Matrix Representation (LGM)

4.1 Introduction

The first 3-D surface representation considered in this thesis, and that presented in this chapter, is the Local Geometry Matrix (LGM) method. The input to the LGM representation is a "coordinate grid" of the form described previously in Chapter 3. The LGM method is directed at the capture of local 3-D surface geometries in terms of the "local" neighbourhoods associated with each coordinate grid square. As will become apparent later in this chapter, the idea behind LGMs is founded on the concept of Local Binary Patterns (LBPs) as used, for example, with respect to image texture analysis [88, 159]. A neighbourhood to a grid square comprises the grid squares surrounding it up to a certain distance away (some authors use the term "radius", see for example [207]). In the context of this thesis neighbourhoods are defined in terms of levels. A level one neighbourhood comprises the grid squares immediately adjacent (in all eight direction) to the current grid square. A level two neighbourhood comprises the grid squares one "step" away (in all eight direction) from the current grid square, and so on. The concept of levels of neighbourhoods will become clearer later in this chapter. For the purpose of the evaluation of the proposed LGM method, level one and two neighbourhoods were considered independently and in combination. The metrics used for the evaluation were the accuracy and Area Under the ROC Curve (AUC). As will be reported later in this chapter, good classification AUC results were obtained using the LGM method, not only when the classifiers are trained and tested on identical shapes, but also when the classifiers are trained on one shape and tested on another. The rest of this chapter is organised as follows. The proposed LGM method is described comprehensively in Section 4.2 where a number of different models of the LGM technique are also presented. Section 4.2 also includes some examples illustrating how the different LGM models are constructed and how they are used to generate feature vectors to be used later with respect to the RASP framework. The data sets introduced in Chapter 3

were used with respect to the evaluation, the results of which are described in Section 4.3. The chapter is concluded with a summary of the main findings in Section 4.4.

4.2 The Local Geometry Matrix (LGM)

As noted above the idea behind the LGM model is to describe the geometry of a given 3D surface shape in terms of the local geometry associated with each grid square in the input. In general terms an LGM is typically a 3×3 matrix whose elements describe some geometric feature with respect to a given grid square represented by the central element of the matrix. The matrix elements surrounding the central element are referred to as its neighbours, the entire matrix thus describes a "neighbourhood". As mentioned in the introduction to this chapter different kinds of neighbourhood may be considered. In this chapter we initially consider what we have referred to as level one and level two neighbourhoods as shown in Figures 4.1 and 4.2. Both are described using 3×3 LGMs with the point of interest p_0 at the centre of the matrix (shown in black and the eight neighbourhoods P_i shown in red). The distinction is that the level 2 neighbourhood covers a larger area and, it was conjectured, might serve to better capture the geometry surrounding a grid square (although the grid size d would clearly also have a role to play in this respect). A composite model was also consider (Figure 4.3) represented by a LGM measuring 5×5. There are two different options with respect to the values that may be stored in an LGM:

1. The first option is to calculate *the positive or negative difference in hight* (δz_i) between the centre grid point P_0 and each of its neighbours P_i.
2. The second option is to calculate *the angle* (θ_i), above or below the horizontal, between P_0 and each of its neighbours P_i.

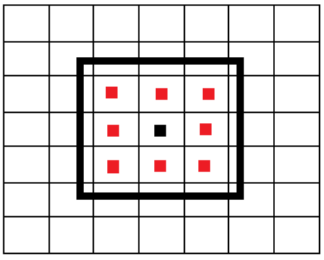


FIGURE 4.1: Level one neighbourhood model. The eight closest surrounding neighbours (P_i coloured in red) for the grid square are considered and represented using a 3×3 LGM (P_0 is the centre point coloured in black).

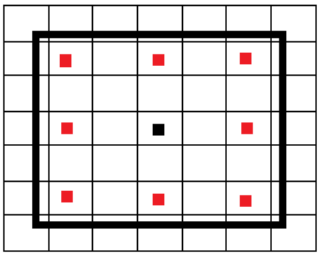


FIGURE 4.2: Level two neighbourhood model. The eight surrounding neighbours (P_i coloured in red) for the grid square that are "one step away" are considered and represented using a 3×3 LGM (P_0 is the centre point coloured in black).

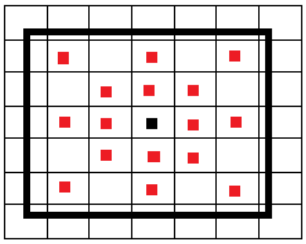


FIGURE 4.3: The composite model founded on a 5×5 LGM to represent the surrounding neighbourhood (P_i coloured in red) of P_0 coloured in black.

Whatever the case at the end of the process we have LGMs describing each grid point in the input coordinate grid. With respect to the variations considered, the LGMs comprise either $(3 \times 3) - 1 = 8$ or $2 \times ((3 \times 3) - 1) = 16$ values in terms of either δz_i or θ_i (-1 to exclude the centre point which always has the value zero) each with an error value associated with it (calculated as described in Chapter 3).

The next step is to convert the LGMs into a feature vector representation that can be used as input to the classifier generation process. A set of qualitative labels L is used to describe the nature of "the slope" for the surrounding neighbours. Therefore, L^8 different *local geometries* can be described with respect to the level one and level two neighbourhoods. For the composite neighbourhood model there will be L^{16} different *local geometries*. Thus, if the label set is $L = \{negative, level, positive\}$ then $3^8 = 6,561$ different combinations can be used to describe the nature of the local geometry surrounding each grid point in the level one and level two neighbourhood models. An example of a resulting feature vector might then be of the form:

$$\langle positive, negative, positive, level, negative, positive, negative, level, E \rangle$$

where E is the associated error value. It should be noted that the order of the geometrical information around P_0 is important which means that the geometries are not rotation invariant. For example, if two feature vectors have the same occurrence for each label ($negative = 3$, $level = 2$, $positive = 3$) but with different order as follows.

$$\langle positive, positive, positive, level, negative, negative, negative, level, E_1 \rangle$$
$$\langle negative, negative, negative, level, positive, positive, positive, level, E_2 \rangle$$

then both are not equivalent so they will be associated with different E values (E_1 and E_2 respectively as shown). The order of reading the geometrical information is fixed starting from the top left and moving in clockwise direction with respect to LGM. For the composite model, $3^{16} = 43,046,721$ different combinations can be used to describe the nature of the 16 neighbours surrounding the grid point. Different sizes of $|L|$ may be used to generate configurations of the desired binary valued feature vectors. To illustrate the operation of the LGM representation method two examples are presented

below (the examples are taken from [125]):

Example 1: Side Location. Considering a "side" location of a flattened square based pyramid of the form indicated in Figure 4.4. Table 4.1 shows the z values for a (3×3) grid representing this location. The associated level 1 LGM will then be of the form shown in Table 4.2 calculated using the δz values between the centre point and its neighbourhoods. The δz values are then recorded in a clockwise direction from the top left, with an associated E (springback) value to form the desired feature vector that will be used later in the RASP framework. If the label set is $L = \{negative, level, positive\}$, then the feature vector would be:

$$\langle positive, positive, positive, level, negative, negative, negative, level, E \rangle$$

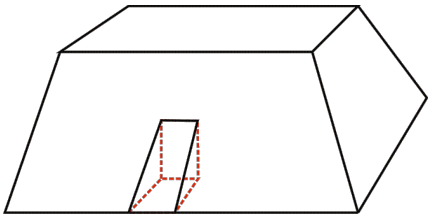


FIGURE 4.4: Square based pyramid with side location highlighted (Example 1).

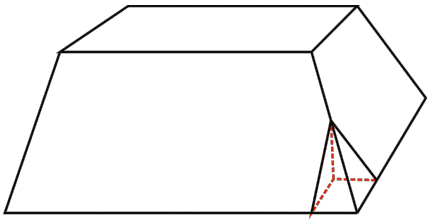


FIGURE 4.5: Square based pyramid with corner location highlighted (Example 2).

40	40	40
20	20	20
0	0	0

TABLE 4.1: Z matrix for Example 1.

20	20	20
0	0	0
-20	-20	-20

TABLE 4.2: LGM for Example 1.

Example 2: Corner Location. Considering a corner location of a flattened square based pyramid of the form indicated in Figure 4.5. Table 4.3 shows the z values for a 3×3 grid representing this location. The associated LGM is presented in Table 4.4. If the label set $L = \{negative, level, positive\}$ is again used, then the resulting feature vector, including an E value, would be:

$$\langle positive, level, negative, negative, negative, negative, negative, level, E \rangle$$

40	20	0
20	20	0
0	0	0

TABLE 4.3: Z matrix for Example 2.

20	0	-20
0	0	-20
-20	-20	-20

TABLE 4.4: LGM for Example 2.

However, in this thesis only classifiers that operate with binary valued data are considered, the LGM information needs to be represented in a binary (zero-one) format. Thus each possible pairing of location p_1 to p_8 (p_1 to p_{16} in the case of the composite neighbourhood model) and value from L is considered to be a binary valued "data attribute" that can exist or not exists. Thus we have:

$$\langle p_1 : negative, p_1 : level, p_1 : positive, \ldots, p_8 : negative, p_8 : level, p_8 : positive \rangle$$

or in the case of the composite neighbourhood model:

$$\langle p_1 : negative, p_1 : level, p_1 : positive, \ldots, p_{16} : negative, p_{16} : level, p_{16} : positive \rangle$$

Thus in the case of the two examples above the LGMs would be represented as follows:

$$\text{Example 1: } \langle 0,0,1,0,0,1,0,0,1,0,1,0,1,0,0,1,0,0,1,0,0,0,1,0 \rangle$$
$$\text{Example 2: } \langle 0,0,1,0,1,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,0,1,0 \rangle$$

In practice it makes sense to only store the 1s, we do this by giving a sequential number to each attribute value pair (for convenience we will simply call this the attribute number). Thus, in the case of the above examples, the resulting vectors will be:

$$\text{Example 1: } \langle 3,6,9,11,13,16,19,23 \rangle$$
$$\text{Example 2: } \langle 3,5,7,10,13,16,19,23 \rangle$$

Some further examples of feature vectors extracted using the LGM model, with respect to the Gonzalo and Modified pyramid data sets, are presented in Tables 4.5, 4.6 and 4.7. Table 4.5 presents a fragment of the Gonzalo pyramid dataset represented using the level one neighbourhood model with $|L| = |L_E| = 3$ ($|L_E|$ is the set of labels for the error value). Table 4.6 presents a fragment of the Modified pyramid dataset represented using the level two neighbourhood model with $|L| = |L_E| = 3$. Table 4.7 presents a fragment of the Modified pyramid dataset represented using the composite neighbourhood model with $|L| = |L_E| = 7$. The tables merit some further discussion. If $|L| = 3$ a point P_i will have either label 1, 2, 3 associated with it, or 4, 5, 6 associated with it, and so on. It should be noted that E is also discretised using another set of labels L_E. Thus using the level one or level two neighbourhood models the range of attribute identifiers associated with each record will be from 1 to $(8 \times |L|) + |L_E|$ inclusive. In the case of the composite model the range of identifiers will be from 1 to $(8 \times |L|) + (8 \times |L|) + |L_E|$ inclusive. For example, with respect to Table 4.5 the maximum value is $(8 \times 3) + 3 = 27$. With respect to Table 4.7 the maximum value is $(8 \times 7) + (8 \times 7) + 7 = 119$.

Once the feature vectors have been generated the next stage, within the RASP framework, is to apply an appropriate classifier generator to the data to produce the desired classifier which can then be used to predict the errors associated with new shapes.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	E
3	6	9	12	15	18	21	24	27
2	5	8	11	14	17	20	23	25
3	6	9	12	15	18	21	24	27
2	5	8	11	14	17	20	23	25
2	5	8	11	14	17	20	23	25
2	5	8	11	14	17	20	23	26
2	5	8	11	14	17	20	23	27
1	4	7	10	13	16	19	22	25

TABLE 4.5: Sample feature vectors for the Gonzalo pyramid data using $|L| = |L_E| = 3$ and the level one neighbourhood model

P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	E
1	4	7	10	13	16	19	22	25
1	4	7	10	13	16	19	22	25
1	4	7	10	13	16	19	22	25
2	5	8	11	14	17	20	23	25
3	5	8	12	15	18	21	24	27
3	6	8	12	15	18	21	24	27
3	5	8	12	15	18	21	24	26
2	5	8	11	14	17	20	23	25

TABLE 4.6: Sample feature vectors for the Modified pyramid data using $|L| = |L_E| = 3$ and the level two neighbourhood model

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	E
1	8	15	22	29	36	43	50	57	64	71	78	85	92	99	106	113
2	9	16	23	30	37	44	51	58	65	72	79	86	93	100	107	114
3	10	17	24	31	38	45	52	59	66	73	80	87	94	101	108	115
4	11	18	25	32	39	46	53	60	67	74	81	88	95	102	109	116
5	12	19	26	33	40	47	54	61	68	75	82	89	96	103	110	117
6	13	20	27	34	41	48	55	62	69	76	83	90	97	104	111	118
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119
3	11	18	25	29	38	44	54	59	66	72	80	87	93	102	110	113

TABLE 4.7: Sample feature vectors for the Modified pyramid data using $|L| = |L_E| = 7$ and the composite neighbourhood model

4.3 Experiments and Evaluation

To evaluate the proposed LGM method, with respect to its three variations (the level one, level two and the composite neighbourhood models), a variety of experiments were conducted using different data sets generated from the test shapes presented in Chapter 3. Recall that the generic shape used for experimentation purposes was a flat-topped pyramid with a square base (the flat topped pyramid shape is a commonly used shape with respect to AISF related research) similar to the shape shown previously in Figures 4.4 and 4.5 taken from [125]. Recall also that the RASP framework comprise three phases: (i) the data pre-processing phase where the initial input grid representation is generated and error calculation performed, (ii) the surface representation phase that results in a set of feature vectors, and (iii) the classifier generation and evaluation phase. In the context of the work described in this chapter the RASP surface representation is the LGM representation.

The main objectives for the experiments were:

1. To identify whether the δz or θ LGM representation was the most effective.
2. To identify the most appropriate value for d so that the LGM models could best capture the geometry of the surface of interest.
3. To identify the best label size $|L|$ for the LGM models.

4. To identify the most appropriate LGM model (level one, level two or the composite model).

5. To identify the best associated classification generation methods.

6. To identify whether a generic classifier, trained on one shape and applied to another, can be produced.

7. To determine the time complexity of the proposed approach.

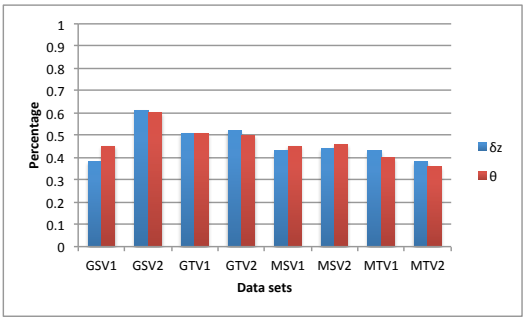
Each experiment is considered in further detail in the following seven subsections (one per objective). All experimental results were recorded in terms of accuracy and Area Under ROC Curve (AUC) [68, 79].

4.3.1 Identifying whether the δz or θ LGM representation is the most effective

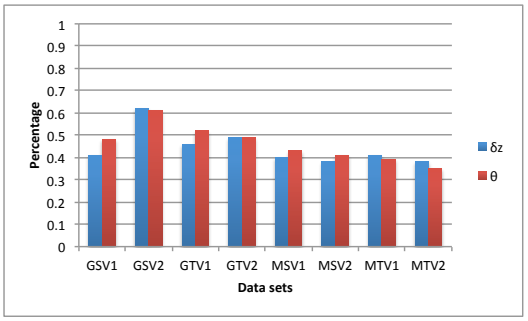
The experiments reported on in this sub-section were concerned with finding the most appropriate option for the values to be stored in the considered LGM models, either δz or θ. The advantage offered with respect to use of the θ value was that it was bounded (between $-90°$ and $+90°$), whereas the potential δz values are unbounded. A range of d values were considered $\{2.5, 5, 10, 15, 20\}$ (mm). All other parameters were kept constant and set to their most appropriate values as indicated by a number of previously conducted experiments which will be summarised in the following sub-sections. Thus: (i) $|L| = 3$, (ii) the C4.5 algorithm for classifier generation and (iii) the composite LGM model, were used. The results are presented in Figures 4.6 and 4.7 in the form of a set of histograms. The histograms plot AUC and accuracy using both the δz and θ options (δz in blue and θ in red). From the figures it can be seen that both δz and θ produced good results and that there is little difference in their operation. However, it is possible to make the argument that δz produced slightly better AUC results than θ as the best AUC result of 0.90 was obtained using δz with respect to the GSV2 data set when $d = 15$ (the corresponding best θ result was 0.88). The best accuracy result of 0.76 was obtained again using δz with respect to the GSV2 data set when d=20. Therefore, for ease of understanding, only results produced using δz are reported with respect to the other experiments considered later in this chapter.

4.3.2 Identification of the best value for d (grid size)

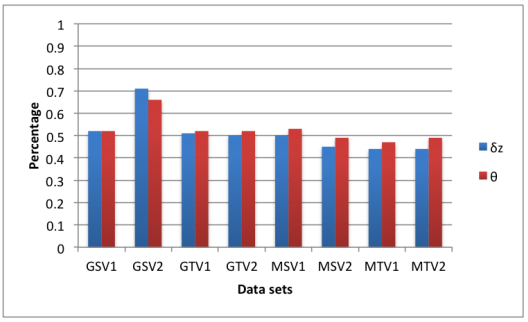
In order to establish the best value for the grid size d a range of d values were considered: $\{2.5, 5, 10, 15, 20\}$ (mm). Other parameters were kept constant: (i) the composite LGM model coupled with δz was used, (ii) $|L| = 3$ and (iii) C4.5. The results are presented in Figure 4.8. From the figure it can be seen that the highest AUC and accuracy results were obtained when the grid sizes were $d = 10$, $d = 15$ and $d = 20$. A summary table for the range of AUC and accuracy results obtained for the different grid sizes is presented



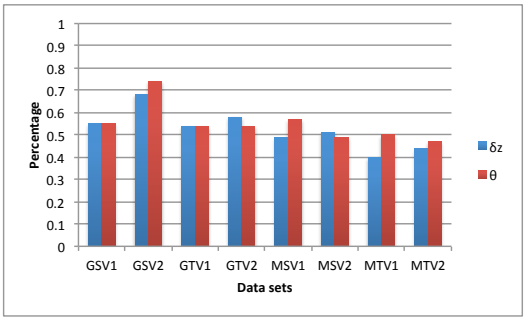
(a) *Accuracy* ($d = 2.5$)



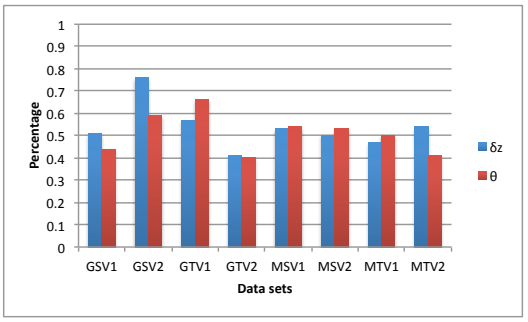
(b) *Accuracy* ($d = 5$)



(c) *Accuracy* ($d = 10$)



(d) *Accuracy* ($d = 15$)



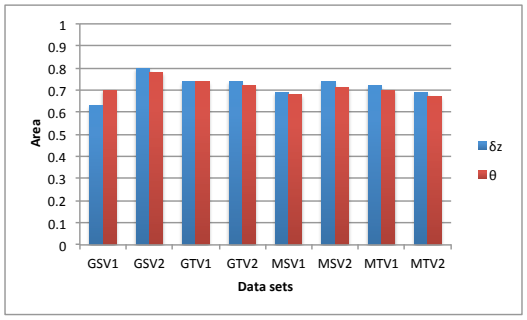
(e) *Accuracy* ($d = 20$)

FIGURE 4.6: Comparison of the δz and θ LGM representations in terms of accuracy, with respect to the eight test data sets using: different grid sizes, $|L| = 3$, C4.5 and the composite LGM model.

in Table 4.8. The best accuracy result of 0.95 was obtained using the MSV2 data set and $d = 20$. However, the best AUC result of 0.96 was obtained using the GSV2 data set and $d = 10$. From Figures 4.6 and 4.7, it can also be seen that $d = 10$ produced a good performance. Therefore it was concluded that $d = 10$ was the most appropriate grid size for use with respect to the further evaluation presented. Note also that AUC is a better performance measure than simple accuracy, because it takes into consideration the "class priors".

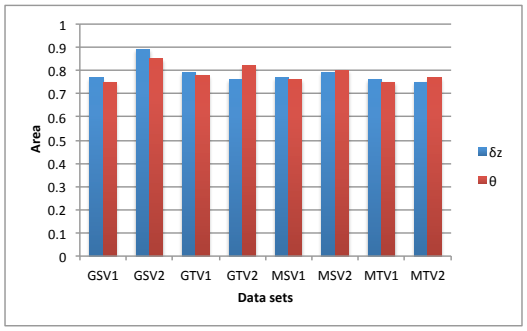
4.3.3 Identification of best label size ($|L|$)

The main objective with respect to the experiments described in this subsection was to identify the most suitable label size $|L|$ to be used with the LGM models. From the

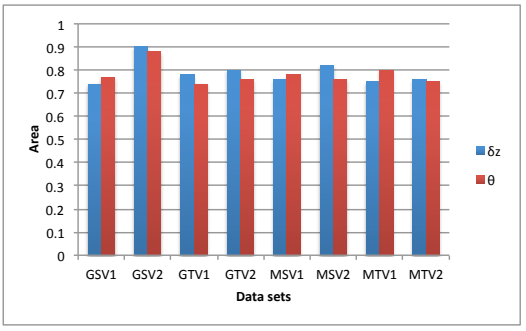


(a) AUC $(d = 2.5)$

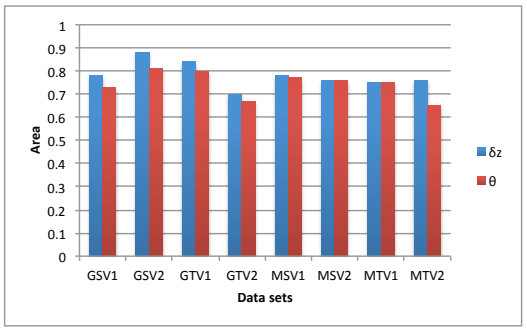
(b) AUC $(d = 5)$



(c) AUC $(d = 10)$



(d) AUC $(d = 15)$

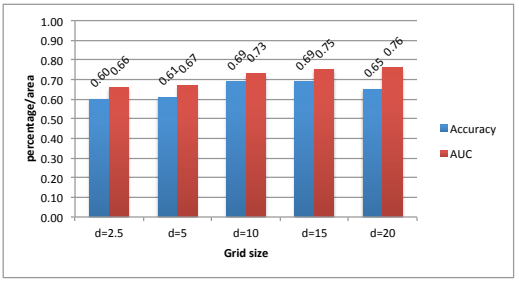


(e) AUC $(d = 20)$

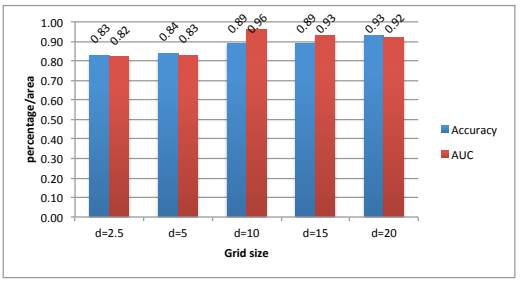
FIGURE 4.7: Comparison of the δz and θ LGM representations in terms of AUC, with respect to the eight test datasets, using: different grid sizes, $|L| = 3$, C4.5 and the composite LGM model.

d	AUC	Accuracy
2.5	0.50-0.82	0.49-0.83
5	0.63-0.83	0.54-0.84
10	0.73-0.96	0.66-0.89
15	0.75-0.93	0.67-0.89
20	0.72-0.94	0.62- 0.95

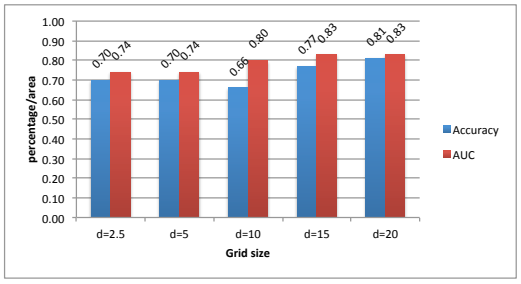
TABLE 4.8: Summary results for the obtained AUC and accuracy values (as ranges) for different grid sizes $d = \{2.5, 5, 10, 15, 20\}$.



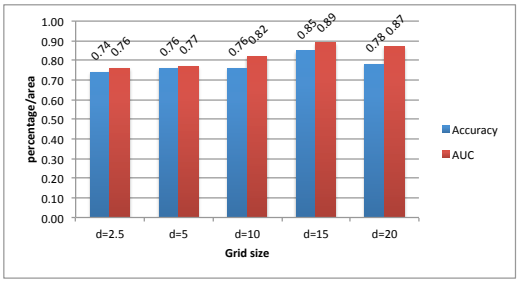
(a) $GSV1$



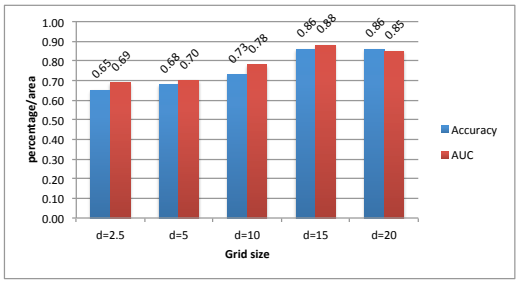
(b) $GSV2$



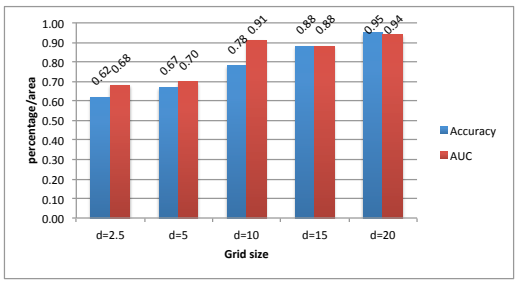
(c) $GTV1$



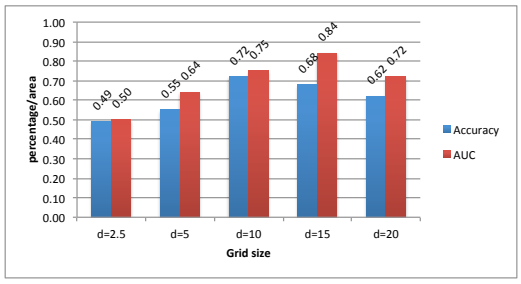
(d) $GTV2$



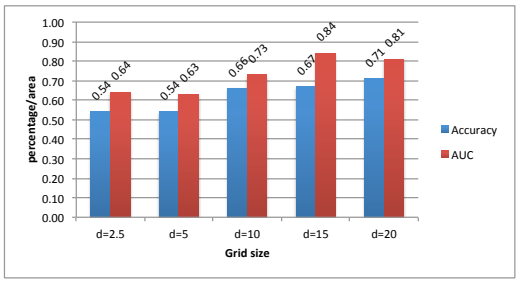
(e) $MSV1$



(f) $MSV2$



(g) $MTV1$



(h) $MTV2$

FIGURE 4.8: Comparison of different values of d in combination with the LGM model in terms of AUC, with respect to the eight datasets, using: $|L| = 3$, C4.5 and the composite LGM model coupled with δz values.

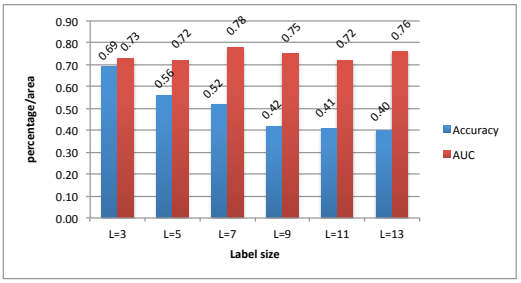
previous experiments the δz value option and $d = 10$ were chosen as both had been shown to produce good AUC results. The C4.5 algorithm and the composite LGM model were again used. A range of values for $|L|$ were considered: $\{3, 5, 7, 9, 11, 13\}$. The results are presented in Figure 4.9. From the figure it can be seen that despite the decline in the recorded accuracy values as the label size increased, the AUC was almost the same for the different Label size (by a small margin). The best AUC and accuracy results for the different data sets were obtained mostly when $|L| = 3$. For $|L| = 3$, the best AUC result obtained was 0.96 and the best accuracy result obtained was 0.89, both results were achieved with respect to the GSV2 data set. Experiments were not conducted with lower values of $|L|$ as it was conjectured that to produce a good error predictor a range of errors was required. Of course for $|L| = 1$ all records will be the same and an AUC value of 1.0 will result, a very effective classifier but of little practical use!

4.3.4 Identification of the best LGM model

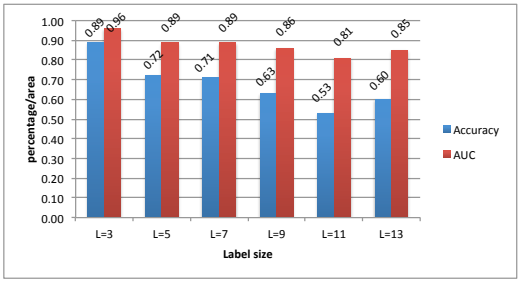
The experiments presented in this subsection were designed to determine which of the three proposed variations of the LGM representation model was the most effective, either: (i) the level one neighbourhood model, (ii) the level two neighbourhood model or (iii) the composite model where the level one and two neighbourhood models were combined. As a result of the outcomes of previous experiments (described above) the following parameters were adopted: $|L| = 3$, $d = 10$, δz value calculation and C4.5. The results obtained are presented in Figure 4.10. From the figure it can be observed that the composite and level two neighbourhood models outperformed the level one neighbourhood model. Closer inspection also indicates that the composite model performed slightly better than the level two neighbourhood model (this is why only results obtained using the composite model were reported earlier in this chapter). It was conjectured that this was because the composite and level two neighbourhood models "capture" a wider area and that this was beneficial with respect to classifier performance. The best accuracy was 0.89 obtained by the composite model. The best AUC result obtained for both the composite model and the level two neighbourhood model was 0.96 using the GSV2 data set. Further experiments (not reported here for reasons of succinctness and clarity) using alternative classification algorithms also demonstrated that the composite model outperformed the other LGM models. The same result was also produced using different values for d and $|L|$.

4.3.5 Identification of the most appropriate classification algorithms

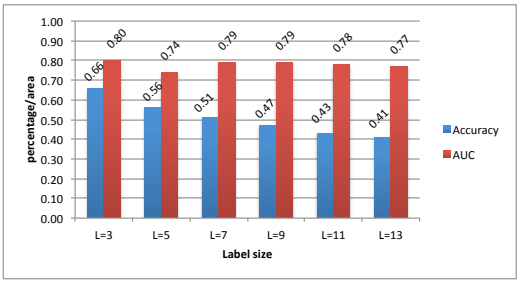
The effect of using different classification algorithms with the proposed LGM representation model was tested using five classification algorithms: (i) C4.5 [173], (ii) Bayes [116], (iii) JRIP [41] (iv) PART [72] and (v) Neural Network [23]. The composite model coupled with δz values was again used together with $d = 10$ and $|L| = 3$. The obtained results are presented in Figure 4.11. From the figure it can be seen that very little



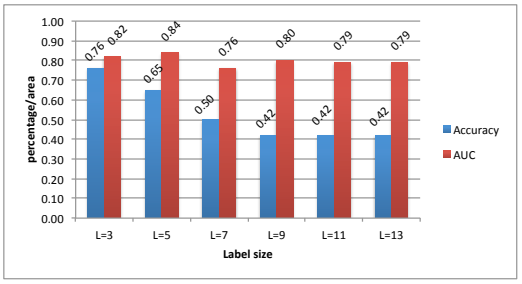
(a) $GSV1$



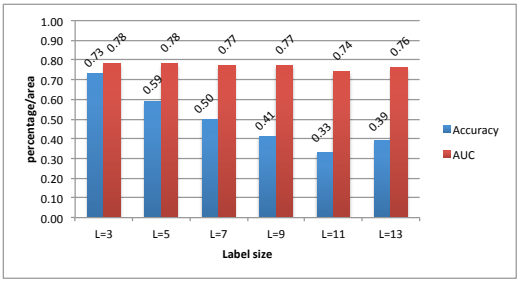
(b) $GSV2$



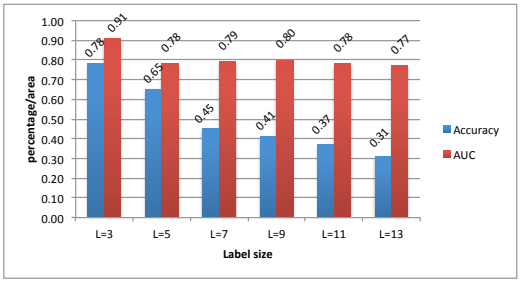
(c) $GTV1$



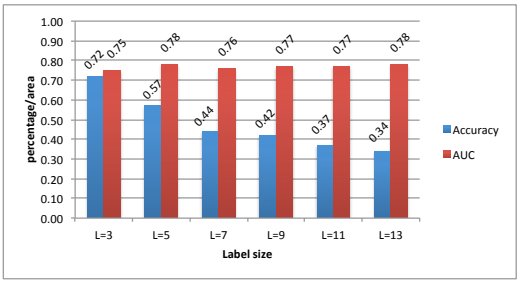
(d) $GTV2$



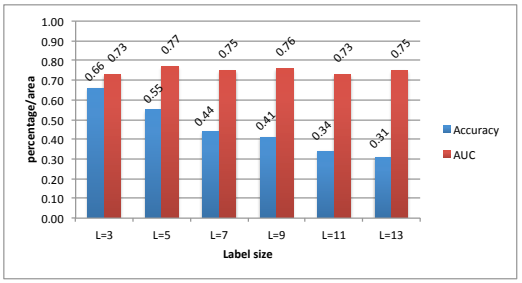
(e) $MSV1$



(f) $MSV2$

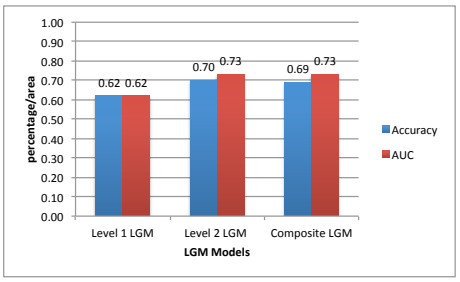


(g) $MTV1$

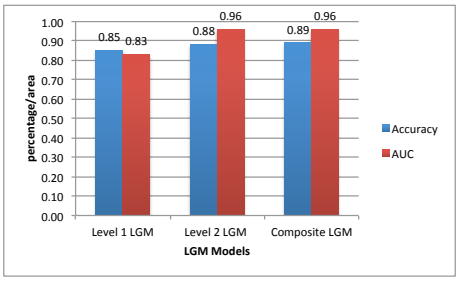


(h) $MTV2$

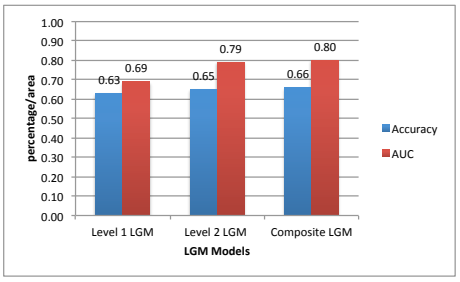
FIGURE 4.9: Comparison of different values of $|L|$ in combination with the LGM model, in terms of accuracy and AUC, with respect to the eight datasets, using: $d = 10$, C4.5 and the composite LGM model coupled with δz values.



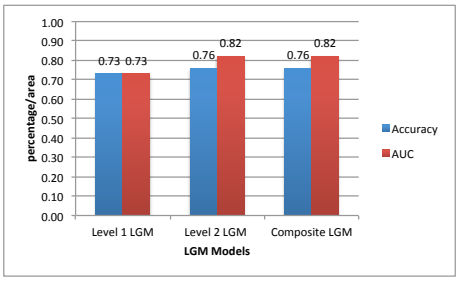
(a) $GSV1$



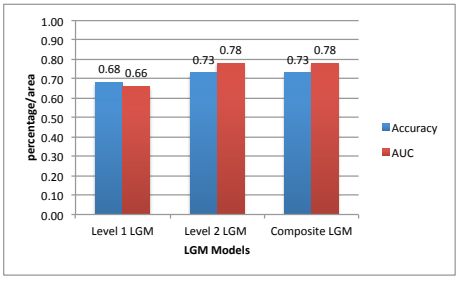
(b) $GSV2$



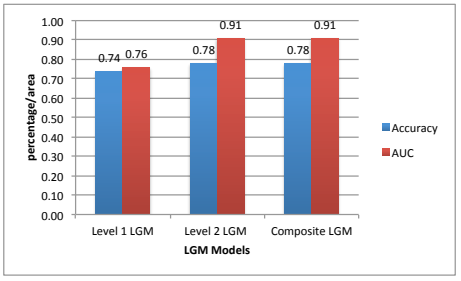
(c) $GTV1$



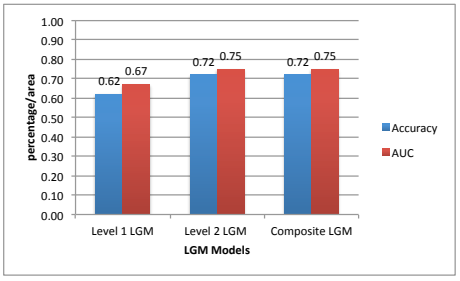
(d) $GTV2$



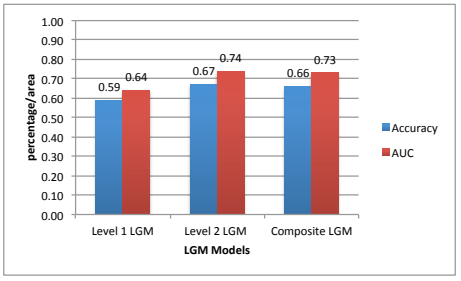
(e) $MSV1$



(f) $MSV2$



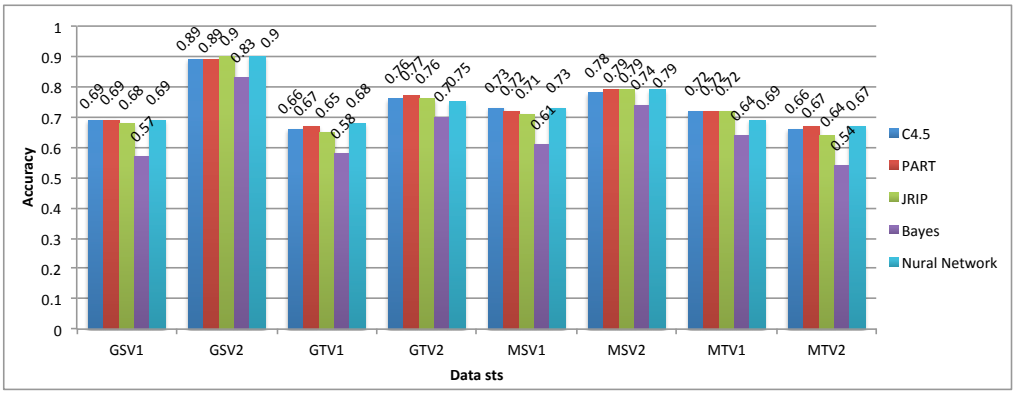
(g) $MTV1$



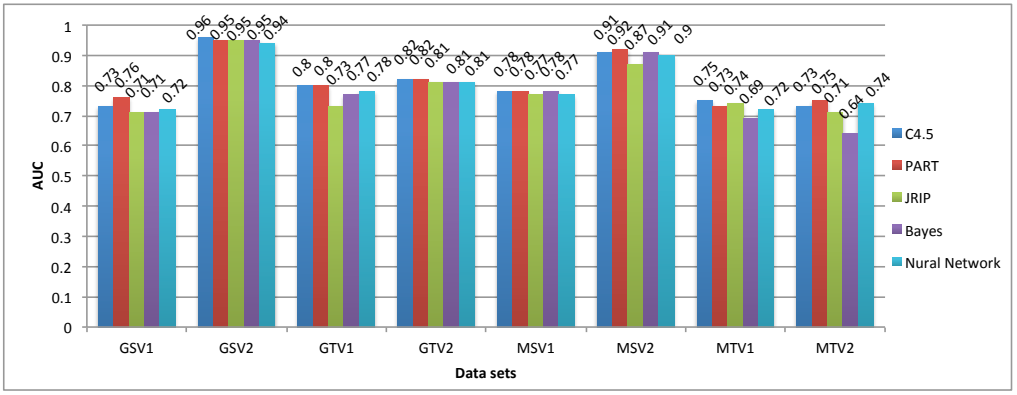
(h) $MTV2$

FIGURE 4.10: Comparison of the three variations of LGM model, in terms of accuracy and AUC, with respect to the eight datasets, using: $d = 10$, $|L| = 3$, C4.5 and δz values.

difference was recorded with respect to the performance of C4.5, Bayes, PART, JRIP and Neural Network. The best accuracy value of 0.90 was obtained with respect to the GSV2 data set using the Neural Network and JRIP classifiers. The best AUC result of 0.96 was obtained using C4.5 with respect to the GSV2 data set. Thus it can be concluded that, to an extent, the proposed LGM representation model is tolerant to the use of different classification approaches. Therefore, for the remaining experiments (and the previous experiments) described in this chapter only results obtained using C4.5 are (were) reported.



(a) *Accuracy*



(b) *AUC*

FIGURE 4.11: Comparison of use of different classifiers with the LGM model, in terms of accuracy and AUC, with respect to the eight datasets, using $d = 10$, $|L| = 3$ C4.5 and the composite LGM model coupled with δz values.

Other reasons for choosing C4.5 were: (i) it is one of the top ten most popularly used classification algorithms [216], (ii) the decision trees produced are easy to interpret and (iii) it is straight forward to extract rules from the generated decision tree classifier (if desired).

4.3.6 Training and testing the classifier on a different data set

The objective of the experiments described in this subsection was to determine whether the proposed LGM representation model was sufficiently generic, in other words whether

the model captured a sufficiently extensive sample of geometries for general application (important in the context of AISF). The experiments were conducted by training a classifier on one shape and applying it to another shape. For the experiments the following parameters were used: $d = 10$, $|L| = 3$, C4.5 and the composite LGM representation model coupled with δz values. The obtained results are presented in Figure 4.12. From the figure it can be observed that:

- In most cases, the best overall results in terms of AUC and accuracy were obtained when the classifier was trained and tested on the same data set. For example, the AUC and accuracy results were 0.73 and 0.69 and obtained when the classifier was generated using GSV1 data set. Similarly, the best overall AUC and accuracy results were 0.96 and 0.89 respectively and obtained when the classifier used GSV2 data set for training and testing.

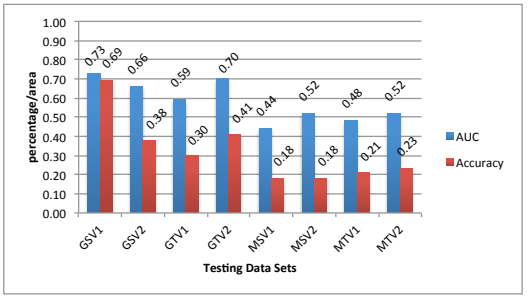
- A classifier trained on one shape using one material can be successfully applied to another shape using another material. For example, the classifier constructed using the MTV1 data set (titanium) was successfully applied to the GSV2 data set (steel); an AUC value of 0.74 and an accuracy value of 0.53 were recorded (recall that AUC takes into account the class priors while accuracy does not, therefore AUC is a better indicator of performance).

Therefore, it can be concluded that applying the classifiers trained on one shape to another shape produces good AUC results. In other words, using the LGM representation model we can produce generic classifiers that serve to capture a sufficiently wide collection of different geometries.

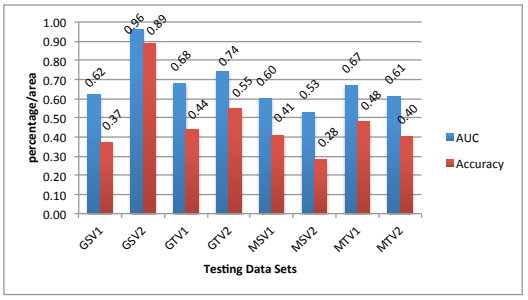
4.3.7 Run Time Analysis

The section considers the run time for the composite LGM model with δz values for a range of d values, using $L = 3$, the C4.5 classification algorithm and the eight evaluation data sets used previously (GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1, MTV2). All the experiments were performed with a 2.7 GHz Intel Core i5 PC with 4 GB 1333 MHz DDR3 memory, running OS X 10.8.1 (12B19). The code for the preprocessing phases in RASP framework, where the grid representation is applied and each centre grid point is associated with error value, was implemented using the Java programming language. Weka version 3-6-8 was used to apply the classification algorithms. Figures from 4.13 to 4.17 present the run times for a range of grid sizes $d = \{2.5, 5, 10, 15, 20\}$. Note that the figures only present recorded run times for the required preprocessing as the time required for the classification phase was found to be negligible. From the figures it can be observed that:

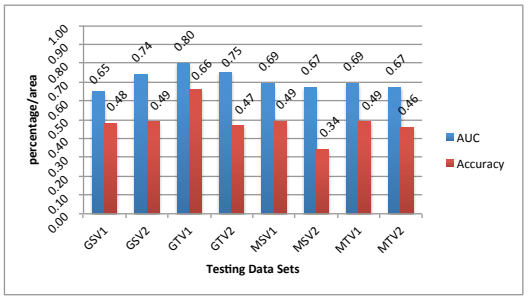
- The maximum run time was 207 seconds (to the closest second) reported when GTV2 data set was used at $d = 2.5$.



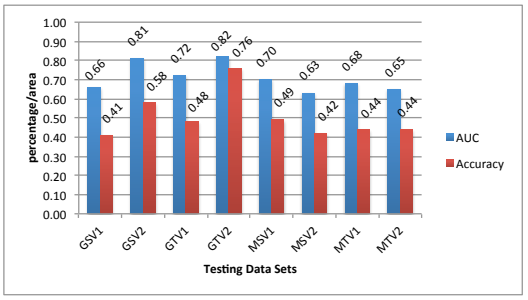
(a) *Training the generic classifier on GSV1*



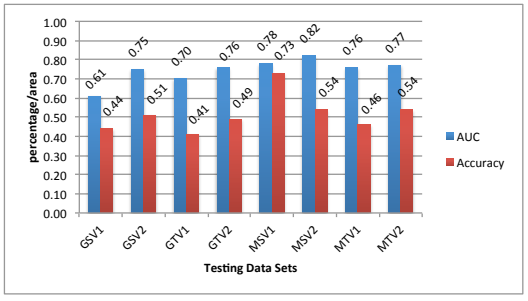
(b) *Training the generic classifier on GSV2*



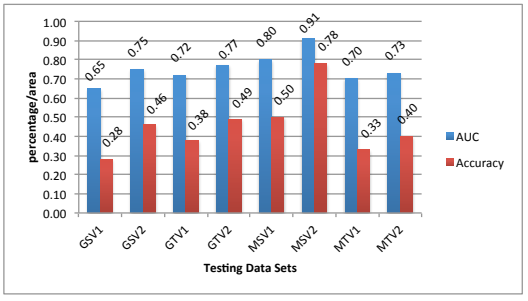
(c) *Training the generic classifier on GTV1*



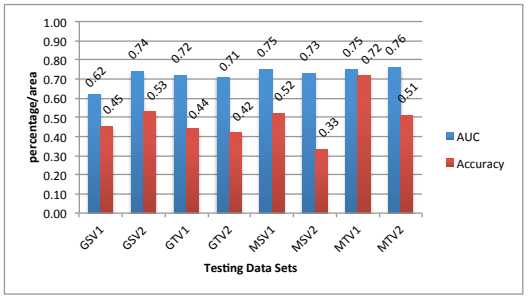
(d) *Training the generic classifier on GTV2*



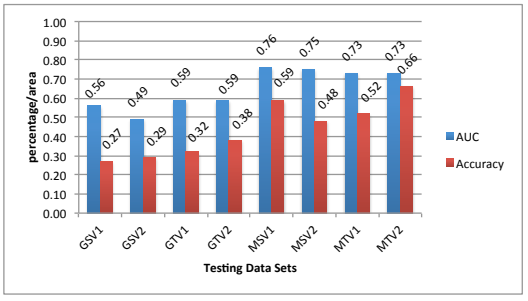
(e) *Training the generic classifier on MSV1*



(f) *Training the generic classifier on MSV2*



(g) *Training the generic classifier on MTV1*



(h) *Training the generic classifier on MTV2*

FIGURE 4.12: AUC and Accuracy results obtained when generating a generic classifier using different data sets to train and test the classifiers, δz values, $|L| = 3$, $d = 10$ and C4.5 classification algorithm.

- The minimum run time was 1 second (to the closest second) reported when each of GSV2, GTV1, GTV2, MSV1 and MSV2 data sets was used at grid size $d = 20$.
- The average run time for $d = 2.5$, $d = 5$, $d = 10$, $d = 15$ and $d = 20$ were 196.63, 6.00, 2.25, 1.63, 1.5 seconds respectively.
- The computation time increases as the grid size decreases. This is to be expected because the number of representative points (the centre points for the grid squares) in the grid representation increases when the grid size decreases and thus more processing time is required.

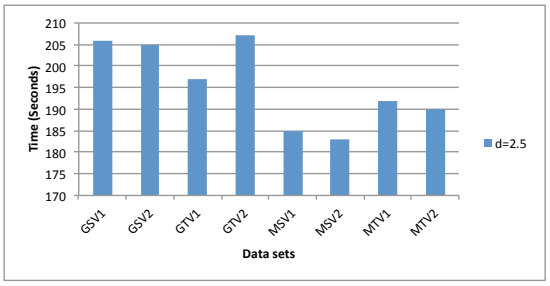


FIGURE 4.13: The run time for $d = 2.5$ for the different data sets.

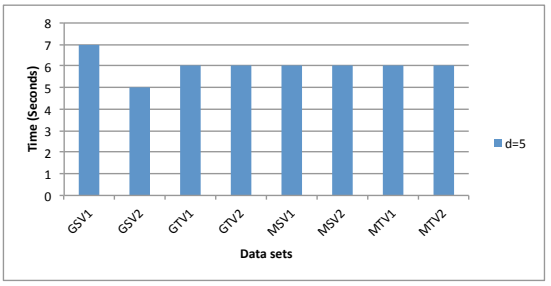


FIGURE 4.14: The run time for $d = 5$ for the different data sets.

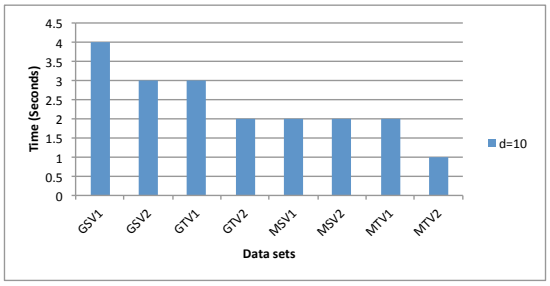


FIGURE 4.15: The run time for $d = 10$ for the different data sets.

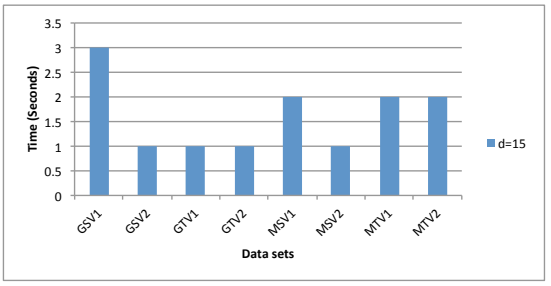


FIGURE 4.16: The run time for $d = 15$ for the different data sets.

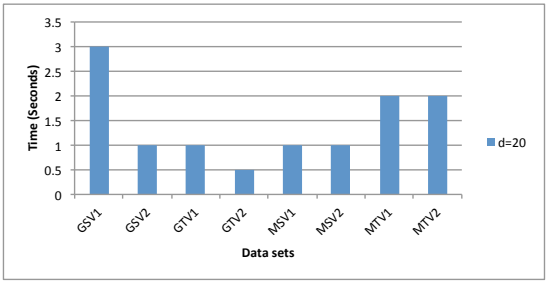


FIGURE 4.17: The run time for $d = 20$ for the different data sets.

4.4 Summary

This chapter has presented the LGM representation model, the first surface representation technique to be considered in this thesis. The LGM model was founded on the

concept of Local Binary Patterns (LBPs) that have been effectively used with respect to image texture analysis. The idea behind the LGM model was to define each grid square in term of its neighbours. Three variations of the LGM model were considered: (i) the level one neighbourhood model which considered the eight immediately surrounding neighbours, (ii) the level two neighbourhood model which considered the eight neighbours that were "one step away" and (iii) the composite model which combined the previous two models. The feature vectors extracted from the LGM representation were used to train and test classifiers. A wide range of experiments were conducted to identify: (i) the most appropriate definition for the LGM representation (δz or θ), (ii) the most appropriate grid size (d), (iii) the most effective label size ($|L|$) to predict the "springback", (iv) the most efficient LGM model (level one, level two or composite), (v) the best classification generation method to be used in relation to the LGM representation, (vi) whether it was possible to build a generic classifier and (vii) the time complexity. The main findings were:

- The δz LGM representation was the most appropriate LGM representation as it achieved the best AUC results.
- The most appropriate grid size was found to be $d = 10$.
- The most effective label size $|L|$ was founded to be $|L| = 3$.
- The composite model outperformed other models (the level one and level two neighbourhood models).
- There was no significant difference between the classification generation methods considered (C4.5, PART, JRIP, Bayes and Neural Network). However, C4.5 tended to outperform the others by a small margin.
- That a generic classifier could be generated if it was trained on a suitable shape which featured a sufficient set of patterns describing all possible geometries.

Overall it was found that the proposed LGM representation technique, with respect to its three variations (the level one, level two and the composite neighbourhood models), could successfully be used to represent 3D surfaces in the context of springback prediction. In the next two chapters, the other two proposed techniques for 3D surface representation will be presented. The Local Distance Measure (LDM) representation is presented in the following chapter.

Chapter 5

Local Distance Measure (LDM) Representation

5.1 Introduction

In this chapter the second surface representation technique considered in this thesis, the *Local Distance Measure* (LDM) representation, is presented. The technique is founded on the observation that critical features (edges and corners) are likely to have an influence over springback distribution. The idea was therefore to define each grid point contained in the grid representation of a given 3D surface in terms of its distance to its nearest edge or corner (critical point). The LDM technique comprises two main steps: (i) identification of critical points in the G_{in} input grid, and (ii) calculation of the *nearest edge distance* for each grid point in G_{in}. The proposed critical point detection mechanism is founded on the concept of identifying the angular variation between the normals between grid points and their immediate neighbours. Once the critical points have been identified, finding the nearest critical point to a given grid point is straight-forward.

The motivation for the LDM representation is that the edges and corners are a significant and integral part of any 3D shape description. In the context of this thesis an edge is where two "planes" come together, and a corner is where three or more "planes" come together. The significance of edges and corners is that they define the distinguishable geometric properties that give a 3D shape its identity. From the industrial perspective, edges and corners are considered to be the "critical points" used for manufacturing quality control [78, 197, 210]. In the context of the point cloud representation it is not immediately obvious where edges and corners occur. Recall that for the sheet metal forming error prediction application the desired 3D shape is represented by a C_{in} point cloud, while C_{out} represents the obtained 3D shape. As described previously in Chapter 3, both the clouds C_{in} and C_{out} are translated into a grid representation to give G_{in} and G_{out} respectively.

As will be seen later in this chapter a comprehensive set of experiments was conducted using the proposed LDM technique. Experiments were also conducted using

combinations of the LDM technique and the three LGM technique variations described in Chapter 4 to see if a combination of the two techniques would result in an improved outcome. Thus: (i) LDM and level one LGMs, (ii) LDM and level two LGMs and (iii) LDM and composite LGMs. As will become apparent, the evaluation of the LDM models demonstrated that describing the 3D surface using a combination of LDM and LGMs produces a better result than when considering LDM (or LGMs) in isolation (LDM coupled with composite LGMs produced the best overall performance).

The rest of this chapter is organised as follows. Section 5.2 provides a discussion of the "philosophical underpinning" for the proposed LDM approach. Section 5.3 describes the LDM technique in detail, both in terms of critical feature identification and distance calculation. Some detailed example LDM calculations are presented in Section 5.4. A discussion of the combination of the proposed LDM technique with the LGM technique described in Chapter 4 is presented in Section 5.5. An evaluation of the proposed LDM mechanism, including its operation in combination with the LGM technique, introduced in the previous chapter, is then presented in Section 5.6. Section 5.7 presents some run time analysis for the most "appropriate" variation for LDM technique. Finally, the chapter is concluded with a summary in Section 5.8.

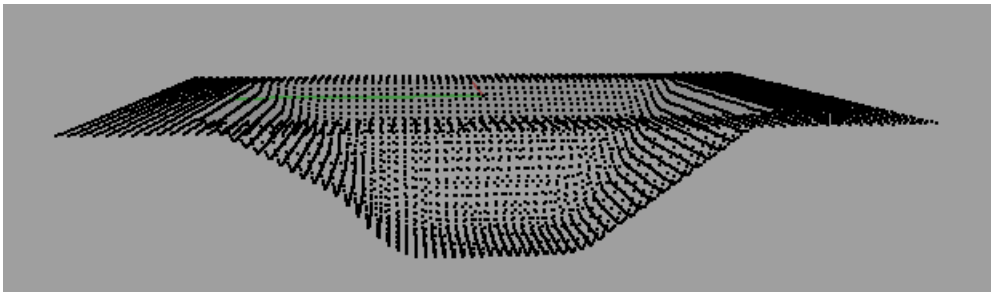


FIGURE 5.1: An example shape represented in terms of a point cloud.

5.2 Effect of proximity of Critical Points on Springback

As noted above the motivation for the LDM representation was the conjecture that springback at specific locations (grid points) was influenced by proximity to critical points (edges and/or corners). To investigate this suggestion some initial analysis was conducted to establish whether this conjecture was true or not. This analysis took the form of visualisations of the springback (error) distribution and magnitude over the sample shapes (GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2). For reference purposes an example shape is shown in Figure 5.1. The colour coding used for the visualisation is presented in Figure 5.2. Figure 5.3 presents two springback visualisations (2D-plots) for the MSV1 data set (Further visualisations illustrating springback distributions with respect to the remaining data sets are presented in Appendix A.) In all cases the grid distance d was set to 2.5 mm. Plots were also generated using other values for d, but $d = 2.5$ was found to provide the simplest to interpret visualisations.

Figure 5.3 (a) considers only the magnitude of the error where the minimum springback (zero) has an error code of 1 (pale yellow) and the maximum an error code of 7 (dark green). The range of springback magnitude values is from 0 mm (yellow) to 1.8 mm (dark green). It can be seen, from the figure, that the larger springback values tend to be concentrated on the pyramid sides away from edges. The lower springback values tend to be concentrated at edges and over the flat-topped square area of the pyramid.

Figure 5.3 (b) takes into consideration both direction and magnitude, thus the maximum negative springback has an error code of 1 (pale yellow) and the maximum positive springback has an error code of 7 (dark green). From the figures the shape of the flat-topped square pyramid can be clearly detected. The range of errors is from -1.22 mm (yellow) to $+1.80$ mm (dark green). The minus ($-$) and plus ($+$) signs indicate that the direction of the springback is either outwards or inwards respectively. The first observation that can be made, as is to be expected, is that the error distribution is similar to that noted with respect to Figure 5.3(a). Further inspection indicates predominantly positive springback values round the base of the pyramid and negative springback on the sides of the pyramid.

Similar observations, as the above, can be drawn from the additional springback distribution visualisations presented on Appendix A. In conclusion, what is clear form the figures is that the degree of springback does seem to be related (at least in part) to distance from edges and corners although not in an entirely symmetric manner.

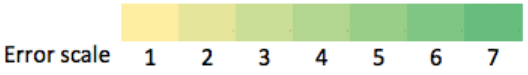
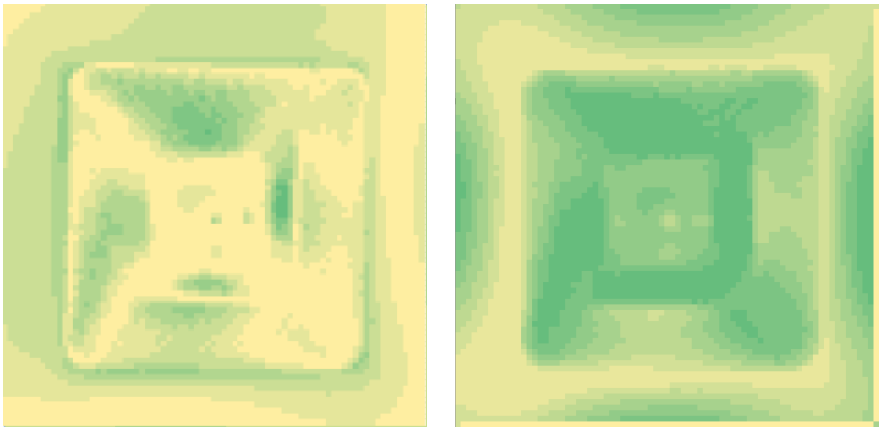


FIGURE 5.2: Colour coding used in Figure 5.3.



(a) *Undirected MSV*1 $d = 2.5$ (b) *Directed MSV*1 $d = 2.5$

FIGURE 5.3: 2D-plot showing springback distribution over a shape (MSV1): (a) magnitude only, (b) magnitude and direction ($d = 2.5$).

5.3 The LDM Mechanism

The proposed LDM technique is described in this section. As already noted in the introduction to this chapter the proposed technique comprises two stages: (i) critical point identification, and (ii) nearest edge distance calculation. Thus this section is divided into two sub-sections describing stages one and two respectively. On completion of the process each grid point will be described in terms of the distance to the nearest critical point. In the case of training data this grid point was coupled with an associated springback (error) value.

5.3.1 Critical Point Detection

Different techniques have been proposed in the literature to detect the critical points (edges and/or corners) for different purposes such as [46, 118, 139, 165, 220]. The underpinning philosophy supporting the proposed critical point detection mechanism was that the technique should be computationally simple as it would need to be applied many times (according to the number of grid centres in the input data). The proposed detection mechanism is founded on normal calculations and comparisons for each grid point located on the G_{in}. A point p_i was considered to be a critical point if the angular difference (θ) between the normal to the point p_i and at least one of the normals associated with the eight level one neighbours is greater than some threshold. The proposed critical point detection mechanism thus operates as follows: for each point p_i located in G_{in}, a sequence of eight angles $\{\theta_1, \theta_2, \theta_3, \ldots, \theta_8\}$ were calculated using the *dot product* between the normal to p_i and the normal to each of its level one neighbours. For instance, if the p_i normal is $n_i = \langle x_i, y_i, z_i \rangle$, where the normal length is defined as $|\vec{n_i}| = \sqrt{(x_i)^2 + (y_i)^2 + (z_i)^2}$; and the normal for the neighbour p_{i1} is $n_{i1} = \langle x_{i1}, y_{i1}, z_{i1} \rangle$, where its length defined as $|\vec{n_{i1}}| = \sqrt{(x_{i1})^2 + (y_{i1})^2 + (z_{i1})^2}$; then the angle θ_{i1} between n_i and n_{i1} is defined as:

$$\theta_{i1} = \arccos \frac{(x_i \cdot x_{i1}) + (y_i \cdot y_{i1}) + (z_i \cdot z_{i1})}{|\vec{n_i}| \cdot |\vec{n_{i1}}|} \tag{5.1}$$

where θ_1 is the smallest angle between the two normals n_i and n_{i1}. The range for θ_i is from 0° (where both normals run parallel to each other in the same direction) to 180° (where both normals run parallel to each other but in opposite directions). A point p_i is considered to be a critical point if at least one of the θ values for one of its level one neighbours is greater than some threshold (tolerance measure) ξ. Algorithm 5.1 itemises the steps for identifying the critical feature points in G_{in}. There are two main factors that affect the process of the critical point detection mechanism:

1. The grid size (d).
2. The tolerance value ξ.

Considering the grid size first. The greater the grid size the more difficult it is to detect edges or corners. Recall that the normal to the centre grid point is actually calculated

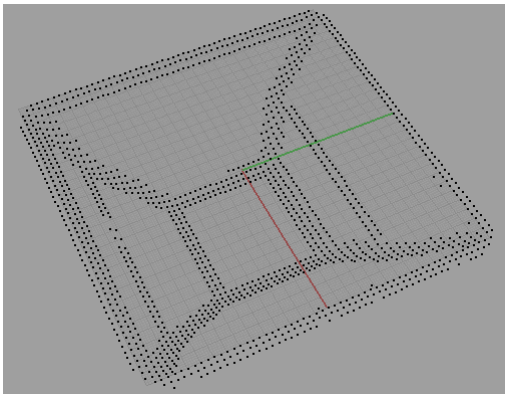
Algorithm 5.1: Critical Point Detection

Input: G, ξ
Output: Updated G with identified critical points

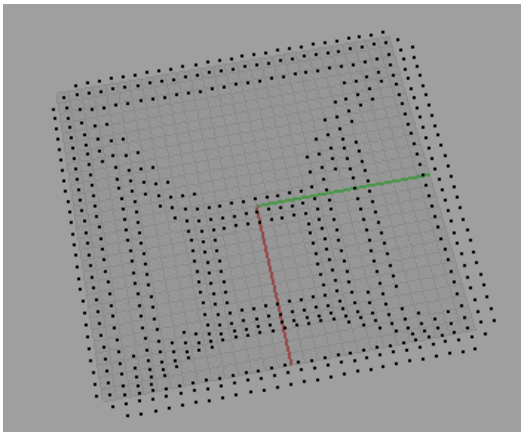
for *each* $p_i \in G$ **do**
  $p_i.status \leftarrow$ "$non - critical$";
  Identify the normal n_i for the p_i;
  Identify level one neighbourhood for p_i, $Nr = \{p_{i1}, p_{i2}, ..., p_{i8}\}$;
  Define the eight normals for the the level one neighbourhood N;
  $\theta \leftarrow \phi$; `// List of` θ_i `associated with` p_i
  `// Find the length of the centre point normal` $\vec{n}$
  $|\vec{n}| \leftarrow \sqrt{(n_x)^2 + (n_y)^2 + (n_z)^2}$;
  foreach $n_i \in N$ **do**
    $|\vec{n_i}| \leftarrow \sqrt{(n_{ix})^2 + (n_{iy})^2 + (n_{iz})^2}$;
    $\theta_i \leftarrow \arccos(\frac{(n_x \cdot n_{ix}) + (n_y \cdot n_{iy}) + (n_z \cdot n_{iz})}{|\vec{n}| \cdot |\vec{n_i}|})$;
    $\theta \leftarrow$ add θ_i;
  end
  foreach $\theta_i \in \theta$ **do**
    if $\theta_i \geq \xi$ **then**
      $p_i.status \leftarrow$ "$critical$";
      break;
    end
  end
end

as an average of a number of normals (dot product calculations) as described previously in Chapter 3. When the grid size is small this calculation is quite precise, but as the grid size increases the differences between the normals starts to increase and hence the averaging also starts to become imprecise, because for a relatively large value of d the grid square will cover a larger area. In the context of critical point identification, as the grid size increases the difference between normals starts to become less distinct and hence normals are more likely to be parallel than if a small value of d is used. The critical points are located on: (i) the square base, (ii) the square top, (iii) the junction of the four pyramid walls and the bends in two sides as shown in Figures 5.4(a), 5.5(a) and 5.5(b). For example, if we consider the GSV1 data set (shown previously in Chapter 3 in Figure 3.8) and apply the critical point identification process to this shape using a variety of d values from $d = 2.5$ to $d = 20$, and using $\xi = 9$ the results are as shown in Figure 5.4. From this figure it can be seen that as d increases critical point detection becomes less precise. When $d = 2.5$ critical points (edges and corners) are clearly identified, however when $d = 20$ the result is unusable. The situation can be addressed by using different ξ values with respect to different d values.

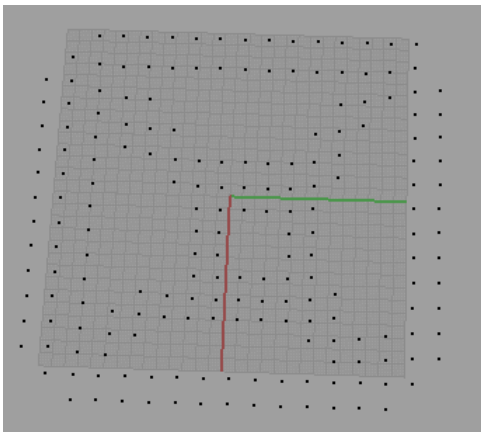
The second influencing factor is the nature of the selected ξ tolerance value. Obviously, as ξ is increased, the number of points identified as edge points decreases. Conversely as ξ is decreased, the number of points identified as edge points increases. This can be illustrated from a sequence of experiments using different values of ξ but



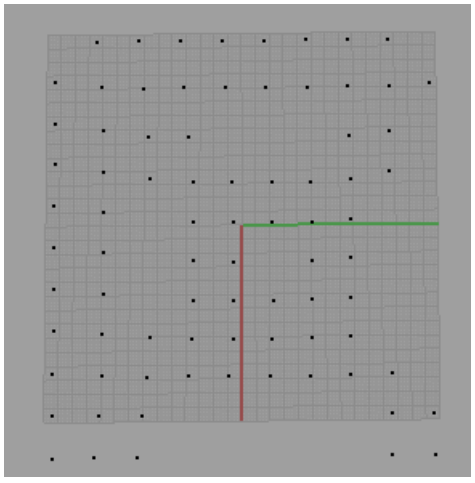
(a) $GSV1$ $d = 2.5$



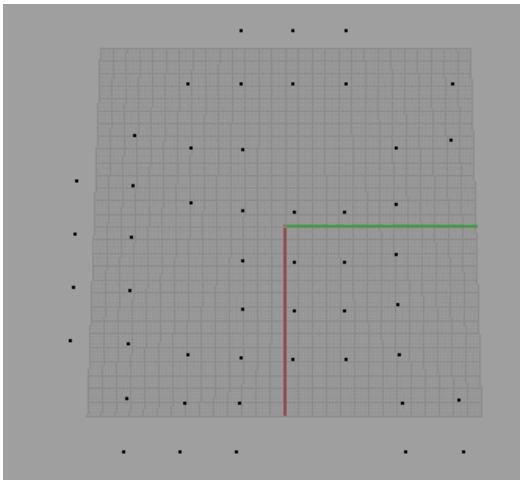
(b) $GSV1$ $d = 5$



(c) $GSV1$ $d = 10$



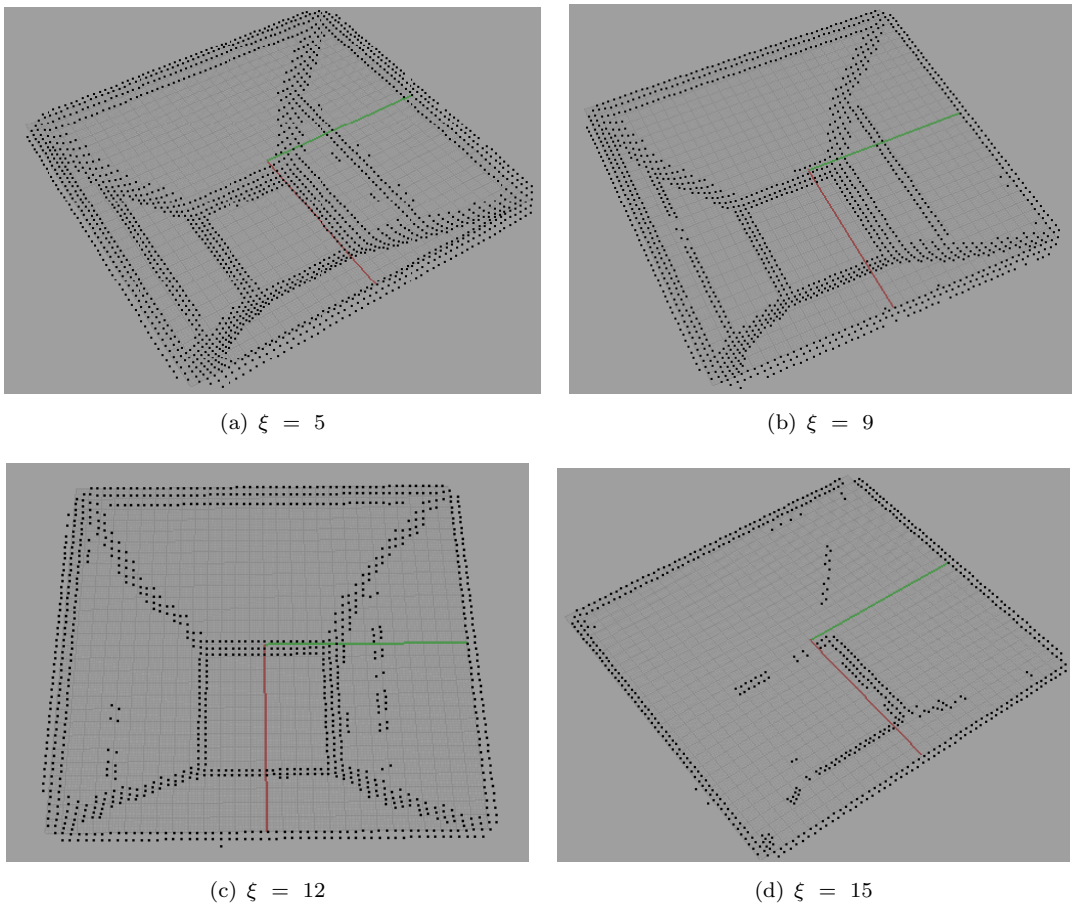
(d) $GSV1$ $d = 15$



(e) $GSV1$ $d = 20$

FIGURE 5.4: The effect of different d values on critical point detection using $\xi = 9$ and the GSV1 data set.

with a constant value of d as shown in Figure 5.5. Careful inspection of this figure demonstrates that more points are identified as critical points when $\xi = 5$ than when $\xi = 15$. Therefore, ξ should be carefully chosen. More experiments to determine the most appropriate value for ξ to be associated with each d value are presented in Section 5.6.1.



(a) $\xi = 5$ (b) $\xi = 9$

(c) $\xi = 12$ (d) $\xi = 15$

FIGURE 5.5: The effect of different ξ values on critical point detection using $d = 2.5$ and the GSV1 data set.

5.3.2 Distance Calculation

Once the critical points have been detected and determined on G_{in}, as previously described in section 5.3.1, the minimum distance between each grid point p_i and its closest critical point can be determined simply by adopting a "region growing" process as shown in Algorithm 5.2. The input to Algorithm 5.2 is again the grid G_{in}. The algorithm recursively tests all the points located on G_{in} (line 1) in order to calculate the critical distance as long as each p_i is not itself a critical point (if so, a distance of 0 is returned). If p_i is not a critical point, the algorithm proceeds in iterative manner, level by level, starting with level 1 neighbours. The minimum distance from p_i to its current level neighbours is calculated using the procedure *Distance* (line 7). This procedure returns: (i) -1 if the region growing has resulted in a set of points outside of G (this might occur

in the event of a uniform flat plane where a high value of ξ and/or d is used), (ii) 0 if no critical points are found at the given level or (iii) a distance otherwise. If a distance of -1 is returned a distance of 0 is recoded and the algorithm terminates. Otherwise, the algorithm continues with the next level as shown in Figure 5.6. The distance between the current grid point p_i and the discovered critical grid points will be calculated using Equation 5.2.

$$d(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{5.2}$$

Algorithm 5.2: RegionGrowing

```
   Input: G
   Output: G′ associated with "critical" distance values
 1 for all p_i ∈ G do
 2     if p_i != "critical" then
 3         dist ← 0;
 4         found ← false ;
 5         level ← 1;
 6         while ¬found do
 7             dist ← Distance(p_i, level) ;
 8             if dist = −1 then
 9                 p_i.distance ← 0;
10                 found ← true;
11             end
12             else if dist > 0 then
13                 p_i.distance ← dist;
14                 found ← true;
15             end
16             else
17                 level + +;
18             end
19         end
20     end
21 end
22 return G′;
```

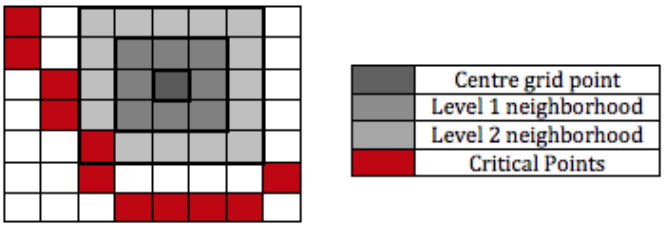


FIGURE 5.6: Region growing example using Algorithm 5.2 where the closest critical point is located within the level two neighbourhood.

Function Distance(p,l)

$P \leftarrow$ set of level l points in G calculated with respect to p;
$d \leftarrow$ array of distance values for critical points found in P;
$dist \leftarrow 0$;
$d \leftarrow dist$;
if $P \longleftarrow \emptyset$ **then**
  $dist \leftarrow -1$;
end
else
  for *all* $p_j \in P$ **do**
    if $p_j.status =$ "$critical''$ **then**
      $dist_j \leftarrow \sqrt{(p_j x - px)^2 + (p_j y - py)^2 + (p_j z - pz)^2}$;
      $d \leftarrow dist_j$;
    end
  end
  $dist \leftarrow minimumdistancevalueind$;
end
return $dist$;

Again, as in the case of the LGM technique a set of qualitative labels L was used to describe the range of possible local distance measures given a particular geometry. Some statistical information for the critical point detection method with respect to both the Gonzalo and Modified pyramids is presented in Table 5.1.

5.4 LDM Detailed Examples

This section presents two worked examples illustrating: (i) critical point detection and (ii) local distance measure calculation. The first example, presented below, shows how critical point are identified using Algorithm 5.1. The second example shows how the distance to the nearest critical point is obtained using Algorithm 5.2.

Example 1: Critical Grid Point Identification.
Consider a hemisphere shape for a given a point $p_0 = (-32.50, -66.96, -0.78)$ associated with a normal of $n_0 = \langle 0.04, 10.07, 18.75 \rangle$ and a set of four points $\{p_1, p_2, p_3, p_4\}$ describing a level one neighbourhood as shown in Figure 5.7. The points' coordinates are as shown in Table 5.2. A set of normals to these points are also listed in the table along with θ values (the angle between the normals of theses points and the normal to p_0). By using $\xi = 9$; all θ values are greater than ξ (as shown in the table) and hence point p_0 would be identified as *critical* point.

TABLE 5.1: Statistical information for the proposed critical point detection technique with respect to the evaluation data sets.

	d	Number of critical points	Critical Points as a percentage (%) of total size
GSV1	2.5	1372	22.54%
	5	618	40.58%
	10	195	48.51%
	15	82	47.95%
	20	49	48.04%
GSV2	2.5	1414	23.23%
	5	620	40.71%
	10	195	48.51%
	15	88	51.46%
	20	51	50.00%
GTV1	2.5	1374	23.18%
	5	640	43.16%
	10	204	53.54%
	15	86	50.29%
	20	50	49.02%
GTV2	2.5	1372	22.54%
	5	618	40.58%
	10	195	48.51%
	15	82	47.95%
	20	49	48.04%
MSV1, MSV2, MTV1, MTV2	2.5	696	11.89%
	5	583	39.31%
	10	201	52.76%
	15	70	40.94%
	20	40	39.22%

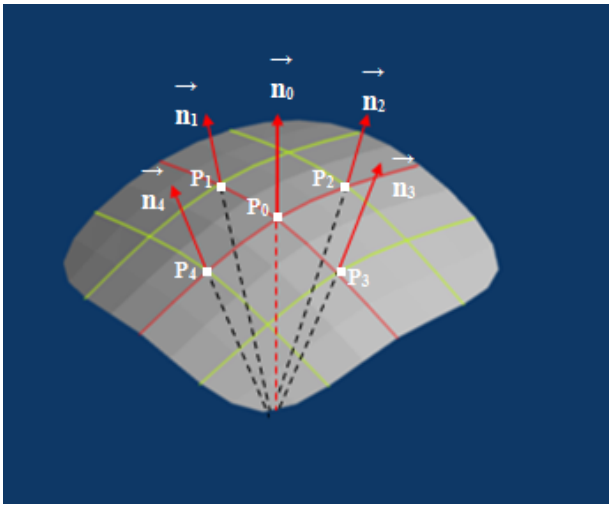


FIGURE 5.7: Example of a hemisphere shape for a given point p_0 with four Level one neighbourhood points p_1, p_2, p_3 and p_4 each associated with its own normal, $\vec{n_1}$, $\vec{n_2}$, $\vec{n_3}$ and $\vec{n_4}$ respectively.

TABLE 5.2: The level one neighbourhood point coordinates, normals (n_i) and the angles (θ_i°) between the normal of the centre grid point p_0 and the neighbouring normals.

p_i	(x_i, y_i, z_i)	n_i	θ_i° (angle with reference to p_0)
p_1	$(-32.5, -61.96, -7.18)$	$\langle -1.41 \times 10^{-4}, 5.76 \times 10^{-5}, 18.75 \rangle$	28.23°
p_2	$(-27.5, -66.96, -0.80)$	$\langle -1.61 \times 10^{-4}, -1.42 \times 10^{-5}, 18.75 \rangle$	28.23°
p_3	$(-32.5, -71.96, 0.05)$	$\langle -1.12 \times 10^{-4}, 2.05 \times 10^{-5}, 18.75 \rangle$	28.23°
p_4	$(-37.5, -66.96, -0.77)$	$\langle -1.55 \times 10^{-4}, 2.72 \times 10^{-5}, 18.75 \rangle$	28.23°

Example 2: Local distance measurement.

Given a *non critical* point $p_1 = (-60.13, -85.13, 0.00)$ and a *critical* point $p_2 = (-60.13, -70.13, 0.00)$ within the level *three* neighbourhood of p_1, the minimum distance between p_1 and p_2 is:

$$\begin{aligned} d(p_1, p_2) &= \sqrt{(0)^2 + (15)^2 + (0)^2} \\ d(p_1, p_2) &= \sqrt{(225.00)} \\ d(p_1, p_2) &= 15.00 \; mm \end{aligned} \tag{5.3}$$

5.5 Combining The LDM Model with The LGM Model

The LDM technique, as described above, was used to represent 3D surfaces of interest in terms of the distance between each point located on G_{in} and the closest critical point. It was conjectured that the effectiveness of the LDM approach might be improved if it was combined with the LGM representation described in the previous chapter (Chapter 4). Recall that three variations of the LGM model were considered: (i) the level one LGM model, (ii) the level two LGM model and (iii) the composite LGM model of which the composite model produced the most effective result. The LDM representation proposed in this chapter can thus be combined with each of these models to give:

1. LDM + Level 1 LGM
2. LDM + Level 2 LGM
3. LDM + Composite LGM

The essential idea was that combining the two approaches would provide for a more detailed representation of 3D surfaces. Information concerning the number of attributes associated with the LDM and LGM combinations is presented Table 5.3 (for comparison purposes the number of attributes associated with the LDM technique is included in the table). The significance is that more attributes require more storage and thus more processing time. With respect to the different models, the number of attributes generated using different values of $|L|$ are presented in Table 5.4. The significance is that when the grid point p_i has n features then there would be L^n geometrical patterns describing the *error (springback)* associated with p_i.

TABLE 5.3: Number of features and the generated feature vector for p_i.

LDM model	No. of Features	The generated feature vector
LDM	2	$\langle dist, e_i \rangle$
LDM + Level 1 LGM	10	$\langle \delta z_{i,1}, \delta z_{i,2}, ..., \delta z_{i,8}, dist, e_i \rangle$
LDM + Level 2 LGM	10	$\langle \delta z_{i,9}, \delta z_{i,10}, ..., \delta z_{i,16}, dist, e_i \rangle$
LDM + Composite LGM	18	$\langle \delta z_{i,1}, \delta z_{i,2}, ..., \delta z_{i,16}, dist, e_i \rangle$

TABLE 5.4: Number of attributes (including the *error* class) for each LDM model using a range of label sizes.

	Label Set Size ($\|L\|$)					
	3	5	7	9	11	13
LDM	6	10	14	18	22	26
LDM + Level 1 LGM	30	50	70	90	110	130
LDM + Level 2 LGM	30	50	70	90	110	130
LDM+ Composite LGM	54	90	126	162	198	234

5.6 Experiments and Evaluation

To evaluate the proposed LDM technique, and the three combinations with the LGM technique proposed in the previous chapter, a number of experiments were conducted using the evaluation data sets introduced in Chapter 3 and the RASP framework. Recall that the RASP framework comprises three phases: (i) data pre-processing where the initial input grid representation is generated and error calculation performed, (ii) surface representation (resulting in a set of feature vectors in the case of the LGM and LDM representation) and (iii) classifier generation and evaluation. The main objectives for the experiments with respect to the LDM representation were:

1. To identify the most appropriate value for ξ associated with respect to each grid size d.
2. To identify the most appropriate value for d with respect to the LDM model and its combinations.
3. To identify the best label size $|L|$ for the LDM model.
4. To identify the most appropriate overall representation (LDM, LDM + Level 1 LGM, LDM + Level 2 LGM and LDM + Composite LGM).
5. To investigate further (in addition to the investigations considered in the previous chapter) whether a generic classifier, trained on one shape and applied to another, can be produced.
6. To conduct some run time analysis concerning both the LDM representation on its own, and the best performing representation identified so far.

Each experiment is considered in further detail in the following six subsections. All experimental results were recorded in terms of Accuracy and the Area Under ROC Curve (AUC) [68, 79]. Work presented in the previous chapter demonstrated that there was no

significant difference between the different classification algorithms: (i) C4.5 [173], (ii) Bayes [116], (iii) JRIP [41], (iv) PART [72] and (v) Neural Network [23]. Nevertheless, C4.5 classifier was selected and identified as the "best" with respect to the LGM model because of its simplicity and popularity, hence this was used again with respect to the evaluation reported in this chapter, coupled with Ten Cross Validation (TCV).

5.6.1 Best value for ξ

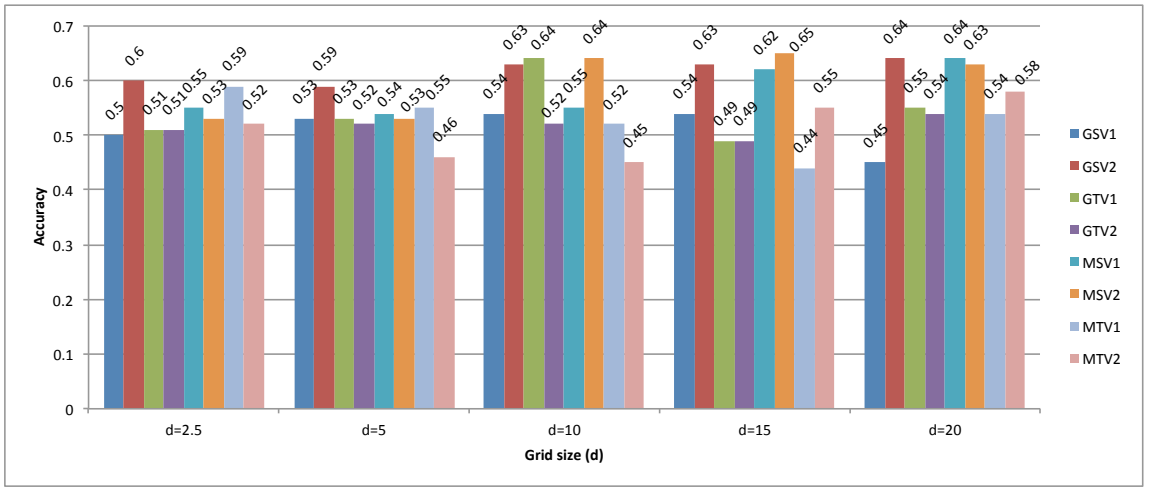
For the evaluation reported in this chapter the ξ values suggested in Table 5.5 were used with respect to different grid sizes. These values were obtained through a process of empirical experimentation. From the table it can be observed that the most appropriate value for ξ increases with d, this is because, as noted above, as d increases the likelihood of high θ values (the difference between the normal at a point and one of its neighbouring normals) increases, hence a higher value of ξ is required.

TABLE 5.5: The tolerance value ξ associated with different grid sizes d

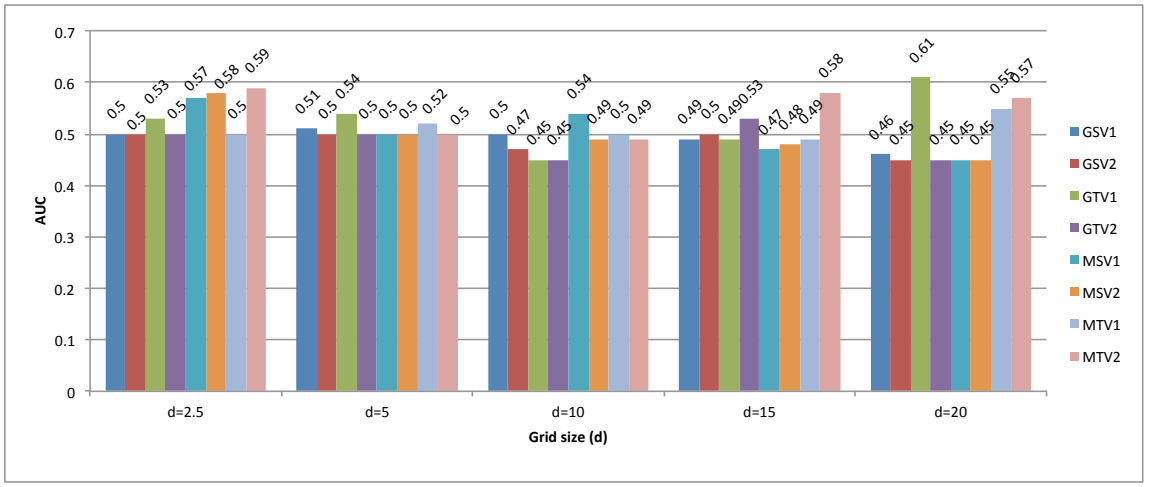
The grid size d	The tolerance value ξ
2.5	9
5	9
10	15
15	18
20	20

5.6.2 Identification of the best value for d (grid size)

Some analysis regarding the most effective grid size with respect to the LDM representation has already been presented earlier in this chapter (see Section 5.3.1). In this section the results obtained from a more comprehensive evaluation of the effect of grid size is presented using the complete collection of evaluation data sets. For the experiments the same range of d values were considered as before: $\{2.5, 5, 10, 15, 20\}$ (mm). $|L| = 3$ was kept constant and the appropriate ξ value as previously shown in Section 5.6.1 with respect to each grid size was used. The results are presented in Figure 5.8. From the figure it can be seen that the best AUC was 0.61 when $d = 20$ achieved using GTV1. The best accuracy result was 0.65 when $d = 15$ achieved using MSV2. Despite the increase in the accuracy results for the grid size $d = 10$, $d = 15$ and $d = 20$, the associated AUC results were below 0.50. Consequently $d = 2.5$ was selected as the most effective grid size for the LDM model as all the AUC results recorded using $d = 2.5$ were greater than or equal 0.50. Accordingly, 0.60 is the best accuracy achieved for $d = 2.5$ using GSV2 (with AUC result = 0.50) while the best AUC for $d = 2.5$ was 0.59 achieved by MTV2 (with accuracy result = 0.52). It was acknowledged that these accuracy and AUC values are not particularly good, but the objective of the reported experiments was to identify a best grid size d for further experimentation rather than identifying a best overall set of parameters straight off.



(a) *Accuracy results for LDM model to identify the best grid size*



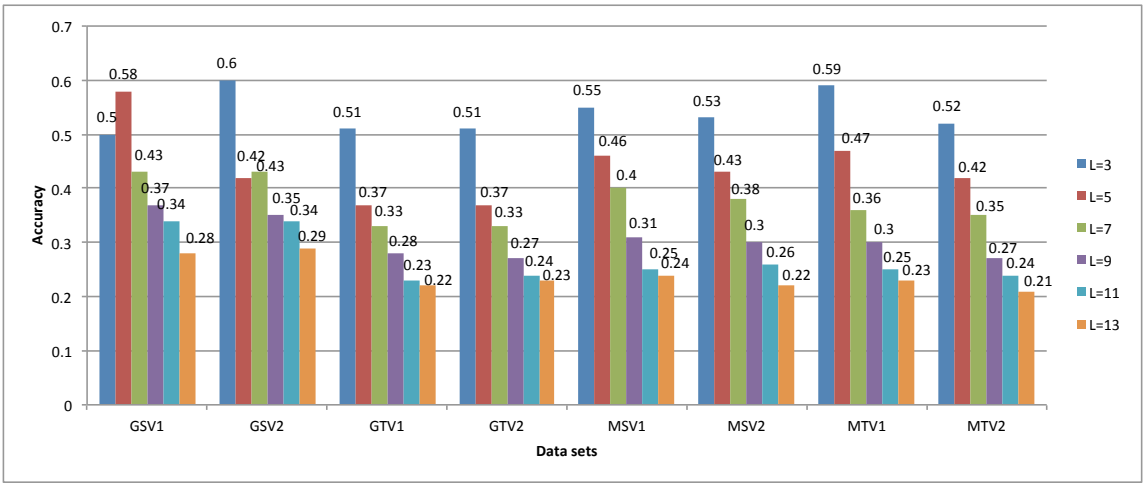
(b) *AUC results for LDM model to identify the best grid size*

FIGURE 5.8: Accuracy and AUC results obtained for the LDM model, using different values for d with respect to the eight test datasets (using $|L| = 3$ and the C4.5 classification algorithm).

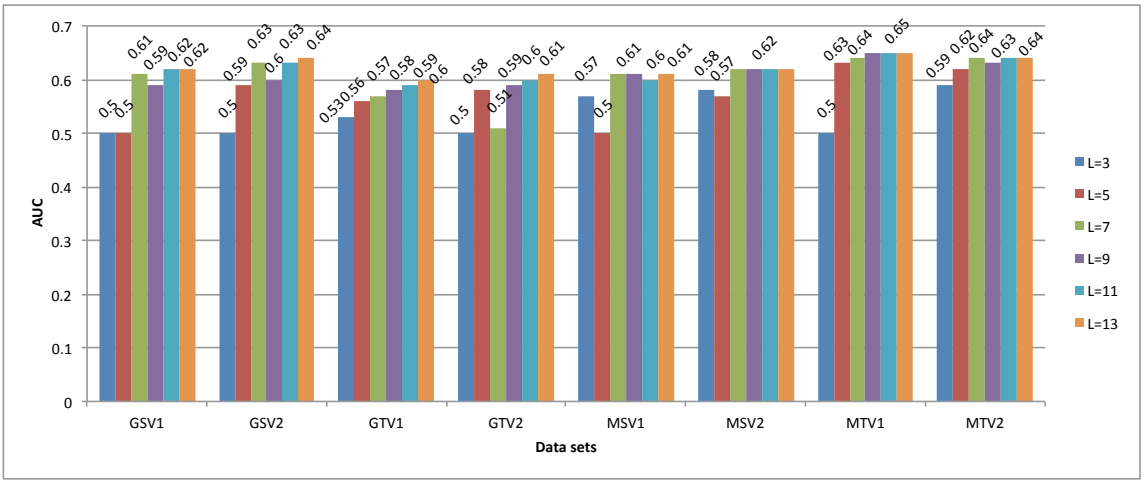
5.6.3 Best number of labels ($|L|$)

The main objective with respect to the experiments described in this sub-section was to identify the most suitable label set size $|L|$ to be used with the LDM model. A range of values for $|L|$ were considered: $\{3, 5, 7, 9, 11, 13\}$. From previous experiments $d = 2.5$ had been shown to be the most effective grid size and thus this was chosen for the d parameter. The results are presented in Figure 5.9. From the figure it can be seen that $|L| = 3$ produced the best accuracy and AUC results compared to other label sizes (0.59 and 0.60 was recorded). Of course as $|L|$ gets smaller the fewer the number of springback classes and hence the classification is more likely to be correct. As noted previously, using $|L| = 1$ we can expect to produce a classification accuracy of 100% and an AUC of 1.0. It should also be noted that for springback correction purposes a

value of $|L|$ that is greater than three may be more useful. However, with respect to the remaining experiments presented in this chapter $|L| = 3$ was used.



(a) *Accuracy results for LDM model to identify the best label size*



(b) *AUC results for LDM model to identify the best label size*

FIGURE 5.9: Accuracy and AUC results obtained for the LDM model using different values for $|L|$ with respect to the eight evaluation datasets (using $d = 2.5$ and the C4.5).

5.6.4 Best LDM model

The experiments described in this subsection were designed to determine whether the LDM model on its own or in combination with the three proposed LGM models was the most effective. The models tested were: (i) LDM, (ii) LDM + composite LGM, (iii) LDM + level one LGM and (iv) LDM + level two LGM. For LDM evaluation, and as a result of the outcomes of the previous experiments, the following parameters were adopted: $|L| = 3$ and C4.5. However, in Chapter 4 $L = 3$ and $d = 10$ were identified as the best parameters for the LGM models. Therefore the experiments used to identify the best LDM model described in this section were conducted using a range of grid sizes

$d = \{2.5, 5, 10\}$. The results obtained are presented in Figures 5.10. From the figure it can be observed that:

1. The LDM + composite LGM and LDM + level two LGM outperformed the other models with respect to $|L| = 3$ for different grid sizes.

2. The best accuracy result achieved was 0.92 obtained using the LDM + composite LGM model applied to GSV2 with $d = 10$.

3. The best AUC result achieved was 0.96 again obtained using the LDM + composite LGM model when applied to GSV2 with $d = 10$.

4. There is no significant difference between the performance for the LDM + level one LGM, LDM + level two LGM and LDM + composite LGM models when the grid size was small ($d = 2.5$ and $d = 5$). However, significant differences in the performance measurements (accuracy and AUC values) for the different models started to be notable when using a grid size $d = 10$. The best performance was obtained using LDM + composite LGM model.

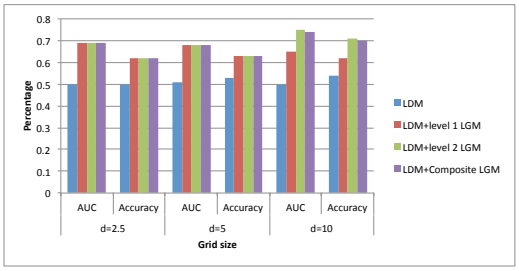
As a result of the above observations, it was concluded that LDM + composite LGM, coupled with $d = 10$, produced the most effective results.

5.6.5 Training and testing the classifier on a different data sets

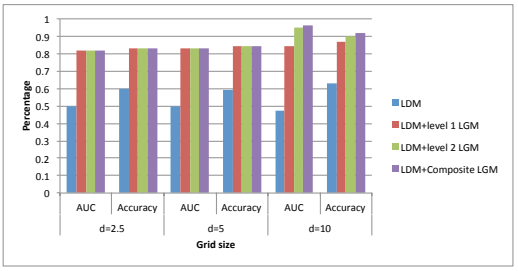
The objective of the experiments described in this sub-section, similar to those described in the previous chapter, was to determine whether the proposed LDM representation model was sufficiently generic. In other words whether the model captured sufficient geometries to describe 3D surfaces for general application. The experiments were conducted by training a classifier on one dataset and applying it to another dataset. More specifically, two sets of experiments were conducted: one using only the LDM representation and the other using the LDM + composite LGM representation.

For the first set of experiments considered in this section, informed by the foregoing experiments (described above), the following LDM parameters were used: $d = 2.5$ and $|L| = 3$ and the C4.5 classification algorithm. The obtained results are presented in Figure 5.11 in terms of accuracy and AUC values. From the figure, it can be seen that the best accuracy was 0.60 obtained when the classifier was trained on GTV2 and tested on GSV2. The best AUC value was 0.61 obtained when the classifier was trained on GTV1 and tested on GSV2. From the results, although not particularly good (many of the recorded AUC values are below 0.50), it can be seen that there is no significant variation when a classifier is trained on one shape and applied to another. From which we can conclude that the LDM representation on its own is a generic representation.

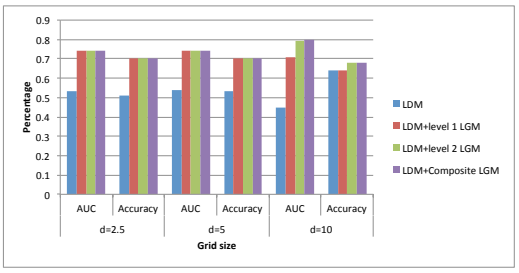
For the second set of experiments, $L = 3$, C4.5 and $d = 10$ was used. The results are presented in Figure 5.12 in terms of accuracy and AUC values. From the figures it can be seen that:



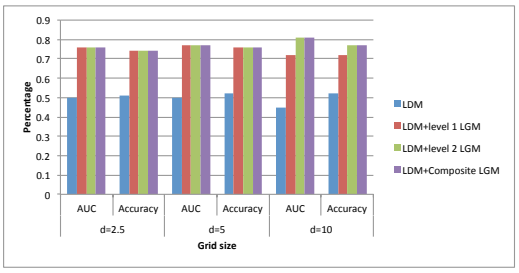
(a) $GSV1$



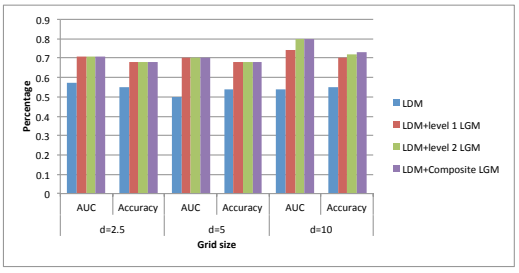
(b) $GSV2$



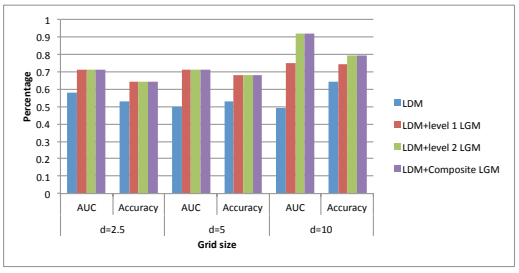
(c) $GTV1$



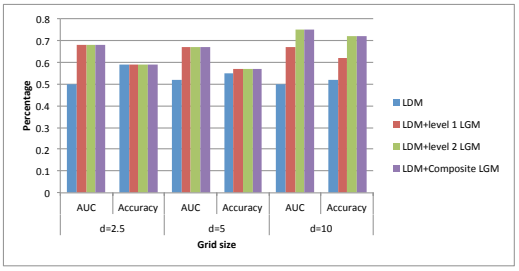
(d) $GTV2$



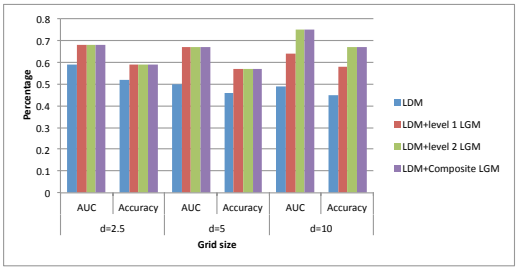
(e) $MSV1$



(f) $MSV2$



(g) $MTV1$



(h) $MTV2$

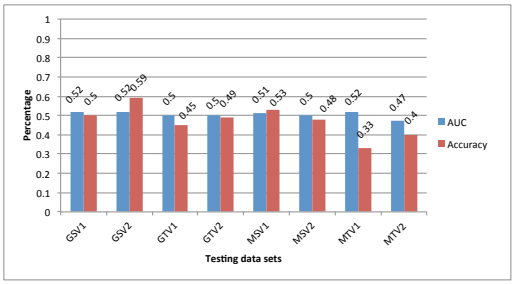
FIGURE 5.10: AUC and Accuracy results to identify the best LDM model using $|L| = 3$ and a range of grid size $\{2.5, 5, 10\}$ with respect to the eight evaluation datasets (using the C4.5 classification algorithm).

- The best AUC was 0.97 obtained when the classifier was trained on GTV2 and tested on GSV2. The best accuracy result was 0.92 obtained again when the classifier was trained on GTV2 and tested on GSV2.
- The main thing that can be observed from Figure 5.12 is that the results are significantly better than those presented in Section 5.6.4.
- For each case presented in Figure 5.12, a classifier was generated using one data set and tested on another, the results indicated that there was no significant difference in operation between when the classifier in question was applied to the same data set ("within data set" testing) or another data set (in "between data set" testing).
- There was no significant distinction between the classification results obtained with respect to the manufacturing material used (steel or titanium). It was thus concluded that to generate a generic classifier the generation process requires training provided data that maximises the number of different potential geometries rather than the number of material types available.

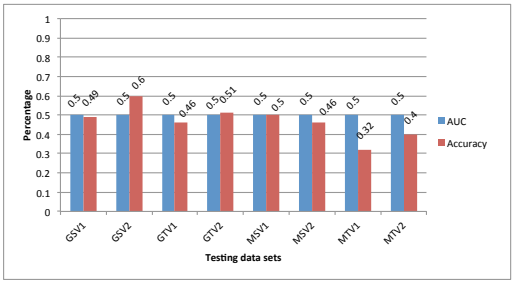
From the foregoing it was therefore possible to assert that the LDM + composite LGM model was sufficiently generic.

5.7 Run Time Analysis

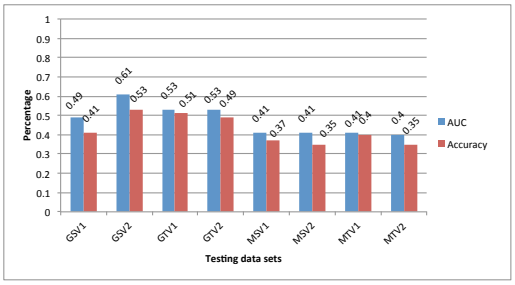
The section provides a run time analysis for both the LDM and LDM + composite LGM model for a range of d values using the C4.5 classification algorithm. The label size considered was $|L| = 3$. The run time analysis was conducted with respect to all eight evaluation data sets (GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1, MTV2). All the experiments were performed with 2.7 GHz Intel Core i5 PC with 4 GB 1333 MHz DDR3 memory, running OS X 10.8.1 (12B19). The LDM technique was designed and implemented using the Java programming language for both phases: (i) the detection for the critical points (corners and edges) and (ii) calculation of the distance between each point and the closest edge or corner using the region growing algorithm. The Weka version 3-6-8 implementation for C4.5 was used. Figures 5.13 and Figures 5.14 present the run time results obtained using LDM and LDM + composite LGM model respectively with respect to a range of grid sizes $d = \{2.5, 5, 10, 15, 20\}$. The run time presented in the figures was measured to the nearest second. From the figures it can be observed that the time required for detecting the critical points and calculating the nearest distance to the closest critical point decreased as the grid size increased. This is to be expected since the number of centre grid points decreases as the grid size increases. The run time for both models were very similar although the LDM + composite LGM required more time (by a very small margin) than the LDM model. Note that, as in the case of the run time experiments reported in Chapter 4, the run time for the classification phase was not included as this was negligible.



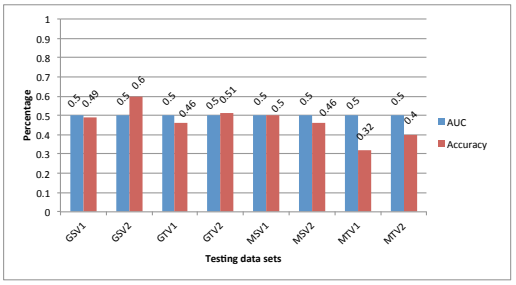
(a) *Training the generic classifier on GSV*1



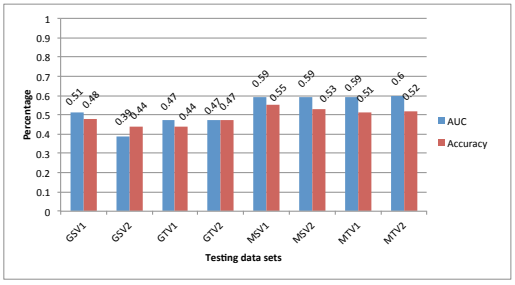
(b) *Training the generic classifier on GSV*2



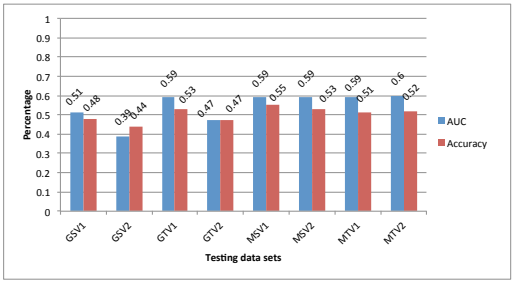
(c) *Training the generic classifier on GTV*1



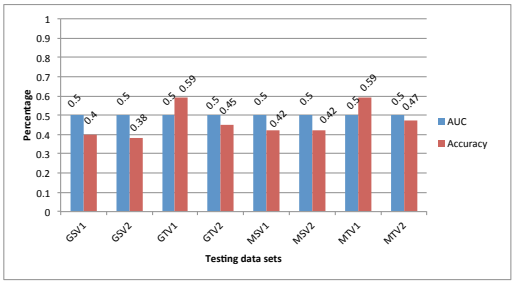
(d) *Training the generic classifier on GTV*2



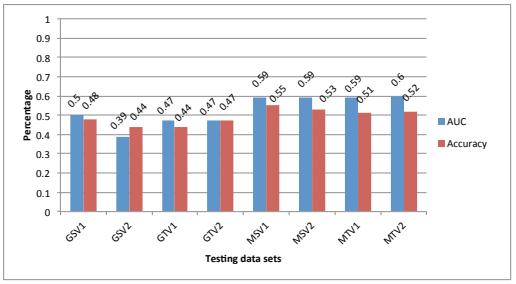
(e) *Training the generic classifier on MSV*1



(f) *Training the generic classifier on MSV*2

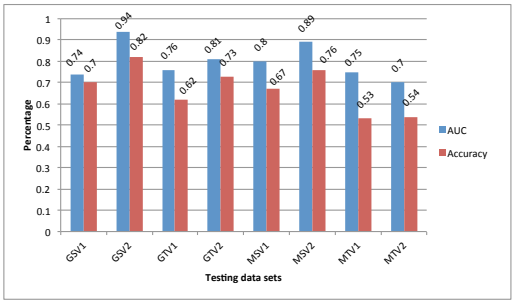


(g) *Training the generic classifier on MTV*1

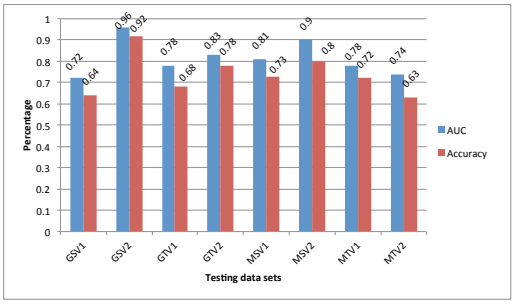


(h) *Training the generic classifier on MTV*2

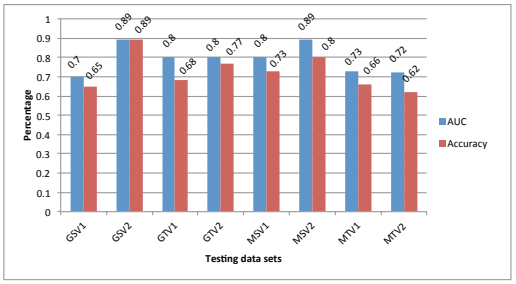
FIGURE 5.11: The AUC and Accuracy results produced when generating a classifier on one data set and applying it to another using the LDM model ($|L| = 3$, $d = 2.5$ and the C4.5 classification algorithm).



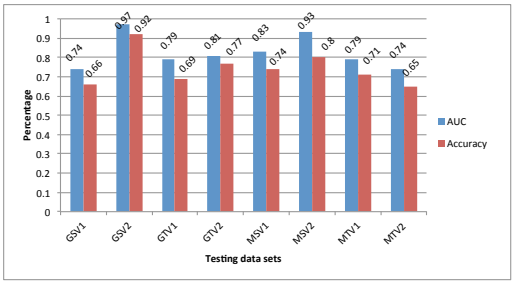
(a) *Training the generic classifier on GSV*1



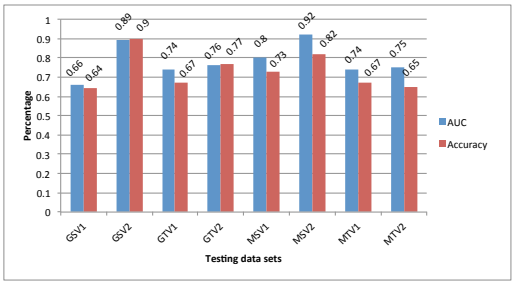
(b) *Training the generic classifier on GSV*2



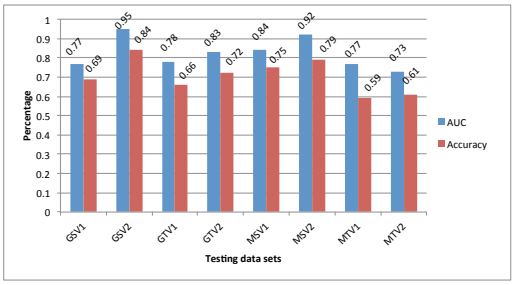
(c) *Training the generic classifier on GTV*1



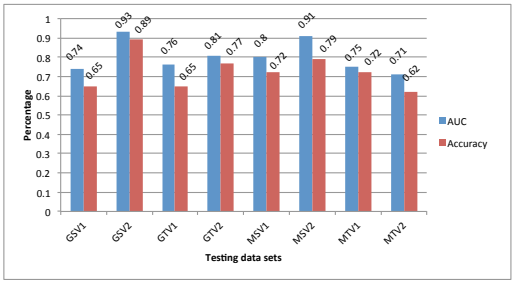
(d) *Training the generic classifier on GTV*2



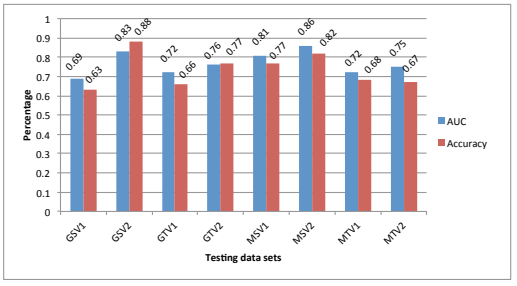
(e) *Training the generic classifier on MSV*1



(f) *Training the generic classifier on MSV*2

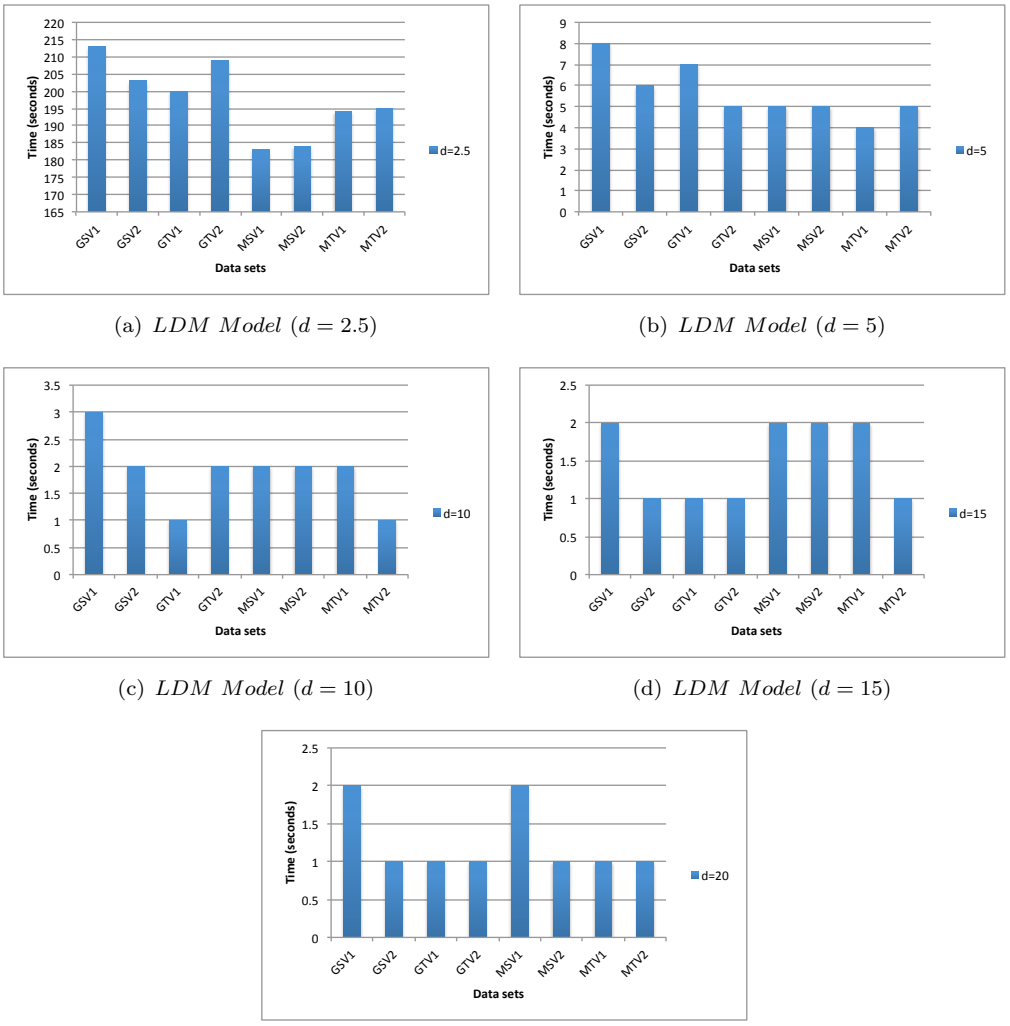


(g) *Training the generic classifier on MTV*1



(h) *Training the generic classifier on MTV*2

FIGURE 5.12: The AUC and Accuracy results produced when generating a classifier on one data set and applying it to another using the LDM+ composite LGM model ($|L| = 3$, $d = 10$ and the C4.5 classification algorithm).



(a) *LDM Model* ($d = 2.5$)

(b) *LDM Model* ($d = 5$)

(c) *LDM Model* ($d = 10$)

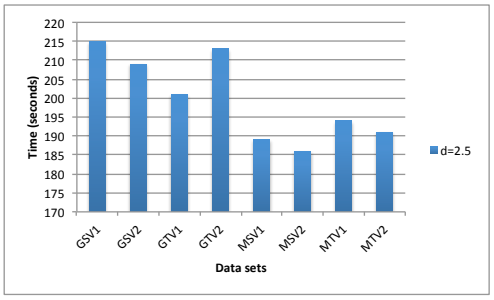
(d) *LDM Model* ($d = 15$)

(e) *LDM Model* ($d = 20$)

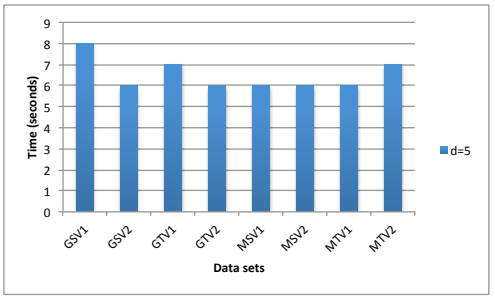
FIGURE 5.13: Run time for LDM model using $d = \{2.5, 5, 10, 15, 20\}$ with respect to the eight data sets GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2.

5.8 Summary

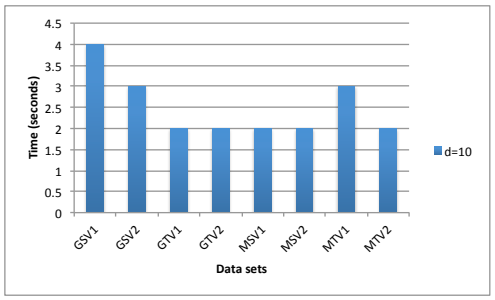
This chapter has proposed a new 3D representation technique called the *Local Distance Measure* (LDM) technique. The technique was founded on the concept of representing 3D surfaces in terms of the distance each grid point is from a critical point (a point representing an edge or corner). The local distance measure for a given grid point is the distance to the closest critical point. The LDM technique commences with the identification of the critical points in a given G_{in} and then, for each grid centre point, calculating the closest distance using a region growing technique. The operation of the proposed LDM mechanism was also considered in combination with the LGM technique described in the previous chapter: (i) LDM + Level 1 LGM, (ii) LDM + Level 2 LGM and (iii) LDM + Composite LGM. The evaluation of the proposed LDM representation resulted in the following main findings:



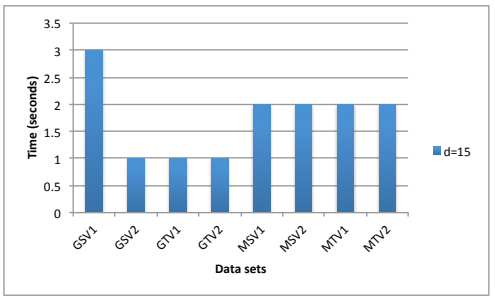
(a) *LDM + Composite LGM Model* ($d = 2.5$)



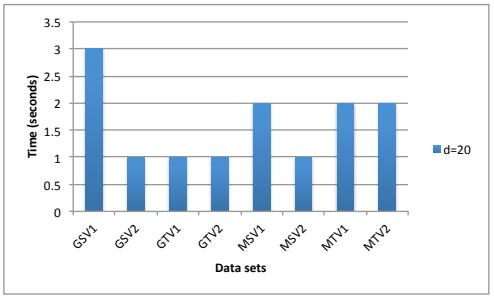
(b) *LDM + Composite LGM Model* ($d = 5$)



(c) *LDM + Composite LGM Model* ($d = 10$)



(d) *LDM + Composite LGM Model* ($d = 15$)



(e) *LDM + Composite LGM Model* ($d = 20$)

FIGURE 5.14: Run time for LDM + composite LGM using $d = \{2.5, 5, 10, 15, 20\}$.

1. The grid size $d = 2.5$ was founded to be the most appropriate grid size that can best be used to define geometrical information with respect to the LDM model on its own because the detection of critical points is more effective using smaller grid sizes.

2. The label size that produced the best AUC and accuracy values was $|L| = 3$ with respect to LDM model (although if we wish to apply corrections for springback an alternative label size may be more desirable).

3. Usage of the LDM representation on its own did not prove effective, leading to the conclusion that the idea that springback is related entirely to distance from a closest edge (critical point) was not entirely well founded.

4. The LDM + composite LGM model was found to be the best overall model as it obtained the best overall performance results, in terms of accuracy and AUC values, indicating that the LDM representation does have a part to play.

5. A grid size $d = 10$ was founded to be the most appropriate with respect to the LDM + composite LGM model (confirming the result reported previously in Chapter 4).

6. A generic classifier can be generated using the LDM technique on its own, however a more effective generic classifier can be produced using the LDM + composite LGM model.

7. The LDM + composite LGM model produced the best generic springback prediction results.

8. A generic classifier can be generated regardless of the manufacturing material (steel or titanium).

In the following chapter, the third surface representation technique considered in this thesis is presented, the *Point Series* (PS) Representation.

Chapter 6

Point Series Representation

6.1 Introduction

This chapter presents the last proposed technique in this thesis with which to represent 3D surfaces in such a way that classification algorithms can be successfully applied. The representation is founded on the idea of using point series (*linearisations* of space) to capture the nature of the local geometries making up a given 3D surfaces. More specifically the representation uses a linearisation process to capture local geometries in the context of the neighbourhood of each grid centre point to form a collection of point series, the representation is thus referred to as the Point Series (PS) representation.

Each linearisation is conceptualised as a "curve" made up of a sequence of points. In the context of the training data required for supervised learning each curve is also associated with a springback value. The overall intuition is that a sufficiently large collection of point series curves will serve to encapsulate every possible local geometry each with an associated springback value. The collection of curves can thus be used to form a "bank of curves". Given a new "unseen" 3D surface the point series curves making up the geometry of this new surface can be "looked-up" in the bank and associated spring back values retrieved. We can conceive of the process as a form of case-based reasoning [1] where the bank represents the case base. Comparison between new curves in the "case base" can be conducted in a number of ways so as to retrieve the"best" match in each case. However in this thesis it is proposed that the well known k-Nearest Neighbour (k-NN) approach is adopted coupled with a Dynamic Time Warping (DTW) approach to determine the similarity between new curves and existing curves (recall that DTW was discussed in Chapter 2).

There are a number of "space filling curve" formats that could have been adopted (for example a Peano curve [166] or a Hilbert curve [102]) with respect to the desired linearisation. However, a straightforward spiral linearisation was chosen as this fits well with the requirement to capture the local geometries associated with each grid centre point p_i in terms of its $n \times n$ neighbours.

Extensive experiments were conducted to evaluate the PS technique. The experiments indicated that the PS technique outperforms the previously proposed LGM and

LDM techniques, in terms of AUC and accuracy. The PS technique is thus considered to be the "best" proposed 3D representation technique considered in this thesis.

The rest of this chapter is organised as follows. Section 6.2 presents the proposed PS technique. Section 6.3 describes the incorporation of the technique into the RASP framework. The evaluation of the proposed PS technique, with respect to prediction effectiveness, is presented in Section 6.4; while the evaluation of the technique, with respect to run time, is presented in Section 6.5. Finally, the chapter is concluded with a summary in Section 6.6.

6.2 Point Series Representation

In a similar manner to the LGM and LDM techniques presented earlier, the input to the PS technique is the grid representation previously described in detail in Chapter 3. Recall that the centre representative point p_i for each grid square is characterised by its z value and associated springback value, and that the differences in z-value (δz) between p_i and each of its neighbours can be used to reflect the local geometry surrounding p_i. Point series are generated as follows. For each centre point in the input grid G_{in} we consider a $n \times n$ neighbourhood centred on p_i. We then prescribe a point series (a curve) that passes through some or all of the points in the neighbourhood such that each point in the point series is described by a δz value. The resulting curves can be visualised by plotting them on a graph where the y-axis represents the δz value and the x-axis the point series ordering (each point can be allocated an ordinal number). Each curve is also associated with p_i springback (error) *label* or *value*.

When generating point series, using the process described above, there are two aspects to be considered:

1. The neighbourhood size and
2. The number of points to be included in a linearisation (*key* points only or *all* points).

The size of the neighbourhood is related to the grid size d in such away that a large value for d will induce larger grid squares and consequently the immediate neighbourhood surrounding p_i will cover a larger area. As noted above we define a neighbourhood in terms of a $n \times n$ block of grid squares where n is an odd number and $n \geq 3$ (so as to ensure a symmetric covering of a potential neighbourhood surrounding a given p_i). With respect to the number of points to be included in the PS representation, two options were considered, either we include *all* the points located in the $n \times n$ blocks (to give point series of length $n^2 - 1$), or we include only selected *key* points located in the corners and "mid-ways" of each block surrounding p_i (to give point series of length $8 \times (n-1)/2$). Intuitively the *key* PS representation would be more efficient from a run time perspective, than the *all* PS representation; especially if n is large. This process is illustrated in Figure 6.1 where a 5×5 *key* point grid is used (the point p_i, is located at

the centre and coloured in red). Figure 6.1(a) shows the neighbouring points of interest coloured in green (different configurations could clearly be used). Figure 6.1(b) shows the adopted spiral linearisation (points are numbered from 1 to 16). Figure 6.1(c) shows a plot of the resulting curve. In this manner a "bank" of curves can be created where each curve is associated with a predefined label (springback value).

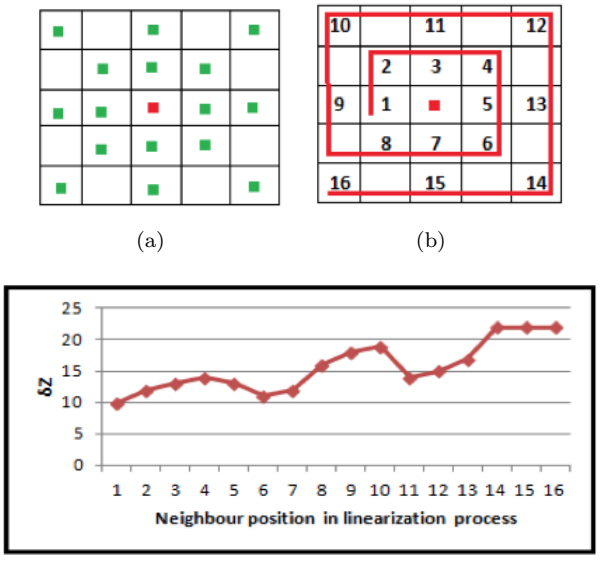


FIGURE 6.1: Example of a spiral linearisation for a 5×5 *key* PS representation.

6.3 The Prediction Framework Mechanism

This section demonstrates the operation of the prediction framework while using the proposed PS representation. As indicated previously in Chapter 3, the RASP framework comprises three stages. The first stage is the preprocessing stage where the grid representation and error (springback) calculations are performed for a given shape. The preprocessing stage results in the G_{in}. Recall that each grid centre point in G_{in} is associated with an error (springback) value. G_{in} is then used as the input to next stage, the surface representation stage, where any of the proposed 3D surface representation technique can be applied (in this case, the PS representation is used). The third stage is classifier generation and testing. In the case of the PS representation, and unlike the previous representations considered, the PS representation operates using the k-NN supervised learning mechanisms. More specifically k-NN classification is used, with k is set to 1. The classification process of a new curve, c_{new}, using k-NN classification is conducted according to a similarity measure. Different approaches have been proposed to achieve this. The most popular similarity measurement is Euclidean distance[1]:

[1]Other similarity measurements can be found in [16, 51, 212].

Given two sequences $A = [a_1, a_2, \ldots, a_n]$ and $B = [b_1, b_2, \ldots, b_n]$ of the same length $|A| = |B| = n$ then the similarity value ($S(A, B)$) between A and B is defined using the standard Euclidean distance measure ($D(A, B)$) as follows.

$$S(A, B) = D(A, B) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

However, with respect to the work described in this thesis, a distance was found to be insufficient with respect to the proposed PS representation. Note also that, although not applicable in the context of the PS representation, distance based similarity measures are not suited to sequences of different length [184, 224, 225]. Hence Dynamic Time Warping (DTW) was adopted (introduced previously in Chapter 2). The rest of this section is organised as follows. The calculation of the DTW measure is presented in Section 6.3.1 while Section 6.3.2 describes the operation of the k-NN classification.

6.3.1 Dynamic Time Warping Similarity Measurement

This section presents the Dynamic Time Warping (DTW) algorithm together with a detailed example. Recall that the background to DTW was described previously in Chapter 2. Algorithm 6.1 presents the DTW algorithm. It should be noted that Algorithm 6.1 makes use of the *Sakoe-Chiba* (S-C) Band [181], a "windowing" mechanism also described earlier in Chapter 2. The *Sakoe-Chiba* (S-C) band window was employed with respect to the work described in this thesis to: (i) eliminate the implications of the *singularity* problem, (ii) improve the complexity to $O(n \times w)$ [108, 147, 184] and (iii) maintain the desired optimal path near the diagonal. The value of $w = 10\%$ of the series length was used as suggested in [156] and as shown in Algorithm 6.1 (line 8).

The algorithm operates as follows. The input is a new curve to be labelled c_{new} and a set of labelled curves $C = \{c_1, c_2, \cdots, c_n\}$. Suppose that $c_{new} = [a_0, a_1, \cdots, a_p]$ and $c_l = [b_0, b_1, \cdots, b_q]$ where $c_l \in C$. The first step is to generate a 2D matrix M of size $A \times B$ (line 7). Each matrix entry $M(i, j)$ then holds the "cost" between the corresponding points a_i and b_i where the cost is defined in terms of the Euclidean distance between the two points $c(a_i, b_j) = D(a_i - b_i) = \sqrt{(a_i - b_i)^2} = |a_i - b_i|$.

Figure 6.2 illustrates the operation of DTW for a given c_{new} and c_l where a matrix M is generated to identify the DTW path and the DTW value. For simplicity, the points of the curve are all integer values such that $c_{new} = [1, 2, 4, 5, 7, 6, 6, 5, 8, 3, 4, 7]$ and $c_l = [2, 2, 3, 6, 7, 7, 8, 5, 4, 3, 6, 5]$. Referring to Algorithm 6.1, and the example in Figure 6.2, the operation of DTW can be described as follows.

1. The inputs are the new unlabelled curve $c_{new} = [a_0, a_1, \cdots, a_p]$ ($|c_{new}| = p+1 = A$) and the set of curves $C = \{c_1, c_2, \cdots, c_n\}$ where $c_l = [b_0, b_1, \cdots, b_q]$ ($|c_l| = B = q + 1$).

2. A matrix M of length $|A \times B|$ is established.

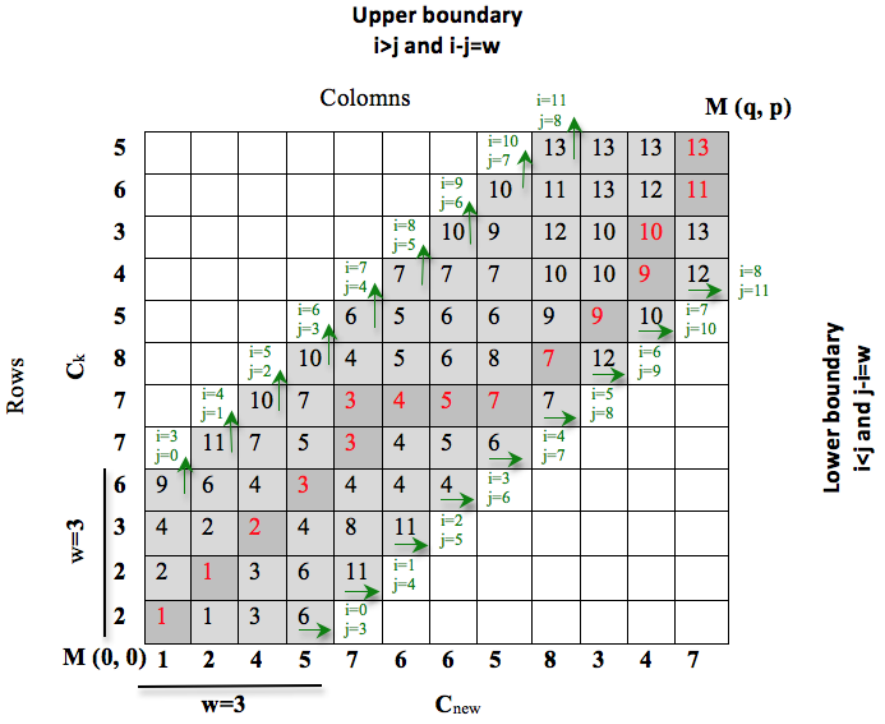


FIGURE 6.2: An example of the operation of DTW using two equal sized curves c_1 and c_2. For illustrative purposes a window size of $w = 3$ was used (as shown in shaded area). The indices of the lower and upper boundary are coloured in green. The optimal DTW path is indicated using dark shading with red text. The path commences at $M(0, 0)$ and ends at $M(11, 11)$. The DTW value is located in $M(11, 11)$ and is equivalent to 13 in this case.

3. The first element $M(0, 0)$ is calculated using the standard euclidean distance (line 9 in the algorithm). With reference to the example in Figure 6.2:

$$\begin{aligned} M(0,0) &= |1-2| \\ &= 1 \end{aligned}$$

4. The first w elements of the first row of M are calculated by adding the cost of the corresponding elements of c_l and c_{new} recursively to the cost of the previous element as shown in line 12. With reference to the example in Figure 6.2:

$$\begin{aligned} M(0,2) &= |4-2| + M(0,1) \\ &= 2+1 \\ &= 3 \end{aligned}$$

Algorithm 6.1: *Dynamic Time Warping (DTW)*

Input: New unlabelled curve (c_{new}), a set of curves ($\mathcal{C}$)
Output: The set of similarity values.

```
1  C = {c_1, c_2, ··· , c_n} ;
2  c_new ← [a_0, a_1, ··· , a_p] ;
3  for all c_l ∈ C do
4  |   c_l ← [b_0, b_1, ··· , b_q];
5  |   A ← p + 1;
6  |   B ← q + 1;
7  |   M ← new[A × B] ;                                  // Initialise M
8  |   w ← ⌈0.10 × A⌉ ;
9  |   M(0,0) ← |c_new(0) − c_l(0)| ;
10 |   i ← 1;
11 |   while i ≤ w do              // Calculate the 1st row of M (M(i,0))
12 |   |   M(i,0) ← |c_new[i] − c_l(0)| + M(i − 1, 0) ;
13 |   |   i + +;
14 |   end
15 |   j ← 1 ;
16 |   while j ≤ w do           // Calculate the 1st column of M (M(0,j))
17 |   |   M(0,j) ← |c_new(0) − c_l(j)| + M(0, j − 1) ;
18 |   |   j + +;
19 |   end
20 |   row ← A;
21 |   column ← B;
22 |   for i ← 1 to row do                       // Calculate the rest of M
23 |   |   for j ← 1 to column do
24 |   |   |   if |i − j| ≤ w then
25 |   |   |   |   M(i,j) ← |c_new(i) − c_l(j)|+Min(M, i, j) ;   // Algorithm 6.2
26 |   |   |   end
27 |   |   end
28 |   end
29 |   c_l.dtw ← M(A, B) ;          // The dtw is the similarity value for c_l
30 end
31 return C′ ;                          // C associated with similarity values
```

5. The same calculations described in step 4 are performed for the first w elements of the first column of M as shown in line 17. With reference to the example in Figure 6.2:

$$\begin{aligned} M(3,0) &= |6-1| + M(2,0) \\ &= 5 + 4 \\ &= 9 \end{aligned}$$

6. The rest of the elements of M are calculated as follows. For $M(i,j)$ the distance between the corresponding points along both curves c_{new} and c_l is calculated and then the value is added to the *minimum* cost of the adjacent cells $M(i-1,j)$, $M(i,j-1)$ and $M(i-1,j-1)$. Algorithm 6.2 is used to identify the minimum cost within the S-C band (as shown in line 25 of Algorithm 6.1). The value of each element in M is used to calculate the minimal warping path. To give an example and with reference to the example in Figure 6.2:

$$\begin{aligned} M(8,7) &= |4-5| + Min(M,8,7) \\ &= 1 + minimum\ value\ of\ \{M(7,7), M(7,6), M(8,6)\} \\ &= 1 + M(7,7) \\ &= 1 + 6 \\ &= 7 \end{aligned}$$

Note that Algorithm 6.2 distinguishes the position of the elements, whether they are located on the upper boundary or the lower boundary based on the i and j values (indices are coloured in green in Figure 6.2). If the element is located on the lower boundary ($i < j$) (as shown in line 2 of Algorithm 6.2) then the minimum value would be selected from two elements $M(i,j-1)$ and $M(i-1,j-1)$ as $M(i-1,j)$ is out of the S-C band. With reference to the example in Figure 6.2:

$$\begin{aligned} M(6,9) &= |8-3| + Min(M,6,9) \\ &= 5 + minimum\ value\ of\ \{M(6,8), M(5,8)\} \\ &= 5 + M(6,8) \\ &= 5 + 7 \\ &= 12 \end{aligned}$$

However, if the element is located on the upper boundary ($i > j$) (line 8 of Algorithm 6.2) then the minimum value would be selected from two elements $M(i-1,j)$

and $M(i-1, j-1)$ as $M(i, j-1)$ is outside of the S-C band. For example:

$$\begin{aligned} M(7,4) &= |5-7| + Min(M,7,3) \\ &= 1 + minimum\ value\ of\ \{M(6,3), M(6,4)\} \\ &= 2 + M(6,4) \\ &= 2+4 \\ &= 6 \end{aligned}$$

7. When the matrix M has finally been generated, the minimal DTW path, and associated similarity value can be identified. In our example, the dark shaded area, with red text, indicates the DTW path (the warping path) and the similarity value of 13 is located in $M(11, 11)$.

Algorithm 6.2: Min Algorithm

Input: Matrix M, i, j
Output: *min value*

```
1  w ← ⌈0.10 × M.length⌉;
2  if (|i − j| = w) & (i < j) then          // Elements on the lower boundary
3  |  if M(i, j − 1) < M(i − 1, j − 1) then
4  |  |  min ← M(i, j − 1)
5  |  else
6  |  |  min ← M(i − 1, j − 1)
7  |  end
8  else if (|i − j| = w) & (i > j) then     // Elements on the upper boundary
9  |  if M(i − 1, j) < M(i − 1, j − 1) then
10 |  |  min ← M(i − 1, j)
11 |  else
12 |  |  min ← M(i − 1, j − 1)
13 |  end
14 else                                     // Elements within boundaries
15 |  min ← M(i − 1, j) ;
16 |  if min > M(i − 1, j − 1) then
17 |  |  min ← M(i − 1, j − 1) ;
18 |  end
19 |  if min > M(i, j − 1) then
20 |  |  min ← M(i, j − 1) ;
21 |  end
22 end
23 return min ;
```

6.3.2 *k*-NN Classification

This section presents the operation of the k-NN classification algorithm with $k = 1$ as adopted with respect to PS representation technique described in this thesis. The combination of *1*-NN with the DTW similarity measure has been shown to outperform other techniques with respect to time series classification [52, 217]. The curve c_l with the minimum DTW would be the most similar curve to c_{new}. If there is a clear "winner" the label from this winner is used to label c_{new}. However, if more than one c_l has the same lowest DTW value then there are two alternatives to define the label of c_{new}, either: (i) some kind of "voting scheme" may be used where the majority label is considered to be the winner label, or (ii) the average value for the labels may be calculated and used to define the label for c_{new}. The first is applicable to the *discretised* PS representation while the second is applicable to the *real* PS representation.

Once the DTW value has been calculated, all $c_l \in C$ are sorted in ascending order according to their DTW values as shown in Algorithm 6.3 (line 1). Then S is initialised as the set of similarity values (line 2 of Algorithm 6.3). The first element of S is assigned to the lowest DTW value as shown in line 3 of Algorithm 6.3. Thus, if $S = \{c_l\}$, i.e. S contains just one curve c_i, the c_{new} is given the label of c_i (line 10 of Algorithm 6.3). Otherwise, all repeated DTW values will be located in S (line 6 of Algorithm 6.3) until the first different DTW value arise. The label of c_{new} will be the average of all c_l labels with the same DTW value located in S as shown in line 16 of Algorithm 6.3.

Algorithm 6.3: *k-NN*

Input: *new unlabelled* c_{new}, set of curves $\mathcal{C}$
Output: *labelled* c_{new}

```
1  C'' ← sorted C' according to DTW values in ascending order ;
2  Initialise S[ ] ← φ ;                    // Array of the similarity values
3  S[0] ← C'[0].dtw ;
4  i ← 1;
5  while C''.dtw = S[0] and i ≤ |C''| do
6  |  S[i] ← C''[i].dtw ;
7  |  i++;
8  end
9  if |S| == 1 then
10 |  c_new.label ← C''[0].label ;
11 else
12 |  sum ← C''[0].label;
13 |  for i ← 1 to |S| do
14 |  |  sum← sum+C''[i].label;
15 |  end
16 |  c_new.label ← sum/|S| ;
17 end
18 return labelled c_new ;
```

6.4 Evaluation

To evaluate the operation of the PS representation, a variety of experiments were conducted using the eight Gonzalo and Modified pyramid data sets: GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2. As in the case of the evaluations presented in Chapters 4 and 5 all the results were evaluated in terms of accuracy and AUC. TCV was used throughout. The objectives of the evaluation were as follows:

1. To compare the effect of using *discretised* or *real* springback error values for the PS representation.
2. To determine the effect of the grid size d used.
3. To analyse the nature of the neighbourhood size n with respect to prediction effectiveness.
4. To identify the most appropriate type of linearisation, *all* points or *key* points, to be adopted.
5. To investigate the generalisation of the proposed PS representation technique.

In the context of the first objective (comparison of the use of discretised versus real springback values) it should be noted that the k-NN classification technique supports the option of real valued springback error prediction, while this was not the case with the classification techniques applicable with respect to the previously considered LGM and LDM representations. Note that the reader should be clear that real values are always used for the δz values used to define the point series. With respect to the springback discretisation, the springback values were ranged and allocated labels from a set of labels L. A sequence of different values for $|L|$ were considered. In the case of the real value approach, to compare the predicted springback values with the real known values a *tolerance* of 0.08 mm was used as suggested by BS EN ISO 1101:2005 [29][1].

To evaluate the effect of different grid sizes, a range of grid sizes $d = \{2.5, 5, 10, 15, 20\}$ was considered. The experiments to determine the effect of grid size were coupled with experiments to determine the effect of the nature of the neighbourhood size n by also considering three different possible values for n ($\{3, 5, 7\}$).

The investigation of the generalisation of the PS representation was conducted so as to provide an answer to the question: can we generate a sufficiently generic bank of curves using a given shape in order to predict the springback values for a new shape. Similar experiments were conducted with respect to the two other proposed representations as reported in earlier chapters.

The evaluation results, in the context of each of the above objectives, are reported and discussed in the following five sub-sections (one per evaluation objective).

[1] BS EN ISO 1101 is a Geometrical Product Specification (GPS) standards. Tolerance in this context is the maximum acceptable variation between the technical design and actual (true) manufactured geometry.

6.4.1 Discretised vs Real Point Series Representation

This section presents and discusses the results obtained from the experiments conducted to compare the effect of using *discretised* springback values versus *real* values. Hence two groups of experiments were recorded: (i) *discretised* PS and (i) *real* PS representation.

For the first group, *discretised* PS, a neighbourhood size of five ($n = 5$) was chosen because this is the mid way value in the range of n values considered and as a result of some provisional experimentation. Moreover, only the *key* points representation were selected so as to limit the number of experiments undertaken and because there was no expectation, in the context of discretisation versus real point comparison, that there would be any difference in operation between the *key* point PS and the *all* point PS representation. The error value was discretised using a range of labels $|L| = \{3, 5, 7, 9, 11, 13\}$. A range of grid sizes $d = \{2.5, 5, 10, 15, 20\}$ mm was used. The "voting scheme" was adopted to address the situation where there was more than one best minimum warping path DTW value. The results obtained using the Gonzalo and Modified pyramids are presented in Tables 6.1 and 6.2 respectively. The best accuracy and AUC results obtained for a range of grid sizes with respect to each label are highlighted in bold font. From the tables, it can be seen that:

1. Most of the best accuracy and AUC results were obtained when $|L| = 3$ such as GSV1 (using $d = 20$) where the accuracy and the AUC were 0.98 and 0.96 respectively (more examples can be detected from the tables).

2. Most of the best accuracy and AUC results were obtained when $d = 20$ such as MSV1 (using $|L| = 3$) where the accuracy and the AUC were 0.87 and 0.60 respectively (more examples can be detected from the tables).

3. Although the best overall accuracy and AUC result was 0.98 and 0.96 respectively, obtained by GSV2 when $|L| = 3$ and $d = 20$, most of the AUC results were much below 0.50 (worse than a guess) and hence this indicated that the discretised PS representation was not effective in many cases.

4. It can be observed that as the label size increased, the accuracy increased and the AUC decreased and this indicates that the majority of records were falling into one class.

From the above it can be concluded that the discretisation of springback values is inappropriate in the context of the PS representation. Closer investigation indicated that this was due to the *skewed* distribution of the error labels[1] whereby the majority of the k nearest neighbours of a given new case would always belong to the most frequent error label. Because the "voting scheme" used the *most frequent* label wins (although in

[1]It should be noted that in the case of the *equal* frequency discretisation, similar values are placed and grouped together under the same label with respect to each attribute and this would therefore result in labels with an unequal distribution. However, this observation indicates that the type of discretisation (equal *width* or equal *frequency*) does not have a significant implication (influence) on the different proposed 3D surface representation techniques.

many cases this was found not to be the correct one). An alternative might have been to consider other voting schemes, for example weighted schemes however this was not undertaken because the real value clearly outperformed the discretised approach.

In the second group, experiments were conducted using a range of values for the neighbourhood size: $n = \{3, 5, 7\}$ and the grid size $d = \{2.5, 5, 10, 15, 20\}$ mm along with a *tolerance* of 0.08 where the label is considered as *correctly predicted* as long as the difference between the actual and predicted error is below or equal the tolerance value. The proportion of the correctly predicted curves to the total number of curves was used to identify the accuracy measurement. However, for the AUC measurement the error values were divided into a collection of sub ranges where each sub range was evaluated against others in the same way as for binary valued classifiers (a detailed example on the AUC calculation is presented in Appendix C). The results are presented in Tables 6.3 and 6.4 in terms of accuracy and AUC respectively. Both tables indicate that impressive results were achieved; where the best accuracy and AUC values of 1.00 were obtained (when the GTV1 dataset was used in the prediction framework). The "best" overall average values for the accuracy and AUC were 0.98 and 0.96 respectively. These were excellent results.

6.4.2 The Effect of Grid Size

This section presents the conducted experiments in the context of identifying the most appropriate value for d (the grid size). The accuracy and AUC results obtained using a range of grid sizes, *real* error (springback) values and the *key* point PS representation were presented in Table 6.3 and 6.4 respectively. From the tables it can be observed that the PS representation was able to effectively represent 3D surfaces using different grid sizes. The best accuracy and AUC results of 1.00 were obtained using GTV1 and the 3×3 *key* point PS representation with grid size of $d = 5$. With respect to the overall average values obtained, the best overall average accuracy of 0.98 was obtained using grid sizes of $d = 2.5, 5$ and 10 and the best overall average AUC of 0.96 was obtained using a grid size of $d = 5$. Tables 6.5 and 6.6 present a summary of the number of occasions that a "best" accuracy and AUC result respectively were obtained for the range of values for d used ($\{2.5, 5, 10, 15, 20\}$) and with respect to the three different values of n considered ($\{3, 5, 7\}$). Note that each row adds up to eight because eight data sets were used for the experiments. From the table, it can be seen that: (i) the majority of best accuracy results were obtained using $d = 2.5$ while (ii) the majority of best AUC results were obtained using $d = 5$. Thus, $d = 5$ was found to be the best grid size for the 3×3 and 7×7 *key* point PS technique, but the AUC results obtained using the 5×5 *key* point PS were not conclusive with respect to a particular value of d as can be seen from Table 6.6. However, an argument can be made that $d = 5$ tends to produce more "best" results than the other grid sizes (total number of best occurrences of 11). Therefore, $d = 5$ was considered to be the most "appropriate" grid size for the PS representation.

TABLE 6.1: Accracy and AUC results using discretised error (springback) labels, the 5×5 *key* point PS technique, $d = \{2.5, 5, 10, 15, 20\}$ mm, TCV, and the Gonzalo pyramid datasets.

		d=2.5		d=5		d=10		d=15		d=20	
	$\|L\|$	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
GSV1	3	0.70	0.55	**0.77**	**0.68**	0.76	0.65	0.66	0.46	0.69	0.50
	5	**0.83**	0.43	0.82	**0.44**	0.80	0.41	0.73	0.25	**0.83**	0.41
	7	0.81	0.31	0.82	**0.44**	0.79	0.30	0.81	0.27	**0.83**	0.30
	9	0.85	0.28	0.85	**0.32**	0.83	0.24	0.85	0.25	**0.88**	0.27
	11	**0.87**	0.21	**0.87**	**0.27**	0.85	0.16	0.86	0.18	**0.87**	0.17
	13	0.88	0.20	**0.89**	0.24	0.87	0.16	**0.89**	**0.26**	0.88	0.16
GSV2	3	0.88	0.58	0.90	0.63	0.87	0.58	0.88	0.88	**0.98**	**0.96**
	5	0.81	0.34	0.81	0.38	0.81	0.39	0.89	**0.64**	**0.90**	0.63
	7	0.86	0.26	0.85	0.27	0.87	0.37	0.87	**0.53**	**0.88**	0.44
	9	0.86	0.21	0.86	0.23	0.87	**0.44**	0.83	0.33	**0.90**	0.41
	11	0.88	0.19	0.88	0.23	0.88	**0.39**	**0.92**	0.35	0.90	0.38
	13	0.89	0.16	**0.90**	0.24	**0.90**	0.23	0.89	0.33	**0.90**	**0.35**
GTV1	3	0.81	0.54	0.80	0.52	0.76	0.60	0.85	0.66	**0.88**	**0.68**
	5	0.76	0.30	0.75	0.31	0.78	0.33	**0.79**	**0.54**	0.78	0.49
	7	0.84	0.29	0.84	0.30	0.83	0.35	**0.86**	**0.49**	0.84	0.43
	9	0.84	0.21	0.84	0.25	0.85	0.26	**0.87**	**0.47**	**0.87**	0.39
	11	0.87	0.19	0.87	0.21	0.88	0.23	0.87	**0.41**	**0.89**	0.30
	13	0.88	0.15	0.88	0.19	0.89	0.18	0.88	0.33	**0.90**	**0.25**
GTV2	3	0.82	0.51	0.81	0.51	0.81	0.54	0.84	0.58	**0.89**	**0.76**
	5	0.78	0.33	0.80	0.50	0.78	0.36	0.81	0.44	**0.84**	**0.52**
	7	0.83	0.30	0.84	0.38	0.84	0.35	**0.88**	**0.48**	0.83	0.40
	9	0.84	0.21	0.85	**0.30**	**0.86**	0.26	**0.86**	**0.30**	**0.86**	**0.30**
	11	0.86	0.16	0.86	0.29	0.88	0.30	**0.89**	**0.41**	0.86	0.30
	13	0.88	0.16	0.88	0.27	0.89	0.20	**0.90**	**0.28**	0.89	0.22

TABLE 6.2: Accracy and AUC results using discretised error (springback) labels, the 5×5 *key* point PS technique, $d = \{2.5, 5, 10, 15, 20\}$ mm, TCV, and the Modified pyramid datasets.

		d=2.5		d=5		d=10		d=15		d=20	
	$\|L\|$	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
MSV1	3	0.77	0.56	0.76	0.56	0.70	0.43	0.68	0.43	**0.87**	**0.60**
	5	0.78	0.31	0.78	0.39	0.77	0.29	0.73	0.22	**0.84**	**0.41**
	7	0.81	0.27	0.79	0.22	0.81	0.28	0.80	0.19	**0.89**	**0.53**
	9	0.84	0.25	0.83	0.17	0.85	0.26	0.82	0.14	**0.86**	**0.27**
	11	0.86	0.22	0.86	0.17	0.86	0.26	0.86	0.26	**0.89**	**0.31**
	13	0.88	**0.21**	0.87	0.13	0.88	0.20	0.87	0.11	**0.90**	0.18
MSV2	3	0.78	0.62	0.80	0.66	0.88	**0.72**	0.89	0.67	**0.90**	0.63
	5	0.82	0.45	0.80	0.49	0.83	**0.57**	0.76	0.31	**0.90**	**0.57**
	7	0.83	0.35	0.83	**0.37**	**0.84**	0.36	0.77	0.25	0.82	0.21
	9	0.86	0.34	0.86	0.31	0.84	0.26	0.83	**0.34**	**0.87**	0.32
	11	0.87	**0.29**	0.87	0.25	0.87	0.20	0.86	0.26	**0.88**	0.20
	13	**0.89**	0.22	0.88	0.20	**0.89**	0.20	0.88	**0.25**	**0.89**	0.19
MTV1	3	0.77	0.61	0.78	**0.63**	**0.80**	**0.63**	0.72	0.56	0.63	0.46
	5	0.78	0.43	0.79	0.44	**0.83**	**0.61**	0.78	0.42	0.73	0.37
	7	0.83	0.39	0.83	0.38	**0.85**	**0.52**	0.82	0.34	0.83	0.34
	9	0.85	0.32	0.85	0.33	**0.87**	**0.42**	0.85	0.24	0.83	0.20
	11	**0.87**	0.25	**0.87**	0.26	0.88	**0.34**	**0.87**	0.23	0.86	0.18
	13	0.89	0.24	0.89	0.23	**0.90**	**0.35**	0.89	0.27	0.87	0.11
MTV2	3	0.70	0.58	0.70	0.54	**0.79**	**0.61**	0.63	0.42	0.71	0.49
	5	0.77	0.34	0.76	0.34	0.79	0.37	**0.80**	0.39	0.75	**0.40**
	7	0.81	0.31	0.81	0.29	**0.83**	**0.32**	0.81	0.29	0.82	0.29
	9	0.84	0.23	0.83	0.20	**0.86**	**0.30**	0.81	0.12	0.84	0.23
	11	0.86	0.18	0.85	0.17	**0.87**	0.24	0.86	0.16	0.86	0.20
	13	0.87	0.15	**0.88**	0.19	**0.88**	0.18	**0.88**	0.18	**0.88**	**0.21**

TABLE 6.3: The Accuracy results obtained using *real* error (springback) values, the *key* point PS representation with $n = \{3, 5, 7\}$, $d = \{2.5, 5, 10, 15, 20\}$ mm and TCV.

Data set	$n \times n$ *key* PS representation	Grid size (d) mm				
		2.5	5	10	15	20
GSV1	3×3 PS	0.97	**0.98**	**0.98**	0.97	0.90
	5×5 PS	0.97	0.97	0.98	**0.99**	0.84
	7×7 PS	0.98	**0.99**	0.94	0.87	0.78
GSV2	3×3 PS	**0.99**	0.98	0.97	0.96	0.96
	5×5 PS	**0.99**	0.97	0.96	0.94	0.92
	7×7 PS	**0.99**	0.98	0.94	0.89	0.67
GTV1	3×3 PS	0.99	**1.00**	0.99	0.94	0.93
	5×5 PS	**0.99**	**0.99**	0.94	0.94	0.91
	7×7 PS	0.98	**0.99**	0.87	0.79	0.81
GTV2	3×3 PS	**0.99**	**0.99**	**0.99**	0.98	0.96
	5×5 PS	0.99	**0.99**	0.96	0.95	**0.99**
	7×7 PS	**0.99**	0.98	0.88	0.83	0.48
MSV1	3×3 PS	0.97	0.97	**0.98**	0.97	0.97
	5×5 PS	0.96	0.96	0.96	**0.98**	0.91
	7×7 PS	0.96	0.97	**0.99**	0.89	0.75
MSV2	3×3 PS	0.96	**0.98**	**0.98**	0.97	0.92
	5×5 PS	0.96	0.96	**0.97**	**0.97**	0.92
	7×7 PS	**0.97**	**0.97**	0.96	0.92	0.67
MTV1	3×3 PS	**0.98**	**0.98**	**0.98**	0.97	0.90
	5×5 PS	**0.98**	**0.98**	0.96	0.97	0.84
	7×7 PS	**0.98**	**0.98**	0.92	0.97	0.70
MTV2	3×3 PS	0.97	0.96	0.93	**0.98**	0.82
	5×5 PS	0.97	0.96	0.93	0.96	**0.99**
	7×7 PS	**0.97**	0.96	**0.97**	0.92	0.56
Average	3×3 PS	**0.98**	**0.98**	**0.98**	0.97	0.92
	5×5 PS	**0.98**	**0.98**	0.96	0.96	0.92
	7×7 PS	**0.98**	**0.98**	0.93	0.89	0.68

6.4.3 The Effect of Neighbourhood Size

This section considers the effect of using different neighbourhood sizes (the size of the surrounding area, to be covered by a point series, with respect to p_i). Considering only the accuracy and AUC results obtained using a grid size of $d = 5$ as previously presented in Tables 6.3 and 6.4 and the summaries in Tables 6.5 and 6.6, it can be clearly seen that:

- The best accuracy and AUC of 1.00 was obtained using $n = 3$.
- The best overall average accuracy value was 0.98 obtained regardless of the values of n (see Table 6.3). However, the best overall average AUC value of 0.96 was obtained using $n = 3$ and $n = 7$ (see Table 6.4).
- Best AUC values tended to be produced when $n = 3$ (see Table 6.6). Thus it can be argued that the combination of the 3×3 *key* PS representation and $d = 5$ produced the most effective results.

Therefore, the 3×3 PS representation was selected to be the most "suitable" PS representation for further consideration.

TABLE 6.4: The AUC results obtained using *real* error (springback) values, the *key* point PS representation with $n = \{3, 5, 7\}$, $d = \{2.5, 5, 10, 15, 20\}$ mm and TCV.

Data set	$n \times n$ *key* PS representation	Grid size (d) mm				
		2.5	5	10	15	20
GSV1	3×3 PS	0.96	**0.97**	0.96	0.95	0.82
	5×5 PS	0.96	0.95	0.97	**0.99**	0.80
	7×7 PS	0.97	**0.98**	0.92	0.70	0.33
GSV2	3×3 PS	**0.95**	0.89	0.84	0.64	0.78
	5×5 PS	**0.94**	0.89	0.75	0.64	0.73
	7×7 PS	**0.96**	0.93	0.85	0.65	0.50
GTV1	3×3 PS	0.97	**1.00**	0.98	0.76	0.72
	5×5 PS	0.96	**0.99**	0.89	0.70	0.74
	7×7 PS	0.93	**0.99**	0.77	0.57	0.89
GTV2	3×3 PS	0.96	**0.99**	0.97	0.93	0.96
	5×5 PS	0.96	0.98	0.85	0.90	**0.99**
	7×7 PS	0.95	**0.96**	0.72	0.75	0.19
MSV1	3×3 PS	0.92	0.92	**0.97**	0.94	0.87
	5×5 PS	0.91	0.92	0.92	**0.97**	0.62
	7×7 PS	0.92	0.95	**0.99**	0.81	0.67
MSV2	3×3 PS	0.94	**0.97**	0.93	0.92	0.71
	5×5 PS	**0.94**	**0.94**	0.91	0.93	0.64
	7×7 PS	0.95	**0.97**	0.95	0.86	0.50
MTV1	3×3 PS	0.96	**0.97**	0.94	0.96	0.81
	5×5 PS	**0.96**	**0.96**	0.89	0.95	0.70
	7×7 PS	0.95	0.97	0.72	**1.00**	0.47
MTV2	3×3 PS	**0.96**	0.94	0.90	0.94	0.73
	5×5 PS	0.96	0.94	0.92	0.93	**0.98**
	7×7 PS	0.94	0.93	**0.95**	0.74	0.25
Average	3×3 PS	0.95	**0.96**	0.94	0.88	0.95
	5×5 PS	**0.95**	**0.95**	0.89	0.88	0.78
	7×7 PS	0.95	**0.96**	0.86	0.76	0.48

TABLE 6.5: Occurrences of the best accuracy results obtained using the 3×3, 5×5 and 7×7 *key* point PS representation, $d = \{2.5, 5, 10, 15, 20\}$ and *real* error (springback) values.

Neighbourhood size n	Grid size (d) mm					*Total*
	2.5	5	10	15	20	
3	1.66	**2.66**	**2.66**	1	0	8
5	2.33	1.33	0.5	**2.5**	1.33	8
7	**3.5**	3	1.5	0	0	8
Total	**7.49**	6.99	4.66	3.5	1.33	

TABLE 6.6: Occurrences of the best AUC results obtained using the 3×3, 5×5 and 7×7 *key* point PS representation, $d = \{2.5, 5, 10, 15, 20\}$ and *real* error (springback) values.

Neighbourhood size n	Grid size (d) mm					*Total*
	2.5	5	10	15	20	
3	2	**5**	1	0	0	8
5	**2**	**2**	0	**2**	**2**	8
7	1	**4**	2	1	0	8
Total	5	**11**	3	3	2	

6.4.4 The Nature of the Linearisation: *all* Points vs *key* Points Representation

This section considers the results obtained in the context of the comparison of the *all points* and *key points* variations of the PS representation. Recall that all the earlier experiments were conducted using the *key* points PS representation since the intuition was that using the *key* points would be more "efficient" than using the *all* points in the context of the PS representation because of it has less points and therefore it would require less memory storage and less processing time. It should be clear to the reader that a $n \times n$ block of points surrounding a given centre point p_i was used to describe the area covered by a linearisation and that $n \geq 3$. The *all points* PS representation included all the surrounding points covered by a block while the *key points* PS representation included only the points at the corners and "mid ways" of a block (the latter was illustrated in Figure 6.1(a)). Only the results produced using $d = 5$ are explicitly identified in this section (as it was considered earlier as the best grid size for the PS representation), however, it should be noted that similar results to those reported in this section were also obtained with respect to the other d values considered. Tables 6.7, 6.8 and 6.9 summarise the results using $n = 3$, $n = 5$ and $n = 7$ respectively (best values are in bold font). Note that there is no difference between the *all* points and the *key* points variations when $n = 3$ (all points are the key points). Inspection of the tables indicates that for $n = 5$ and $n = 7$, there is also no significant difference in operation between using either the *all* points or the *key* point variations with respect to accuracy and AUC.

6.4.5 Generalisation

This section considers the conducted evaluation in terms of the generalisation of the PS representation. Experiments were conducted whereby the "curve bank" was generated using one data set and tested using another. Note that not only the AUC and accuracy results of the *in between* data set testing are reported, but also the AUC and accuracy results obtained when the classifier was trained and tested using the same dataset (within data set testing). The $n = 3$ and $d = 5$ parameter settings were used with respect to

TABLE 6.7: The accuracy and AUC results when $n = 3$ (*key* vs *all* point variations).

Data sets	3×3 *key* PS representation ($d = 5$ mm)		3×3 *all* PS representation ($d = 5$ mm)	
	Accuracy	AUC	Accuracy	AUC
GSV1	0.98	0.97	0.98	0.97
GSV2	0.98	0.89	0.98	0.89
GTV1	**1.00**	**1.00**	**1.00**	**1.00**
GTV2	0.99	0.99	0.99	0.99
MSV1	0.97	0.92	0.97	0.92
MSV2	0.98	0.97	0.98	0.97
MTV1	0.98	0.97	0.98	0.97
MTV2	0.96	0.94	0.96	0.94

TABLE 6.8: The accuracy and AUC results when $n = 5$ (*key* vs *all* point variations).

Data sets	5×5 *key* PS representation ($d = 5$ mm)		5×5 *all* PS representation ($d = 5$ mm)	
	Accuracy	AUC	Accuracy	AUC
GSV1	0.97	0.95	0.97	0.95
GSV2	0.97	0.89	0.98	0.90
GTV1	**0.99**	**0.99**	**0.99**	**0.99**
GTV2	**0.99**	0.98	**0.99**	0.98
MSV1	0.96	0.92	0.96	0.93
MSV2	0.96	0.94	0.96	0.94
MTV1	0.98	0.96	0.98	0.95
MTV2	0.96	0.94	0.96	0.94

TABLE 6.9: The accuracy and AUC results when $n = 7$ (*key* vs *all* point variations).

Data sets	7×7 *key* PS representation ($d = 5$ mm)		7×7 *all* PS representation ($d = 5$ mm)	
	Accuracy	AUC	Accuracy	AUC
GSV1	**0.99**	0.98	0.98	**0.98**
GSV2	0.98	0.93	0.98	0.93
GTV1	**0.99**	**0.99**	**0.99**	**0.98**
GTV2	0.98	0.96	0.98	0.97
MSV1	0.97	0.95	0.97	0.95
MSV2	0.97	0.97	0.97	0.96
MTV1	0.98	0.97	0.98	0.97
MTV2	0.96	0.93	0.95	0.92

the *key* point PS representation, because earlier experiments had indicated that these values were the most effective. Figures 6.3(a) to 6.3(h) present the results obtained in terms of accuracy and AUC values. From the figures it can be observed that:

- A best accuracy value of 1.00 was obtained when GTV1 and MTV1 were used as the training sets, and GSV2 was used as the testing set.
- A best AUC value of 1.00 was obtained when curve banks were generated using the GSV2, GTV2, MSV1 and MTV1 data sets and tested using the MTV2, MSV2, GSV2 and GSV2 datasets.

- The AUC and accuracy results obtained for the classifiers generated using GTV1 and GTV2 data sets were better than the results obtained using the GSV1 and GSV2 data sets. Similarly, the AUC and accuracy results obtained using the classifiers generated using the MTV1 and MTV2 data sets were much better than the results obtained with respect to the classifiers generated using the MSV1 and MTV2 data sets. Thus, it can be concluded that a classifier generated using data sets describing objects manufactured from titanium (such as GTV1, GTV2, MTV1 and MTV2) produced more generically effective classifiers than than those generated using data sets describing objects manufactured from steel. Collings et. al in [42] showed that, based on the anisotropic properties of titanium, springback is considerably greater than for shapes (surfaces) manufactured from steel. Note also that the greater the variety of different geometrical patterns that can be provided when training the classifier the greater the generated classifier's ability to predict springback in the context of different shapes.

- For a classifier generated using one data set and tested on another, there is no significant difference in operation between testing the classifier on another data set (in between data set testing) or testing it on the same data set (within data set testing) which means that the proposed PS technique succeeded in capturing all possible geometries of a given shape effectively regardless of whether it has been applied on the same or another shape.

Overall these are excellent results. The results obtained in terms of both AUC and accuracy also indicated that, no matter what the nature of the 3D surface to be manufactured or the material from which it is to be manufactured, an effective generic prediction framework can be produced using the proposed PS Representation.

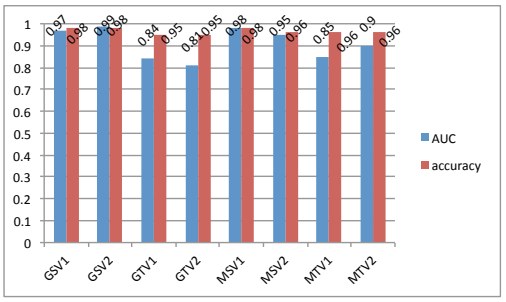
6.5 Run Time Evaluation

This section presents a run time analysis of the proposed PS representation. The run time analysis was conducted using two groups of experiments as follows.

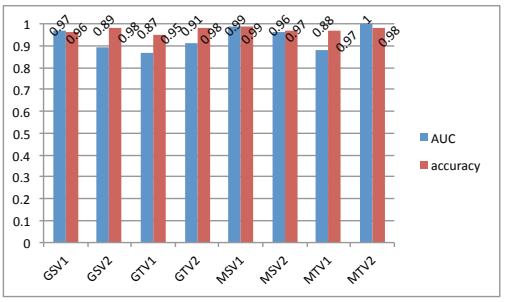
1. Using the *key* point and *all* point variations with $n = \{3, 5, 7\}$, $d = 5$ to clearly identify the effectiveness of the *key* point representation in terms of run time.

2. Using $n = 3$, as the 3×3 PS representation had been found to be the most appropriate variation of the PS representation, and a range of grid sizes $d = \{2.5, 5, 10, 15, 20\}$.

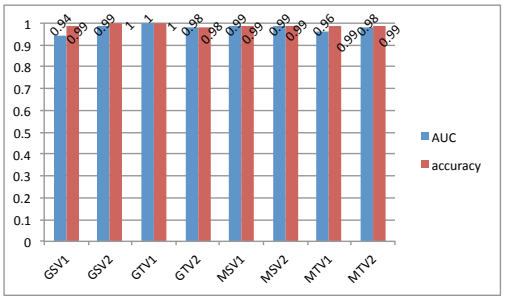
The experiments was conducted using all eight data sets, GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2; and a 2.7 GHz Intel Core i5 PC with 4 GB 1333 MHz DDR3 memory, running OS X 10.8.1 (12B19). The PS technique was implemented using the Java programming language and incorporated into the RASP framework. The run time includes all the preprocessing steps (grid representation and



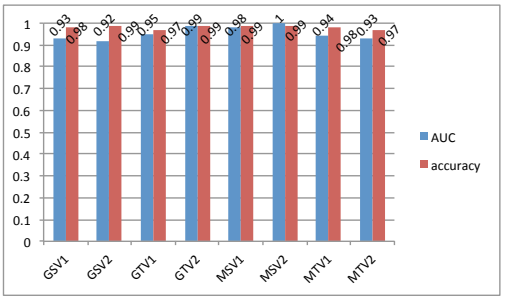
(a) *Training the generic classifier on GSV*1



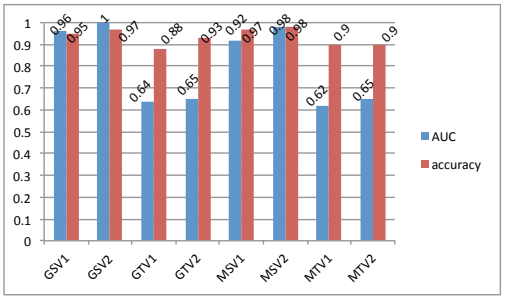
(b) *Training the generic classifier on GSV*2



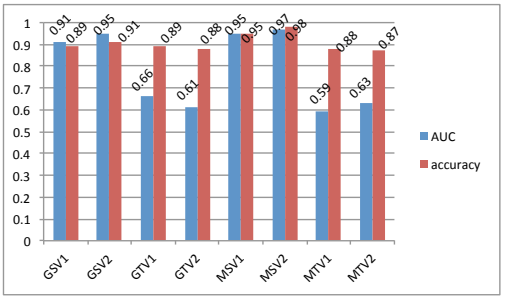
(c) *Training the generic classifier on GTV*1



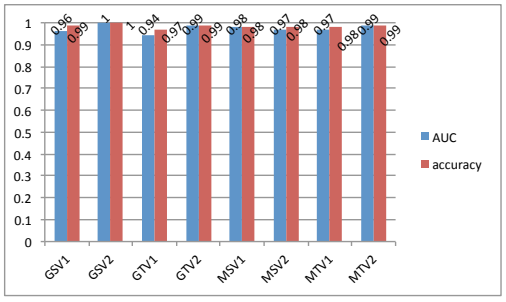
(d) *Training the generic classifier on GTV*2



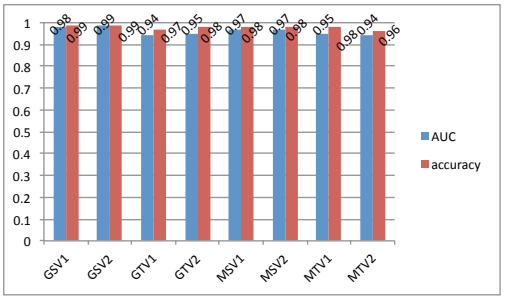
(e) *Training the generic classifier on MSV*1



(f) *Training the generic classifier on MSV*2



(g) *Training the generic classifier on MTV*1



(h) *Training the generic classifier on MTV*2

FIGURE 6.3: The AUC and Accuracy results produced when generating a classifier on one data set and applying it to another using $n = 3$, the *key* point PS representation and $d = 5$.

error calculation), the PS generation and k-NN coupled with similarity calculation. In addition, TCV was used. For the first group, Figures 6.4 to 6.6 show the recorded run times for $n = \{3, 5, 7\}$ in seconds. From the figures, it can be seen that the run times for the *key* point and *all* point variations for $n = 3$ were the same for reasons already noted. However, the *key* point variation outperformed the *all* point variation when $n > 3$ because fewer points need to be processed.

For the second group, 3×3 PS, Figures 6.7(a) to 6.7(e) shows the run time (seconds) recorded for the grid sizes of $d = \{2.5, 5, 10, 15, 20\}$ with respect to the eight data sets. The figures indicate that: (i) the recorded run time for the eight data sets are very similar for the same grid size, however smaller grid sizes requires more run time than larger grid sizes and this is clearly because using smaller grid sizes require more preprocessing and more PS curves to be generated.

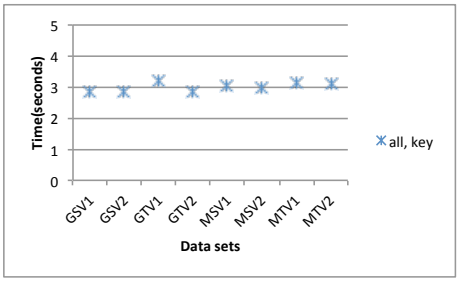


FIGURE 6.4: Recorded run time (s) for both the *all* point and the *key* point PS representation using $n = 3$ and $d = 5$.

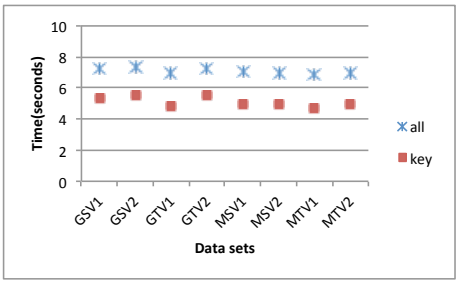


FIGURE 6.5: Recorded run time (s) for both the *all* point and the *key* point PS representation using $n = 5$ and $d = 5$.

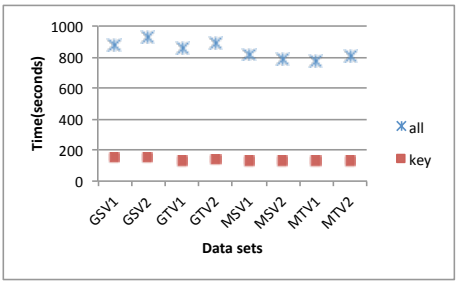
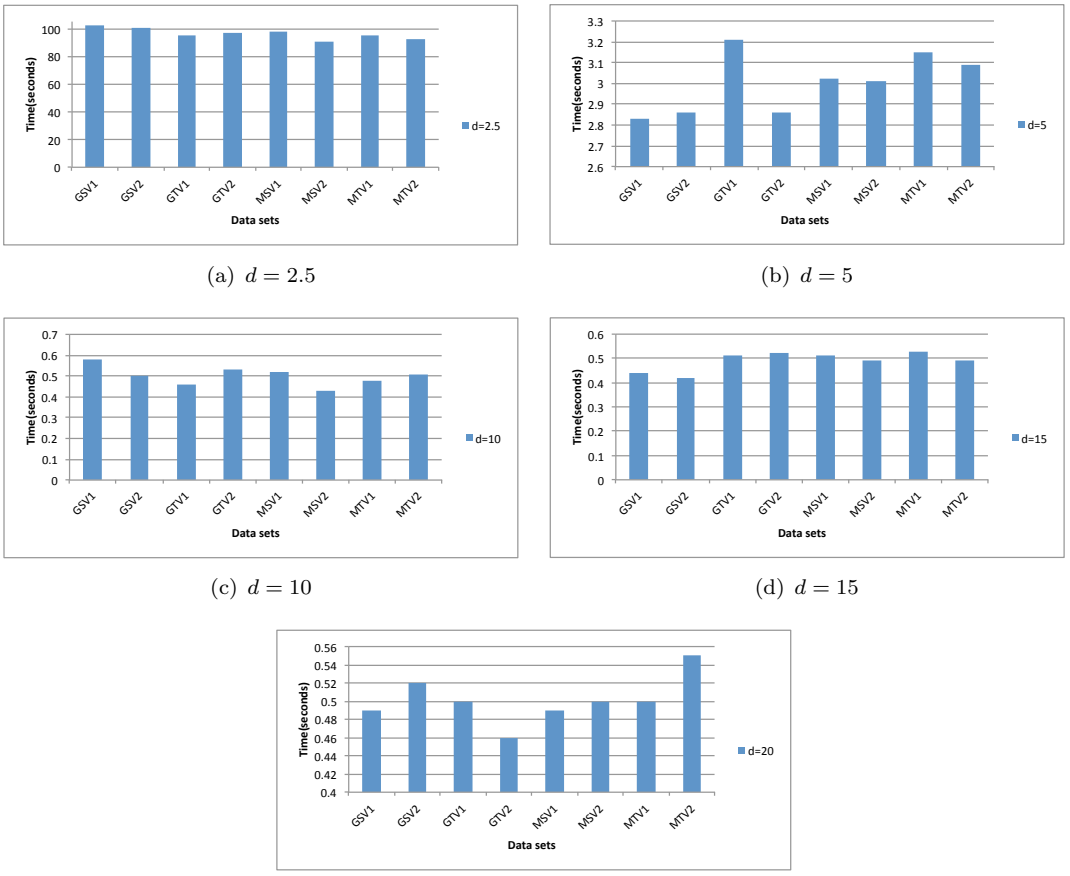


FIGURE 6.6: Recorded run time (s) for both the *all* point and the *key* point PS representation using $n = 7$ and $d = 5$.

6.6 Summary

The Point series (PS) 3D surface representation technique has been proposed in this chapter. The technique was founded on the concept of a linearisation of space whereby local geometric information was captured using a spiral linearisation. The proposed technique was incorporated into the RASP prediction framework using a k-NN classification mechanism which used the warping distance, calculated using Dynamic Time Warping (DTW), as the similarity measure. The "best" Warping Distance value was



(a) $d = 2.5$

(b) $d = 5$

(c) $d = 10$

(d) $d = 15$

(e) $d = 20$

FIGURE 6.7: Run time (in seconds) for 3×3 PS representation using $d = \{2.5, 5, 10, 15, 20\}$ with respect to the eight data sets GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2.

used to label new curves within a predefined tolerance value of 0.08 when springback was recorded in terms of real values (as opposed to discretised values). A range of related issues concerned with the PS representation were considered: (i) the most effective PS representation in terms of using either discretised or real springback values, (ii) the effect of using different grid sizes and consequently the most effective value for d, (iii) the effect of using different neighbourhood sizes n ($n = \{3, 5, 7\}$) and hence the most effective neighbourhood size, (iv) the most "appropriate" linearisation in terms of using *all points* or only *key points* and finally (v) the general applicability of the proposed representation. The main findings were as follows.

1. Using *real* error (springback) values was found to produce the most effective results in the context of the PS representation.

2. The most appropriate grid value was founded to be $d = 5$.

3. The reported results show that there was no significant difference between using different neighbourhood sizes ($n = \{3, 5, 7\}$) with respect to the recorded AUC

results. However, the best accuracy and AUC were obtained using $n = 3$, and thus $n = 3$ was identified as the most appropriate variation for the PS representation.

4. The run time experiments indicated that, as might be expected, larger neighbourhood sizes required more run time.

5. The *key* point PS representation was founded to be the most efficient PS representation in the context of run time.

6. An effective generic springback prediction model can be produced using the proposed PS 3D surface representation.

The overall results indicated that the PS technique outperformed the results obtained earlier using all other proposed surface representation techniques (LGM and LDM representation techniques). The following chapter presents a statistical study comparing all the proposed representation techniques in order to identify the significant difference between them.

Chapter 7

Statistical Comparison Between the Proposed 3D Representations

This section presents a statistical comparison of the three different 3D surface techniques proposed in this thesis. The objective is to demonstrate that the differences in performance of the proposed techniques are statistically significant. To this end, the Friedman statistical test was applied to evaluate the performance of the techniques to determine whether the results produced were truly significant or not with respect to the AUC measure. On completion of the Friedman test, the Nemenyi test was used to identify the "critical distances" between the techniques so as to identify exactly where the differences actually occurred. The statistical comparison was performed using the Gonzalo and Modified data sets used earlier. Section 7.1 presents an overview of a number of statistical performance evaluation methods and presents the reasons for choosing the Friedman statistical test with respect to the work presented in this thesis. Section 7.2 then describes the Friedman Statistical test and the associated Nemenyi test. Section 7.3 presents the results obtained using the Friedman and Nemenyi tests when the proposed techniques were trained and tested using the same data set, whilst Section 7.4 presents the best result obtained when the proposed techniques were trained on one data set and tested on the remaining seven data sets. The later test was concerned with demonstrating the generality of the proposed techniques. More considerations about the run time analysis of each proposed technique with respect to statistical analysis are presented in Section 7.5. Finally, this chapter is concluded in Section 7.6 with a brief review of the main findings of the chapter.

7.1 Overview of Statistical Performance Comparison

This section provides some background concerning the most common (popular) techniques used to perform statistical comparison of competing classification techniques. The statistical comparison between the operation of different classifiers has recently attracted more and more attentions from the data mining community. This kind of statistical analysis is increasingly being used [93**?**]. A number of different approaches

have been proposed to conduct such comparisons. Typically, these approaches can be categorised as being either: (i) *parametric* or (ii) *non-parametric*. The first is used when the distribution of the data set is drawn from a *normal* (*Gaussian*) distribution. The second makes no assumption about the distribution of the data set (Distribution-Free). The size of the data sets also has an impact on whether parametric or non-parametric analysis is conducted. For large data sets, both the parametric and the non parametric statistical tests have the same implication (in other word, there is no difference between the parametric and non-parametric tests for large data sets). Several forms of statistical test have been proposed with respect to both parametric and non-parametric testing. For a comparison between only two classifiers over different data sets, the *paired t-test* was proposed for parametric data while the *Wilcoxon Signed-Rank Test* has been proposed for non-parametric data. For more details and examples, see [50]. To identify the significant difference between more than two classifiers the Analysis Of Variance (ANOVA) and Friedman test have been extensively used [100, 186]. The ANOVA statistical test is based on two assumptions: (i) the normal distribution of the classification results and (ii) the data sets to have equal variance (the homogeneity of variance) [50]. Although both assumptions cannot be guaranteed, the violation of them would cause a greater effect on the post-hoc tests. Thus the ANOVA statistical test is not recommended for classification analysis unless both assumptions are certainly satisfied. However, the Friedman test is mainly directed at non-parametric testing. The Friedman test offers two advantages over parametric techniques (such as ANOVA): (i) ease of computation and interpretation and (ii) its ability to demonstrate the classification performance in terms of ranks rather than vague averages [82]. The Friedman test was thus chosen to evaluate the performance of the different proposed techniques with respect to this thesis. In addition to the practical advantages offered by the Friedman test, it was also chosen because [81, 82, 202]:

- There is no guarantee that the AUC results obtained from the proposed techniques follow the normal (Gaussian) distribution (the data is thus assumed to be non-parametric).
- The Friedman statistical test is generally recommended (see for example [50]) for use with related data sets while the ANOVA test is recommended for unrelated data sets. With respect to the work described in this thesis, the Gonzalo and Modified data sets were considered to be related data sets given that both describe flat-topped pyramids. Therefore, the Friedman test was considered to be the most suitable statistical test given our related data sets.

7.2 Friedman Statistical Test

This section describes the operation of the Friedman statistical test. The Friedman test is commenced by *ranking* each classification techniques, with respect to each of

the data sets, according to the recorded AUC values. Then, the average rank for each classification techniques was obtained from across data sets. The Friedman test statistic, χ_F^2, is then calculated as follows [50, 66, 76]:

$$\chi_F^2 = \frac{12n}{k(k+1)} \left[\sum_{i=1}^{k} \mu_i^2 - \frac{k(k+1)^2}{4} \right] \tag{7.1}$$

where: (i) n is the number of data sets, (ii) k is the number of classification techniques and (iii) μ_i is the average rank for classification technique i which in turn is calculated as follows:

$$\mu_i = \frac{1}{n} \sum_{j=1}^{n} r_j \tag{7.2}$$

where r_j is the (AUC) rank for classification technique i on data set j.

The Friedman test was applied with respect to the proposed techniques in the context of the evaluation data sets. Two different cases were considered: (i) where the classification techniques was trained and tested on the same data set and (ii) where the classification techniques was trained on one data set and tested on another. In both cases the parameters that produced the best results in terms of the AUC measure, as obtained with respect to the experiments reported in the foregoing chapters, were used (as listed in Table 7.1). Recall that these results were obtained using TCV. More specifically the Friedman process is as follows:

1. Each of the proposed techniques is given a *rank* with respect to each data set as shown in Tables 7.2 and 7.3 (Table 7.2 shows the rankings when the classifier is trained and tested on the same data set, while Table 7.3 shows the rankings when the classifier is trained and tested on different data set). The ranks in both tables are presented in parenthesis where the best performing algorithm is given a rank of 1 and so on.

2. Note that (with reference to Tables 7.2 and 7.3) where two techniques share a ranking r, we used the so called *ties rule*. For example if two techniques are ranked fourth then they will be given a ranking of 4.5 ($\frac{4+4+1}{2}$)

3. The average rank μ for each of the proposed technique is given in the last column of the two tables using Equation 7.2 where $n = 8$.

4. The *Null hypothesis* (H_0) that there is no significant difference between the operation of the techniques, and the *Alternative hypothesis* (H_1) that there is were established.

5. Using the Friedman test there are two situations where the null hypothesis H_0 may be rejected, these are discussed in further detail later in this section.

6. The rejection of the null hypothesis H_0 means the automatic acceptance of the alternative hypothesis H_1.

7. If the null hypothesis is rejected we can proceed with a post-hoc test to identify the critical distances between pairs of techniques to identify which technique(s) are significantly different (in terms of recorded AUC values). It should be noted that we can not proceed with a post-hoc test if the null hypothesis H_0 is not rejected.

Number	Technique (variation)	Classifier Generator	Best parameters
1	Level 1 LGM	Decision tree	$d = 10$, $\|L\| = 3$ andδz representation
2	Level 2 LGM	Decision tree	$d = 10$, $\|L\| = 3$ and δz representation
3	Composite LGM	Decision tree	$d = 10$, $\|L\| = 3$ and δz representation
4	LDM	Decision tree	$d = 2.5$ and $\|L\| = 3$
5	LDM+ Level 1 LGM	Decision tree	$d = 10$ and $\|L\| = 3$
6	LDM+ Level 2 LGM	Decision tree	$d = 10$ and $\|L\| = 3$
7	LDM+ Composite LGM	Decision tree	$d = 10$ and $\|L\| = 3$
8	Point Series (PS)	k-NN with DTW technique	$d = 5$, 3×3 *key* PS representation

TABLE 7.1: The best parameter settings for the proposed techniques (variations) with respect to each 3D representation technique.

Before proceeding with the operation of the Friedman test, the level of significance (α), *p-value* and degree of freedom concepts should be clearly defined. The level of significance, known as α, is the probability of wrongly rejecting the null hypothesis H_0 where it is in fact true. Sometimes it is known as the level of risk. The commonly used value is $\alpha = 0.05$ [27, 76]. By using this value, there is 95% chance that the statistical results are real and not due to chance. The "critical value" is the χ^2 distribution of α and normally is denoted as χ^2_α. The *p-value* is defined as the probability of obtaining a result that is at least as extreme as the one we actually obtained assuming that the null hypothesis is true and it is typically $0 \leq p\text{-}value \leq 1$. More simply, it is the probability of obtaining the same results by chance. Normally, the *p-value* is compared with the α value. Figure 7.1 shows the χ^2 distribution curve where α is the shaded area under curve for χ^2_α while the *p-value* is the shaded area under curve for χ^2_F. With reference to the figure, if the $p\text{-}value > \alpha$, the test is inconclusive and more evidence will be required to support the alternative hypothesis (H_1), if the $p\text{-}value < \alpha$, then this means that we have a statistically significant result and hence the null hypothesis H_0 can be rejected. Finally, the degree of freedom is a positive number that indicates the *variability*. In our case the number of independent classifiers that have been generated using the different proposed techniques is $k = 8$ and thus the degree of freedom is $k - 1 = 7$.

As noted above, if the calculated Friedman test statistic χ^2_F is greater than the critical value for the Chi-square distribution χ^2_α obtained from a look up table of the form shown in Figure 7.4 then this means that the null hypothesis H_0 should be rejected and the alternative hypothesis H_1 should be accepted. However, this is not sufficient; to qualify the strength of evidences against the null hypothesis, a *p-value* is calculated.

As already noted the rejection of the null hypothesis H_0 indicates the existence of a significant difference amongst the proposed techniques, it does not provide information

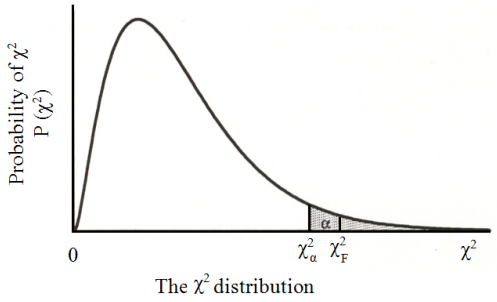


FIGURE 7.1: The χ^2 distribution. The shaded area is equal to α and denoted by χ^2_α, and represents the region of rejection. The *p-value* is the area under curve right of the calculated χ^2_F.

about the nature of this difference. Therefore, in the case where the null hypothesis is rejected we can proceed with a post-hoc test to determine which techniques performed differently. With respect to the work described in this thesis the Nemenyi test [155] was adopted. This operates using the concept of a "critical difference" calculated using Equation 7.3:

$$CD = q_{\alpha,\infty,k}\sqrt{\frac{k(k+1)}{12n}} \tag{7.3}$$

Where the critical value for $q_{\alpha,\infty,k}$ is calculated based on the Studentised range statistic divided by $\sqrt{2}$. Here, $k = 8$, $\alpha = 0.05$ and $q_{0.05,\infty,8} = 3.03$ according to a table of critical values for or $q_{\alpha,\infty,k}$ presented in [50]. A *Critical Difference* (CD) in this context is thus used to identify the difference between the average ranks of pairs of classifiers. A classifier performance is considered to be distinct from that of the other classifiers if their average ranks differ by at least the CD.

7.3 Using the Same Data Set for Statistical Comparison

The application of the Freidman test in the case where the classifiers were trained and tested on the same data set is considered in this section. From the work described in the foregoing chapters the best AUCs values produced for the techniques considered (listed in Table 7.1) are presented in Table 7.2. The Friedman statistical test was then applied to the eight different Gonzalo and Modified data sets (GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2). Table 7.2 presents the ranks (indicated in parentheses) and the average ranks μ for the $k = 8$ proposed techniques (using the best AUCs). From Table 7.2, it can be seen that the best average rank, 1.25, was achieved using the Point Series (PS) technique. The Friedman test value calculated using Equation 7.1 with $k - 1 = 7$ degrees of freedom, is as the follows:

$$\begin{aligned}\chi_F^2 &= \frac{12 \times 8}{8 \times (9)} \left[197.04 - \frac{8 \times (9)^2}{4}\right] \\ &= 1.33 \times 35.04 \\ &= 46.72\end{aligned}$$

Recall that the Friedman test statistic χ_F^2 is distributed according to the χ^2 distribution, so the computed value χ_F^2 was tested against the critical value of χ^2 with $\alpha = 0.005$ given 7 degree of freedom. Thus in this case $\chi_\alpha^2 = 20.28$ using the χ^2 statistical distribution table [66, 117]. As the calculated Friedman test value is $\chi_F^2 = 46.72 > 20.28$ the Null Hypothesis (H_0) could be rejected at *p-value* < 0.005, in other words there is a significant difference in the operation of the proposed techniques (when trained and tested on the same data set). The Critical Difference (CD) is then:

$$\begin{aligned}CD &= 3.03 \times \sqrt{\frac{8(9)}{12 \times 8}} \\ &= 3.03 \times 0.87 \\ &= 2.62\end{aligned}$$

In other words two individual techniques are significantly different if the difference between their average rank is at least 2.62. Figure 7.2 shows the average rank for the $k = 8$ proposed techniques along with the CD measure to highlight the techniques which are significantly different to each other. As shown in the figure, an *interval* is represented by a head (indicates the average rank) and the tail indicates the CD value. If two classifiers have non-overlapping intervals then the operation of both are significantly different. However, the overlapped intervals of two classifiers indicates that there is no significant difference between them. From the figure, it can be seen that the PS technique was ranked first which means that the operation of the PS technique outperformed the other techniques with an average rank of $\mu = 1.44$. The operation of LDM techniques was found to be the worst performing with an average rank of $\mu = 8$. Also, from the table and the figure we can note that:

1. The operation of the PS technique is significantly different with respect to LDM, Level 1 LGM and LDM + Level 1 LGM techniques while there is no significant difference between the operation of PS technique and Level 2 LGM , composite LGM, LDM + Level 2 LGM and LDM + composite LGM techniques as the difference between their average ranks is less than the CD value (overlapped intervals).

2. The average rank of the composite LGM was 3.63 and this is an indication that it is the "best" variation technique with respect to the different variations of the LGM technique. However, there is no significant difference between the operation of the composite LGM and the Level 2 LGM as shown in the figure.

3. The average rank of the Level 1 LGM was 6.63 which indicates that it is the "worst" technique with respect to the different variations of the LGM technique. However, the operation of Level 1 LGM was slightly improved when combined with the LDM technique where the average rank of the LDM + Level 1 LGM was 6.38.

4. The operation of the LDM + composite LGM technique is significantly different with respect to LDM, Level 1 LGM and LDM + Level 1 LGM techniques while there is no significant difference between the operation of PS and Level 2 LGM , composite LGM, LDM + Level 2 LGM and PS techniques as shown in the figure.

5. The LDM + composite LGM has an average rank of 2.87 with respect to other combinations between LGM and LDM techniques and this would be an indication that describing a 3D surface using a combination of local geometries and critical features may serve better than describing the 3D surface using either of them and consequently this would produce an effective classifier especially when it is generated using a *single shape*.

TABLE 7.2: The best AUC results for the proposed techniques (variations) using the same data sets for training and testing with respect to each 3D representation technique.

Technique	Dataset								μ_i
	GSV1	GSV2	GTV1	GTV2	MSV1	MSV2	MTV1	MTV2	
Level 1 LGM	0.62 (7)	0.83 (7)	0.69 (7)	0.73 (6)	0.66 (7)	0.76 (6)	0.67 (6.5)	0.64 (6.5)	6.63
Level 2 LGM	0.73 (4.5)	**0.96 (2)**	0.79 (4.5)	0.82 (2.5)	0.78 (4.5)	0.91(4.5)	0.75 (3.5)	0.74 (4)	3.75
Composite LGM	0.73 (4.5)	**0.96 (2)**	0.80 (2.5)	0.82 (2.5)	0.78 (4.5)	0.91 (4.5)	0.75 (3.5)	0.73 (5)	3.63
LDM	0.50 (8)	0.50 (8)	0.53 (8)	0.50 (8)	0.57 (8)	0.58 (8)	0.50 (8)	0.59(8)	8.00
LDM + Level 1 LGM	0.65 (6)	0.84 (6)	0.71 (6)	0.72 (7)	0.74 (6)	0.75(7)	0.67 (6.5)	0.64 (6.5)	6.38
LDM + Level 2 LGM	0.75 (2)	0.95 (4.5)	0.79 (4.5)	0.81 (4.5)	0.80 (2.5)	0.92 (2.5)	0.75 (3.5)	0.75 (2.5)	3.31
LDM+ Composite LGM	0.74 (3)	**0.96 (2)**	0.80 (2.5)	0.81 (4.5)	0.80 (2.5)	0.92 (2.5)	0.75 (3.5)	0.75 (2.5)	2.87
Point Series (PS)	**0.97 (1)**	0.95 (4.5)	**1.00 (1)**	**0.99 (1)**	**0.97 (1)**	**0.97 (1)**	**0.97 (1)**	**0.96 (1)**	**1.44**
$\sum_{j=1}^{k} \mu_j^2 = 197.04$ Friedman test statistic = 46.72									

7.4 Using Different Data Sets for Statistical Comparison

In the case where the classifier is generated (trained) using one data set and tested on the remaining $n-1$ data sets, the best recorded AUC values are reported in Table 7.3. The Friedman test value is as follows:

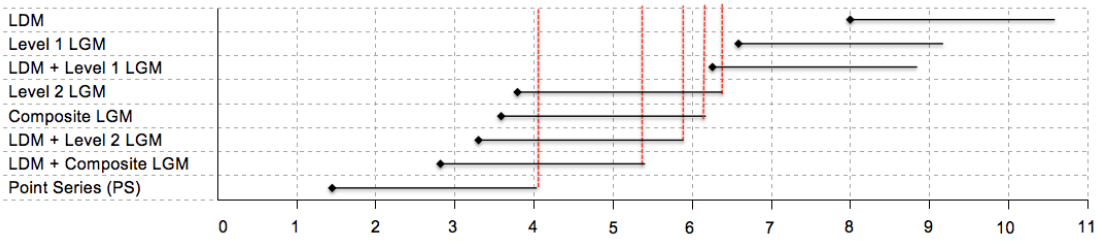


FIGURE 7.2: The average rank (μ_i) associated with CD value for the classifiers generated using the same data set.

$$\begin{aligned}\chi_F^2 &= \frac{12 \times 8}{8 \times (9)} \left[200.75 - \frac{8 \times (9)^2}{4}\right] \\ &= 1.33 \times 38.75 \\ &= 51.67\end{aligned}$$

Again, recall that the computed value of χ_F^2 was tested against $\chi_\alpha^2 = 20.28$. Given that the Friedman test statistic is $\chi_F^2 = 51.67 > \chi_\alpha^2 = 20.28$ then the Null Hypothesis (H_0), that there is no difference, can be rejected at $p\text{-}value < 0.005$; which means that it can be concluded that the average ranks for the $k = 8$ generated classifiers on the $n = 8$ different data sets are significantly different. The value of the Critical Difference (CD) in this case is 2.62 (which is identical to the CD value obtained for the classifiers generated on the same data set as the value of $q_{0.05,\infty,8}$, k and n are the same in both cases). Figure 7.3 shows the average rank for the $k = 8$ proposed techniques along with the CD measure to highlight the techniques which are significantly different to each other. From the table and the figure, it can be observed that:

1. LDM was found to be significantly the worst technique with an average rank of 8.

2. The operation of the PS technique was again found to be the best amongst the techniques considered with a best average rank of 1.25.

3. There is no significant difference between the operation of the PS technique and each of: Level 2 LGM, LDM + composite LGM and LDM + Level 2 LGM techniques as the difference between the average rank of the point series and their average ranks are: $1.31, 1.81$ and 1.88 respectively (all are < 2.62).

4. There is a significant difference between the PS technique and LDM + Level 1 LGM, Level 1 LGM, composite LGM and LDM techniques as the difference between the average rank of the PS technique and the average ranks of the others are: 4.00, 4.56, 5.69, 6.75 and respectively (all are > 2.62).

5. The operation of the Level 2 LGM technique was found to be the best technique with an average rank of 2.56 with respect to the different variations of the LGM technique. From the figure, it can be seen that there is a significant difference between the Level 2 LGM technique, the Level 1 LGM and the composite LGM.

6. The combination of the LDM and LGM techniques resulted in a better performance than when the techniques were used in isolation. The LDM + composite LGM was found to be the best combination with an average rank of 3.06 (as indicated in the table). However, there is no significant performance difference between the LDM + composite LGM and the LDM + Level 2 LGM (as shown in the figure).

7. The average rank of the Level 2 LGM is 2.56 and this would be an interesting result as it is a strong confirmation that using the local geometries is sufficient to describe a 3D surface is sufficient to produce an effective *generic* classifier especially on which *different* shapes are used to generate the classifier.

7.5 Run Time Analysis

This section presents some further insight to the run time analysis conducted with respect to the different proposed techniques. With reference to the run time analysis for the LGM, the LDM and the PS techniques that were previously presented in Chapters 4, 5 and 6 respectively, the reported run times indicated that there was no significant difference in run time between the different techniques. It should be noted that the initial set up time for the classification task was negligible for each technique and therefore the run time analysis presented in this section consider only the preprocessing phase and the application of a particular representation technique used to generate the required classifier. The average run time for the "best" variation of each technique obtained using the eight data sets (GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2) was as follows:

- 2.25 seconds for the composite LGM technique where $|L| = 3$ and $d = 10$ were the "best" parameters.
- 2.50 seconds for LDM + composite LGM technique where $|L| = 3$ and $d = 10$ were the "best" parameters.
- 3.00 seconds for PS technique where 3×3 PS variation and $d = 5$ were the "best" parameters.

.

Note that the average run time using the LDM technique on its own was 199.75 seconds where $|L| = 3$ and $d = 2.5$ were the "best" parameters. The long run time in this case was because the grid size of $d = 2.5$ (which was previously identified in Chapter 5 to be the most appropriate grid size for the LDM technique in terms of classification

effectiveness) generated more records and thus require a greater amount of processing time than the other techniques considered. Therefore, the LDM technique was not only found to be the worst technique in terms of AUC and accuracy, it was also the worst technique in terms of run time.

TABLE 7.3: The best AUC results for the proposed techniques (variations) using different data sets for training and testing the generated classifier with respect to each 3D representation technique.

Technique	Dataset								μ_i
	GSV1	GSV2	GTV1	GTV2	MSV1	MSV2	MTV1	MTV2	
Level 1 LGM	0.82 (6)	0.77 (5.5)	0.85 (6)	0.82 (6)	0.82 (6.5)	0.82 (5.5)	0.81 (5.5)	0.82 (5.5)	5.81
Level 2 LGM	0.90 (3.5)	0.90 (3)	0.90 (2.5)	0.96 (3.5)	0.94 (2)	**0.96(1)**	0.93 (3)	0.94 (2)	2.56
Composite LGM	0.70 (7)	0.74 (7)	0.75 (7)	0.81 (7)	0.82 (6.5)	0.80 (7)	0.76 (7)	0.76 (7)	6.94
LDM	0.52 (8)	0.50 (8)	0.61 (8)	0.50 (8)	0.60 (8)	0.60 (8)	0.50 (8)	0.60(8)	8.00
LDM + Level 1 LGM	0.83 (5)	0.77 (5.5)	0.86 (5)	0.83 (5)	0.86 (5)	0.82(5.5)	0.81 (5.5)	0.82 (5.5)	5.25
LDM + Level 2 LGM	0.90 (3.5)	0.90 (3)	0.90 (2.5)	0.96 (3.5)	0.92 (3.5)	0.95 (3)	0.93 (3)	0.88 (3)	3.13
LDM+ Composite LGM	0.94 (2)	0.90 (3)	0.89 (4)	0.97 (2)	0.92 (3.5)	0.95 (3)	0.93 (3)	0.86 (4)	3.06
Point Series (PS)	**0.99 (1)**	**1.00 (1)**	**0.99 (1)**	**1.00 (1)**	**1.00 (1)**	0.95 (3)	**1.00 (1)**	**0.99 (1)**	**1.25**
$\sum_{j=1}^{k} \mu_j^2 = 200.75$ Friedman test statistic = 51.67									

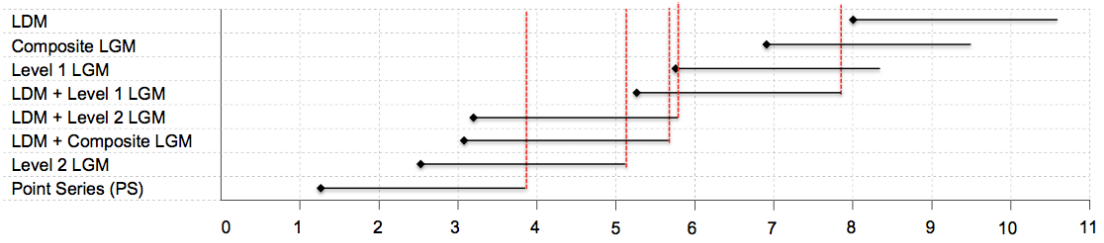


FIGURE 7.3: The average rank (μ_i) associated with CD value for the classifiers generated using different data sets.

7.6 Summary

This chapter has presented a statistical study concerning the performance of the different proposed methods. The statistical evaluation was undertakes using the Friedman test for related data sets. The comparisons were conducted in the context of two different cases: (i) classifiers trained and tested using the same data sets, and (ii) classifiers trained and tested on different data sets. The conducted statistical evaluations was performed using classifiers generated using the best performing (in terms of AUC) parameters for each proposed technique as established in the foregoing chapters. The statistical evaluation using the Friedman test demonstrated a significant differences between the proposed techniques, consequently the null hypothesis (that there was no statistical difference) was rejected when $p\text{-}value < 0.005$ in both cases. In other words, we are 99% certain that there is a significant difference between the operation the different proposed techniques. From the evaluation the following overall observations can be made:

1. The LDM technique demonstrated the worst performance compared to all other proposed techniques as it had the highest average rank value, therefore it can be concluded that the distance from a "critical feature" of a given shape on its own is not enough to generate an effective classifier.

2. The best overall technique was found to be the PS technique and this is an indication that this technique was best suited to capture local geometries so as to produce an effective classifier.

3. There is a significant difference between the operation of PS technique and the LDM technique.

4. The combination between the LDM and LGM techniques resulted in a better performance than when each was used in isolation.

The application of predicted springback values in the context of the AISF process is presented in the following chapter.

	Level of significance (α)										
df	**0.999**	**0.995**	**0.99**	**0.975**	**0.95**	**0.9**	**0.1**	**0.05**	**0.025**	**0.01**	**0.005**
1	0	0	0	0	0	0.02	2.71	3.84	5.02	6.63	7.88
2	0	0.01	0.02	0.05	0.1	0.21	4.61	5.99	7.38	9.21	10.6
3	0.02	0.07	0.11	0.22	0.35	0.58	6.25	7.81	9.35	11.34	12.84
4	0.09	0.21	0.3	0.48	0.71	1.06	7.78	9.49	11.14	13.28	14.86
5	0.21	0.41	0.55	0.83	1.15	1.61	9.24	11.07	12.83	15.09	16.75
6	0.38	0.68	0.87	1.24	1.64	2.2	10.64	12.59	14.45	16.81	18.55
7	0.6	0.99	1.24	1.69	2.17	2.83	12.02	14.07	16.01	18.48	20.28
8	0.86	1.34	1.65	2.18	2.73	3.49	13.36	15.51	17.53	20.09	21.95
9	1.15	1.73	2.09	2.7	3.33	4.17	14.68	16.92	19.02	21.67	23.59
10	1.48	2.16	2.56	3.25	3.94	4.87	15.99	18.31	20.48	23.21	25.19
11	1.83	2.6	3.05	3.82	4.57	5.58	17.28	19.68	21.92	24.73	26.76
12	2.21	3.07	3.57	4.4	5.23	6.3	18.55	21.03	23.34	26.22	28.3
13	2.62	3.57	4.11	5.01	5.89	7.04	19.81	22.36	24.74	27.69	29.82
14	3.04	4.07	4.66	5.63	6.57	7.79	21.06	23.68	26.12	29.14	31.32
15	3.48	4.6	5.23	6.26	7.26	8.55	22.31	25	27.49	30.58	32.8
16	3.94	5.14	5.81	6.91	7.96	9.31	23.54	26.3	28.85	32	34.27
17	4.42	5.7	6.41	7.56	8.67	10.09	24.77	27.59	30.19	33.41	35.72
18	4.9	6.26	7.01	8.23	9.39	10.86	25.99	28.87	31.53	34.81	37.16
19	5.41	6.84	7.63	8.91	10.12	11.65	27.2	30.14	32.85	36.19	38.58
20	5.92	7.43	8.26	9.59	10.85	12.44	28.41	31.41	34.17	37.57	40
21	6.45	8.03	8.9	10.28	11.59	13.24	29.62	32.67	35.48	38.93	41.4
22	6.98	8.64	9.54	10.98	12.34	14.04	30.81	33.92	36.78	40.29	42.8
23	7.53	9.26	10.2	11.69	13.09	14.85	32.01	35.17	38.08	41.64	44.18
24	8.08	9.89	10.86	12.4	13.85	15.66	33.2	36.42	39.36	42.98	45.56
25	8.65	10.52	11.52	13.12	14.61	16.47	34.38	37.65	40.65	44.31	46.93
26	9.22	11.16	12.2	13.84	15.38	17.29	35.56	38.89	41.92	45.64	48.29
27	9.8	11.81	12.88	14.57	16.15	18.11	36.74	40.11	43.19	46.96	49.65
28	10.39	12.46	13.56	15.31	16.93	18.94	37.92	41.34	44.46	48.28	50.99
29	10.99	13.12	14.26	16.05	17.71	19.77	39.09	42.56	45.72	49.59	52.34
30	11.59	13.79	14.95	16.79	18.49	20.6	40.26	43.77	46.98	50.89	53.67
32	12.81	15.13	16.36	18.29	20.07	22.27	42.58	46.19	49.48	53.49	56.33
34	14.06	16.5	17.79	19.81	21.66	23.95	44.9	48.6	51.97	56.06	58.96
36	15.32	17.89	19.23	21.34	23.27	25.64	47.21	51	54.44	58.62	61.58
38	16.61	19.29	20.69	22.88	24.88	27.34	49.51	53.38	56.9	61.16	64.18
40	17.92	20.71	22.16	24.43	26.51	29.05	51.81	55.76	59.34	63.69	66.77
42	19.24	22.14	23.65	26	28.14	30.77	54.09	58.12	61.78	66.21	69.34
44	20.58	23.58	25.15	27.57	29.79	32.49	56.37	60.48	64.2	68.71	71.89
46	21.93	25.04	26.66	29.16	31.44	34.22	58.64	62.83	66.62	71.2	74.44
48	23.29	26.51	28.18	30.75	33.1	35.95	60.91	65.17	69.02	73.68	76.97
50	24.67	27.99	29.71	32.36	34.76	37.69	63.17	67.5	71.42	76.15	79.49
55	28.17	31.73	33.57	36.4	38.96	42.06	68.8	73.31	77.38	82.29	85.75
60	31.74	35.53	37.48	40.48	43.19	46.46	74.4	79.08	83.3	88.38	91.95
65	35.36	39.38	41.44	44.6	47.45	50.88	79.97	84.82	89.18	94.42	98.1
70	39.04	43.28	45.44	48.76	51.74	55.33	85.53	90.53	95.02	100.43	104.21
75	42.76	47.21	49.48	52.94	56.05	59.79	91.06	96.22	100.84	106.39	110.29
80	46.52	51.17	53.54	57.15	60.39	64.28	96.58	101.88	106.63	112.33	116.32
85	50.32	55.17	57.63	61.39	64.75	68.78	102.08	107.52	112.39	118.24	122.32
90	54.16	59.2	61.75	65.65	69.13	73.29	107.57	113.15	118.14	124.12	128.3
95	58.02	63.25	65.9	69.92	73.52	77.82	113.04	118.75	123.86	129.97	134.25
100	61.92	67.33	70.06	74.22	77.93	82.36	118.5	124.34	129.56	135.81	140.17

FIGURE 7.4: The critical values for Chi-Square (χ^2) [66].

Chapter 8

Identified Springback Error Application

8.1 Introduction

Three different 3D surface representation techniques, LGM, LDM and PS, have been presented in Chapters 4, 5 and 6 respectively. The techniques have been extensively evaluated and compared as reported in Chapter 7. The evaluation has demonstrated that: (i) springback can be successfully predicted and (ii) that the PS technique is the most effective. However, being able to predict springback is not the end of the story. We want to be able to test the effectiveness of the identified springback values by applying them in an industrial sheet steel processing (AISF) context. More specifically we want to be able to apply the detected set of springback values E to the original G_{in} grid in order to produce a corrected grid G_{corr} ($G_{corr} = G_{in} - E$) which can then be used to define a new input cloud to be manufactured. If a better shape is produced it can then be argued that the proposed 3D surface representations are effective at least in the context of AISF springback prediction (the focus of the work presented in this thesis).

This chapter firstly presents a mechanism for the application of the predicted springback values so as to define a corrected input shape. Secondly the chapter presents an evaluation of the results. The evaluation was conducted using the Gonzalo and Modified Pyramids. Springback predictions were made with respect to both steel and titanium. The springback errors were then applied and corrected clouds generated. The corrected shapes were then manufactured by IBF and a comparison conducted between the newly manufactured shapes and the originally manufactured shapes.

For the springback prediction the Level One LGM 3D surface representation technique was used in combination with the *equal width* discretisation process. This was because the scheduling of experiments at IBF required a significant lead time and the LGM technique was the first fully operational technique produced with respect to the programme of work described in this thesis. Also, the equal width discretisation process was adopted at that time when the LGM technique was first proposed.

Table 8.1 presents the notation used in this chapter. The rest of this chapter is organised as follows. Section 8.2 presents the mechanism used to generate a corrected cloud C_{corr}. An analysis of the manufactured shapes produced is then presented in Section 8.3. The chapter is concluded with a summary in Section 8.4.

TABLE 8.1: Basic Notation used in this chapter.

Notation	Description
T	A desired shape.
T'	The actual obtained shape after the application of a manufacturing process.
C_{in}	The CAD description for a desired shape T in a point cloud format.
C_{out}	The description of T', measured using some optical measuring tool, in a point data format.
G_{in}	The grid representation for C_{in}, with associated errors values e_i when training a classifier, without error values otherwise.
C_{pred}	The predicted cloud generated with respect to springback errors produced by a classifier.
C_{corr}	The corrected cloud obtained after applying the proposed *correction* mechanism to C_{pred}.

8.2 Corrected Cloud Generation Mechanism

This section presents the proposed mechanism to generate a C_{corr} cloud for a given shape. The process commences with a desired C_{in} shape and a classifier generated using the RASP framework previously presented in Chapter 3. As noted above the Level One LGM surface representation technique was used in this context. In this manner four classifiers were generated:

1. Gonzalo Steel (GS).
2. Gonzalo Titanium (GT).
3. Modified Steel (MS).
4. Modified Titanium (MT).

The parameters used were $d = 1$ mm and $|L| = |L_E| = 5$. The generated classifiers were then applied to the Gonzalo and Modified C_{in} clouds and sets of error predictions produced: E_{GS}, E_{GT}, E_{MS} and E_{MT}.

Recall that the each $e_i \in E$ comprises: (i) a *magnitude* and (ii) a *direction* on which the $+$ indicates an "inward" direction and the $-$ indicates an "outward" direction (as shown earlier in Figure 3.5 in Chapter 3). The mechanism to generate the C_{corr} cloud is shown in Figure 8.1 and operates as follows.

1. **Input Shape Description**. The mechanism commences with the C_{in} of a given shape as an input for a classifier that was previosely generated using one of the

proposed techniques, the Level One LGM in particular which consists of a set of eight attributes $\{att_1, att_2, \cdots, att_8\}$ to describe the geometrical information of the surrounding neighbours for each $p_i \in C_{in}$.

2. **Generate Error Table**. Recall that the predicted springback errors are represented using a set of class attribute labels L_E. The mechanism for defining the class attribute (error), labels was described in Chapter 3 (Section 3.5). Although $|L_E| = 3$ was defined as the best label size for the LGM technique, for the evaluation $|L_E| = 5$ was used because this was requested by the industrial partner (IBF); $|L_E| = 5$ generates smoother surface than other $|L_E|$ values. A table was generated to translate the identified (predicted) error labels back into the mean error value for the range associated with each label (mean values were used due to the skewed nature of springback distribution). Thus $|L_E| = \{l_1, l_2, l_3, l_4, l_5\}$.

3. **Predict Error Labels**. Each $p_i \in C_{in}$ associated with an error label when a generated classifier is applied.

4. **Identify Predicted Error Values**. The predicted error *values* were generated by replacing the predicted error *labels* with their corresponding *mean* values using the error table produced in (1).

5. **Apply Predicted Error Value**. The C_{corr} cloud was then generated by reversing the error along the direction of the normal for each point $p_i \in C_{in}$.

Using the above process four corrected clouds were generated; (i) $C_{corr_{GS}}$, (ii) $C_{corr_{GT}}$, (iii) $C_{corr_{MS}}$ and (iv) $C_{corr_{MT}}$ corresponding to the four generated classifiers listed above. These were manufactured by IBF. It should be noted that the proposed error application mechanism offers the twin advantages that it is both simple and easy to apply. However, other error application mechanisms can be envisioned such as associating a weighting factor with different types of springback, however such mechanisms are beyond the scope of this thesis and require more investigations.

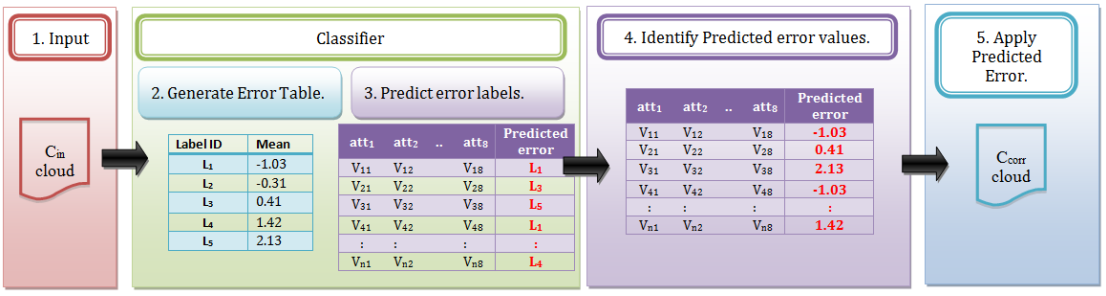


FIGURE 8.1: Corrected Cloud Generation Mechanism.

8.3 Evaluation of Formed Parts

In this section an evaluation with respect to the manufactured shapes is presented. The section is divided into two according to the manufacturing material used. Section 8.3.1 presents the analysis of the results with respect to shapes made out of steel (GS and MS), while Section 8.3.2 presents the analysis of the results with respect to shapes made out of titanium (GT and MT).

8.3.1 Steel Manufactured Shapes (GS and MS)

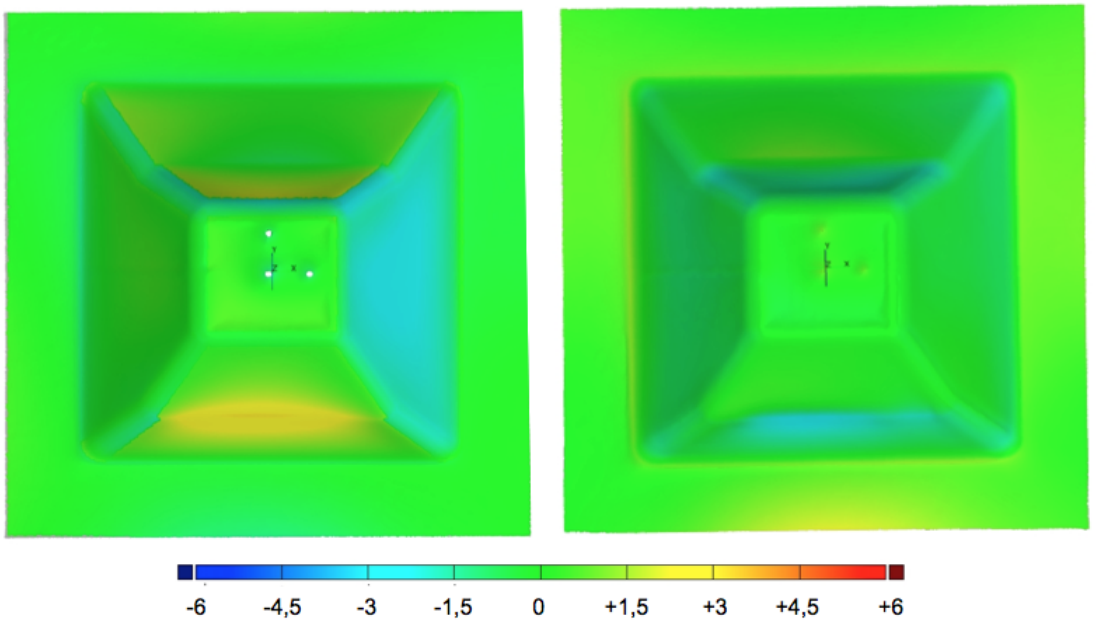


FIGURE 8.2: Springback distribution with respect to the shapes manufactured using C_{in} cloud (left) and C_{corr} (right) for the GS shape and an error scale of ±6 mm.

This section presents and discusses the results obtained in the context of the *Steel* fabricated shapes GS and MS. Figures 8.2 to 8.4 present the springback distribution over the C_{in} cloud for the GS shape using different error *scales* ±6 mm, ±4 mm and ±3 mm respectively. Note that an error scale is used to described the springback distribution and this is included at the bottom of each figure. These scales are divided into sub ranges where each sub range has a different colour from other sub rages. In each figure, the left hand side shows the shape produced using the C_{in} cloud while the right side shows the shape produced using the C_{corr} cloud with respect to the different error scales.

From the figures it can be seen that when the scale covers a small range the springback distribution becomes more obvious and clearer for both shapes and consequently it can be seen that the springback distribution over the new fabricated shapes, defined using the springback prediction process described earlier in this thesis, is clearly minimised. This is an interesting result as it shows that the generated classifiers were successfully able to predict the springback to a sufficient standard so that the springback in the shape manufactured, using the corrected clouds, was significantly reduced.

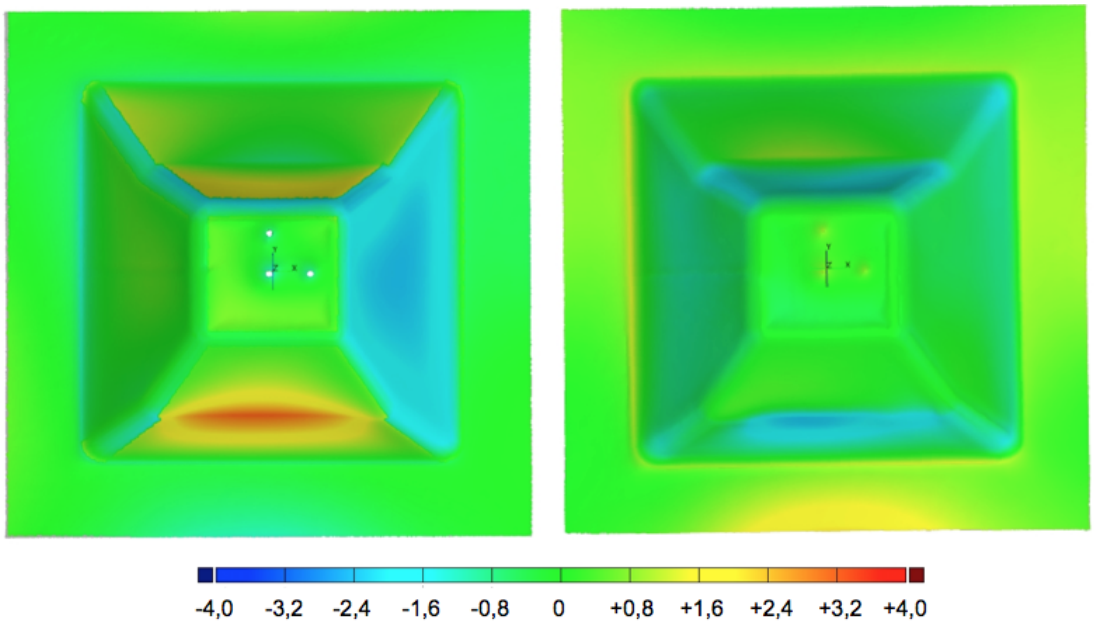


FIGURE 8.3: Springback distribution with respect to the shapes manufactured using C_{in} cloud (left) and C_{corr} (right) for the GS shape and an error scale of ±4 mm.

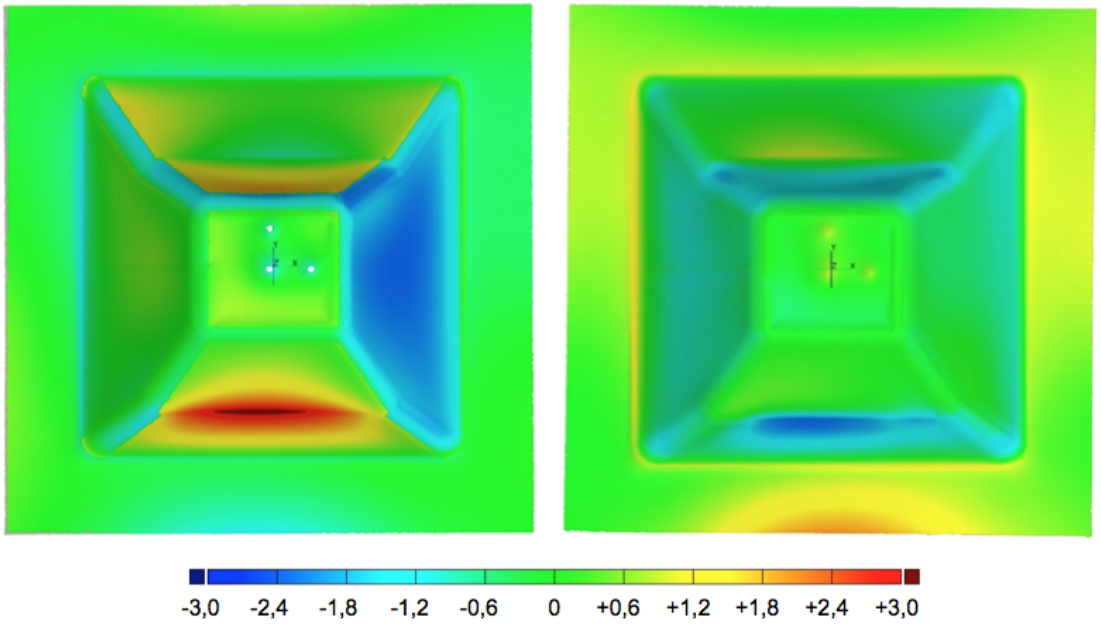


FIGURE 8.4: Springback distribution with respect to the shapes manufactured using C_{in} (left) and C_{corr} (right) for the GS shape and an error scale of ±3 mm.

Similarly results were recorded with respect to the manufactured shapes produced using the C_{in} and C_{corr} clouds for the MS shape. The springback distributions in this case are presented in Figures 8.5 to 8.7 (again using the same error scales, ±6 mm, ±4 mm and ±3 mm respectively). The Figures again confirm that the degree of springback can be effectively minimised using the proposed prediction mechanism. Some statistical information, regarding the GS and MS manufactured shapes, is presented in Tables 8.2 and 8.3. Given these results the following observations can be made:

- For the GS shape, the springback for the shape manufactured using the corrected shape was minimised, as can be clearly observed with reference to the maximum, minimum, average and standard deviation measures presented in the table.
- Similarly for the MS shape, the overall degree of springback was also minimised (despite the observed increase in the maximum springback value).
- Both Tables 8.2 and 8.3 indicate that the maximum and the minimum predicted springback values are very close to the maximum and the minimum actual springback values for both manufactured shapes (GS and MS).

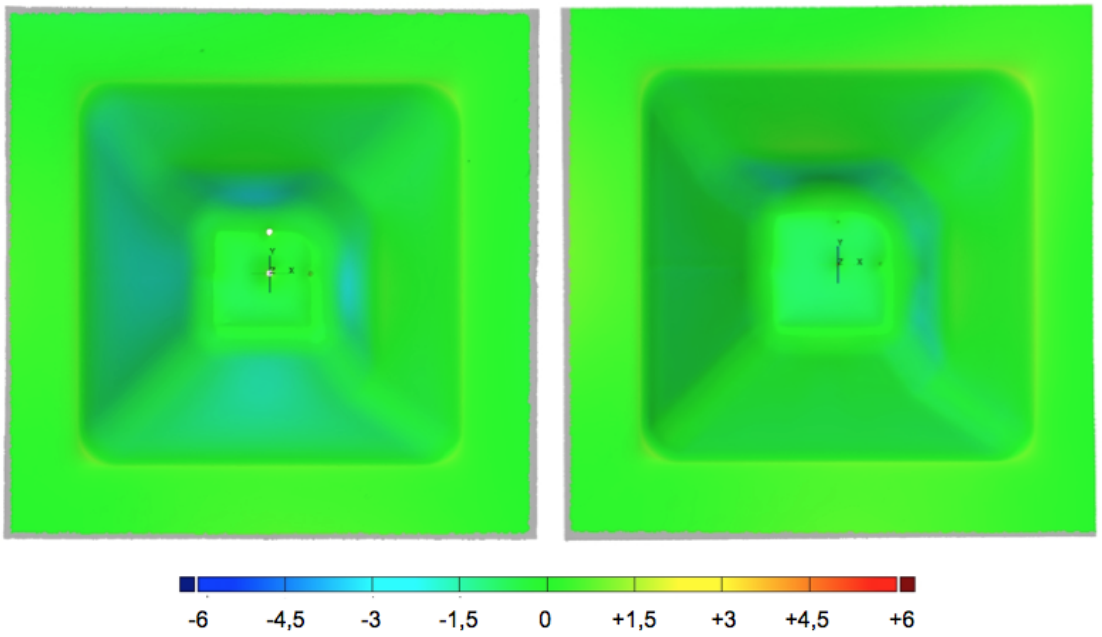


FIGURE 8.5: Springback distribution with respect to the shapes manufactured using C_{in} (left) and C_{corr} (right) for the MS shape and an error scale of ±6 mm.

With respect to the above it should be noted that the generated classifiers will not be able to predict the exact known maximum and minimum springback values because the proposed process demands that springback values are ranged (each with an associated *labelling*) so as to generate the necessary binary valued data sets. Recall that the classifier used with respect to the experiments presented in this chapter were generated using the Level one LGM technique with $|L| = 5$ ($L = \{l_1, l_2, l_3, l_4, l_5\}$)[1]. For

[1]The labelling is interpreted as $\{very\ low, low, neutral, high, very\ high\}$.

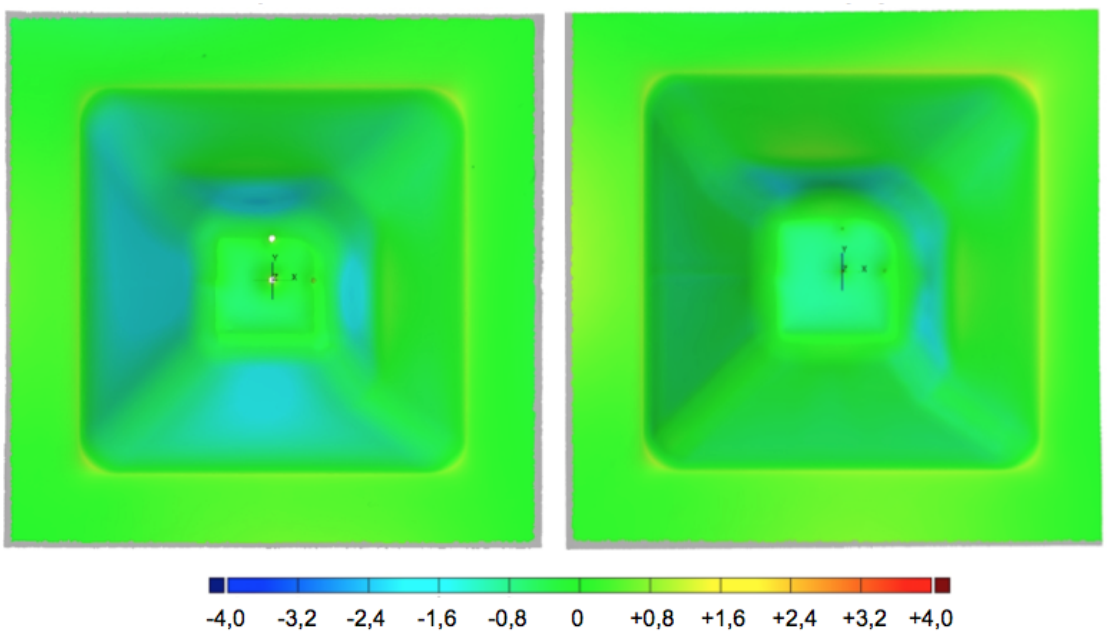


FIGURE 8.6: Springback distribution with respect to the shapes manufactured using C_{in} (left) and C_{corr} (right) for the MS shape and an error scale of ±4 mm.

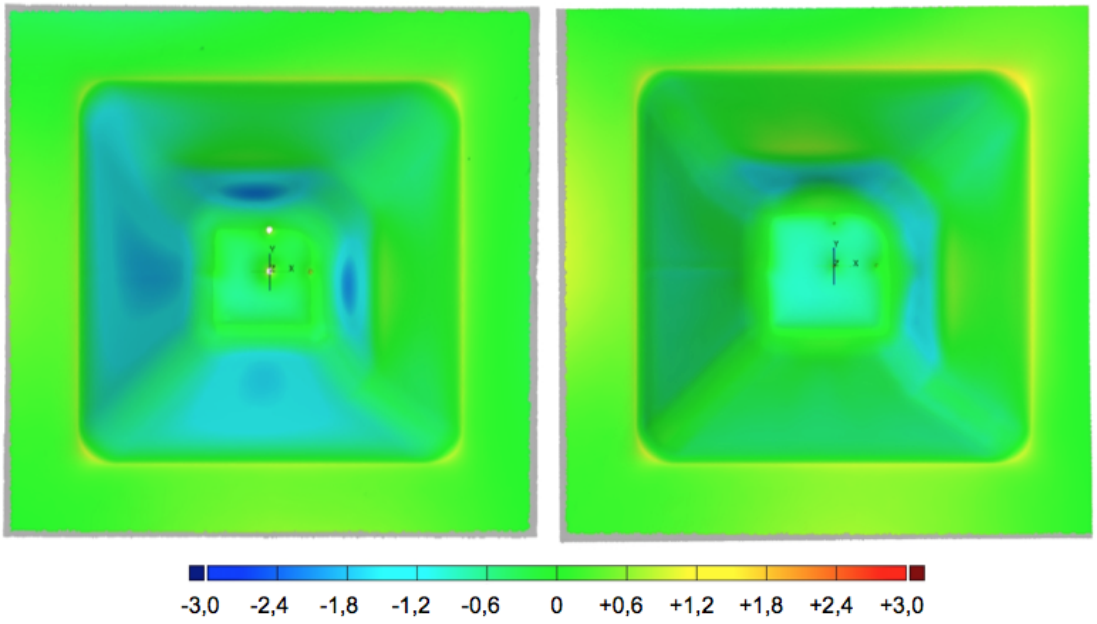


FIGURE 8.7: Springback distribution with respect to the shapes manufactured using C_{in} cloud (left) and C_{corr} (right) for the MS shape and an error scale of ±3 mm.

TABLE 8.2: Springback statistical information (provided by IBF) for the GS shapes manufactured using C_{in} and C_{corr}, (year experiment was conducted included in parenthesis).

	Max.	Min.	Average	SD
GS Springback using C_{in} (2012)	+3.24	−2.39	−0.24	+0.89
GS Springback using C_{corr} (2013)	**+2.15**	**-2.33**	**-0.06**	**+0.74**

TABLE 8.3: Springback statistical information (provided by IBF) for the MS shapes manufactured using C_{in} and C_{corr}. (Year experiment was conducted included in parenthesis.)

	Max.	Min.	Average	SD
MS springback using C_{in} (2012)	**+1.22**	−2.17	−0.30	+0.60
MS springback using C_{corr} (2013)	+1.27	**-1.56**	**-0.04**	**+0.48**

the classifier to operate correctly, the unseen data must be labelled in the same manner which means that the same set of error (springback) labels L_E used to label the error of the generated classifier were used again to label the unseen data. Table 8.4 shows some statistical information about the L_E label set used to describe the springback for the GS shape while Table 8.5 presents the same statistical information for the labels used for the MS shape.

With respect to the work presented in this thesis AUC and the accuracy measurements were used to evaluate classifier performance in the context of springback label prediction. However, by substituting the predicted labels with the mean values for the each label (as in the case of the experiments presented in this chapter), the *Root Mean Square Error (RMSE)* measurement can be used to compare the accuracy of the predicted springback values with the known springback values. Thus, AUC and accuracy have been used in this thesis as measurements of prediction (classifier) performance, whilst RMSE was used as an "accuracy" measurement for the actual springback prediction. RMSE is calculated as shown in Equation 8.1 [121].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(E_i)^2}{n}} \tag{8.1}$$

where n is the number of records in the data sets, $E_i = X_i - X_{pred}$, X_i is the known springback and X_{pred} is the predicted springback value. The RMSE value serves as a measure of how far, on average, the error (the difference between the predicted X_{pred} and the actual X_i springback value) is from zero.

Evaluation using the RMSE measure was performed on two levels: (i) *separately* for each label and in this case n would be the number of records associated with each label and (ii) for the *overall* shape, regardless of the number of labels, where n is the total number of records with respect to each shape. The RMSE results using GS and MS, along with some statistical information, are presented in Tables 8.4 and 8.5 respectively. Each row demonstrates the statistical information associated with each label. For instance, the maximum error (springback) value for l_1 for the GS shape was -0.81 while the minimum error value was -1.71 (as shown in Table 8.4). The mean value for l_1 for the GS shape was thus -1.03 and this value was used to replace label l_1. Similarly for the rest of labels for GS and MS shapes. The last row in each table gives some statistical information for the C_{corr} cloud obtained for GS and MS respectively

when the predicted labels were replaced with their equivalent mean values. For instance, the maximum replaced value for labels was +2.13 while the minimum replaced value for the labels was −1.03, these values correspond to the mean values of the set of labels $L_E = \{l_1, l_2, l_3, l_4, l_5\}$. For the GS shape, the mean value for the complete set of values for the labels was +1.24 and the Standard Deviation (SD) was +0.55. From the results presented in both tables, it can be seen that:

1. The SD values of the individual labels for the GS and MS shapes were less than the SD value of the entire C_{corr} and this is a typical result since each label has similar error (springback) values so the difference between individual error values and the mean error value for the label was very small. However, there was a significant difference between the individual error (springback) value and the overall mean value of the entire shape, with respect to C_{corr} (last row in each table).

2. The closer the RMSE value is to zero the better the *prediction* is.

3. The best RMSE value was 0.53 obtained for MS with respect to l_3; while the best RMSE value for GS was 1.03.

4. MS shape obtained better (lower) RMSE values compared to the RMSE values obtained using the GS shape for each label.

5. The best RMSE value was obtained with respect to the MS shape (1.00) compared to the RMSE value for the GS shape (1.62). These results indicated that the classifier generated using the MS dataset is more accurate than the classifier generated using the GS dataset.

6. The recorded RMSE values indicated that the classifiers were better able to predict the smaller springback values (and $springback = 0$ associated with l_3) than the larger springback values (associated with labels L_1 and L_5); this is possibly because that the small springback values have a higher distribution and more patterns with which to identify them than the large springback values.

TABLE 8.4: Statistical information concerning the L_E label set used to describe the springback values with respect to GS shape manufactured using C_{corr}.

L_E	No. of records	Max.	Min.	Mean	SD	RMSE
l_1	1613	−0.81	−1.71	−1.03	0.20	2.51
l_2	21641	0.09	−0.81	−0.31	0.22	1.73
l_3	8198	0.98	0.09	0.41	0.24	1.03
l_4	5076	1.88	0.98	1.42	0.24	1.59
l_5	530	2.74	1.88	2.13	0.24	2.62
C_{corr}	37058	+2.13	−1.03	+1.24	+0.55	1.62

TABLE 8.5: Statistical information concerning the L_E label set used to describe the springback values with respect to MS shape manufactured using C_{corr}.

L_E	No. of records	Max.	Min.	Mean	SD	RMSE
l_1	4146	−0.50	−1.09	−0.61	0.10	0.85
l_2	16654	0.08	−0.50	−0.22	0.16	0.76
l_3	10056	0.67	0.08	0.34	0.16	0.53
l_4	3984	1.25	0.67	0.93	0.17	1.19
l_5	1072	1.84	1.25	1.39	0.10	2.16
C_{corr}	35912	+1.39	−0.61	+0.49	+0.55	1.00

8.3.2 Titanium Based Shapes

This section reports on the main findings obtained with respect to the corrected flat topped pyramid shapes manufactured from Titanium (the GT and MT shapes). Unfortunately these experiments were unsuccessful because of fractures occurring early on during the manufacturing process. Figures 8.8 and 8.9 shows the maximum level of forming reached by the forming process while producing the GT and MT shapes respectively. Some parameters for the AISF machine were changed, but the cracks continue to reproduce and distort the manufactured shape. The main cause of these fractures was a property of Titanium called the anisotropic property whereby the *wall angle* in the C_{corr} shape should not exceed a certain maximum, approximately 60°. The maximum wall angle in the C_{corr} cloud reached 66° for GT1 and 64° for MT1. The maximum wall angle in the original desired shape C_{in} reached 55° for both GT1 and MT1. It was also unfortunate that the author of this thesis, not being a materials scientist, was unaware of the Titanium anisotropic property (more details on material properties can be found in [199]). Thus, if similar corrected shapes, generated using the proposed classification techniques, are obtained then we would expect to face the same problem. Some work, such as [104], has proposed a solution to solve this kind of problem during the manufacturing process where the shape is formed using a multi stage forming process. Although the wall angles should be kept below the critical wall angle in the context of titanium, the mechanism to achieve this incremental forming with respect to the work presented in this thesis still needs more investigations.

8.4 Summary

This chapter has presented a mechanism to generate and apply corrections to the C_{in} definition of a given shape so as to produce a corrected input shape (C_{corr}) that compensates the springback phenomena (in the context of the AISF process). A number of C_{corr} clouds were produced for four different shapes (MS, GS, MT and GT). These shapes were manufactured by IBF. Analysis of the manufactured shapes confirmed that the overall springback on the newly produced shapes was reduced compared to the shapes

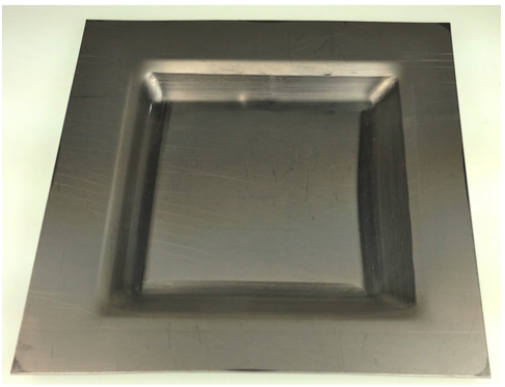


FIGURE 8.8: The uncompleted GT shape manufactured up to the point where fractures occurred.

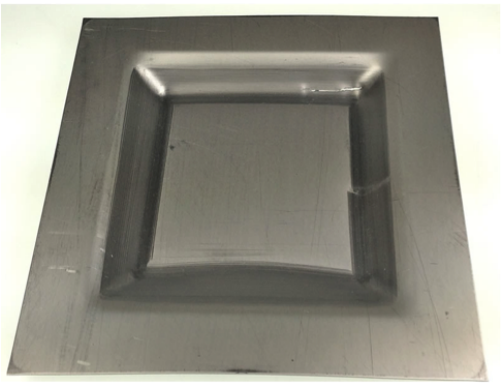


FIGURE 8.9: The uncompleted MT shape manufactured up to the point where fractures occurred.

formed using the original input cloud C_{in}. The process presented in this chapter provides the foundation for an intelligent process model that has potential for adoption by industry. The intelligent process model concept (two variations) is discussed further in the following chapter.

Chapter 9

Intelligent Process Model

9.1 Introduction

In the previous chapter a mechanism was presented and evaluated for applying predicted springback errors so as to produce a corrected shape C_{corr}. This chapter presents an Intelligent Precess Model (IPM) which is designed to automate this process. Two variations of the IPM are proposed: (i) *Single Pass* and (ii) *Iterative*. The Single Pass approach operates in a very similar manner to the process adopted in Chapter 8. A classifier, specifically built for the purpose, is applied to a given C_{in} cloud and the predicted errors are applied (by reversing the error) to produce a corrected cloud C_{corr}. In the iterative approach, the springback predictions are made and applied in an iterative manner until a "best" C_{corr} cloud is obtained. The Single Pass IPM can thus be seen as a special case of the Iterative IPM with only one iteration.

The rest of this chapter is organised as follows. Section 9.2 presents an overview of the Single Pass IPM. The iterative IPM is presented in Section 9.3; the section presents the philosophy behind the iterative IPM and the steps that comprise the iterative IPM. An extensive evaluation of the iterative IPM is presented in Section 9.4. Note that the operation of the Single Pass IPM was extensively evaluated in Chapter 8 so the evaluation of this IPM is not considered further in this chapter. Finally, the chapter is concluded with a summary in Section 9.6.

9.2 Single Pass IPM

The operation of the Single Pass IPM is essentially described in the previous chapter (Chapter 8) except that the process is packaged together as described in this section. Figure 9.1 presents the Single Pass IPM process, on which it starts with the *input* shape description C_{in} and ends with obtaining the C_{corr} cloud.

The C_{in} cloud is preprocessed using one of the proposed 3D surface representations (the PS representation has been shown to be the most effective) and a previously generated classifier applied to produce predicted springback values. Note that it is important that the conducted preprocessing is identical to that applied to the training set used to

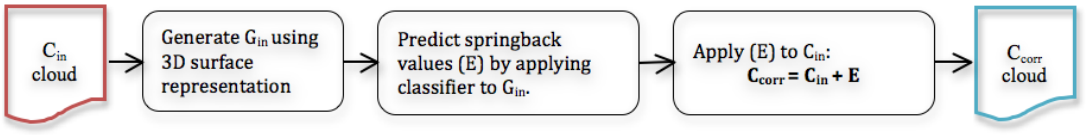


FIGURE 9.1: Single Pass IPM.

generate the classifier (same 3D surface representation, same value for d and the same L and L_E label sets). Recall that the required classifier is obtained using the RASP framework (presented previously in Chapter 3). Then the predicted errors are used to generate a corrected version of the C_{in}, C_{corr}, which can be then translated into a CAD description so as to be used as a new *input* to the forming process in order to minimise the effect of springback so that a shape closer to the desired shape is obtained than that which would have been obtained using the original C_{in} cloud.

The Single Pass IPM algorithm is presented in Algorithm 9.1. The algorithm starts with the C_{in} of a given shape and a previously generated classifier. One of the 3D representation techniques that have been proposed in this thesis is then used to translate the C_{in} into a G_{in} to which the classifier can be applied (as shown in line 1). Once the classifier, has been applied to G_{in} each $p \in G_{in}$ will be associated with one of the labels from L_E which is then replaced by its equivalent mean value e. Recall that the value e is always associated with a sign "+" or "−" (as illustrated earlier in Chapter 3). The sign indicates that e is already the *inverse* of the springback since we choose to associated − for the *outward* direction and + for the *inward* direction (See Chapter 3 Figure 3.5). Hence when applying springback values we apply them in a positive manner to C_{in} as indicated in Figure 9.1. Finally the corrected cloud is generated using Procedure *GenerateCorrectedCloud*1 (line 3). The coordinates of each point located in the corrected cloud C_{corr} are obtained by applying the associated error e to its corresponding point in the C_{in} cloud in the opposite direction along the normal from each point $p \in G_{in}$ as shown in the *GenerateCorrectedCloud*1 procedure (in particular, lines 4).

Algorithm 9.1: Single Pass IPM

Input: C_{in}, classifier
Output: C_{corr} cloud

1 $G_{in} \leftarrow$ Process C_{in} using appropriate 3D surface representation.
2 $E \leftarrow$ Set of springback prediction values obtained from applying classifier to G_{in}, each value in E is correlated to a point p in G_{in}.
3 $C_{corr} \leftarrow$ GenerateCorrectedCloud1(L_E, G_{in}, C_{in}).

The Single Pass IPM approach was extensively evaluated in Chapter 8 and thus this is not considered further in this chapter. The significance of the Single Pass IPM, in the context of this chapter, is that it provided the foundation for the Iterative IPM presented in the following Section.

Function GenerateCorrectedCloud1(E, G_{in}, C_{in})

```
1 C_corr ← C_in;
2 for all (p_in) ∈ C_in and (p_corr) ∈ C_corr do
3 |   e ← Value in E corresponding to p (already reversed sign);
4 |   p_corr ← p_in + e ;
5 end
```

9.3 Iterative IPM

This section presents the iterative IPM process as mentioned in the introduction to this chapter. The main idea behind the iterative IPM is to perform the springback prediction and the application process repeatedly until a "best" C_{corr} cloud is obtained. Best in this context refers to a C_{corr} cloud that is expected to minimise the springback phenomena.

The Iterative IPM process is presented in block diagram form in Figure 9.2. The iterative element of the process can clearly be seen from the figure. From the figure it can be observed that the process incorporates three types of clouds: (i) C_{in} *input* cloud (ii) the predicted cloud C_{pred} and (iii) the corrected cloud C_{corr}. The C_{in} cloud (coloured in *red*) is the description of the desired shape (the target shape). The predicted cloud C_{pred} (coloured in *purple*) is used to see whether the current *input* is close enough to the desired shape C_{in}. The corrected cloud C_{corr} (coloured in *blue*) is generated to provide the Iterative IPM with a new *input* for the next iteration (where required). The iterative IPM operates as follows.

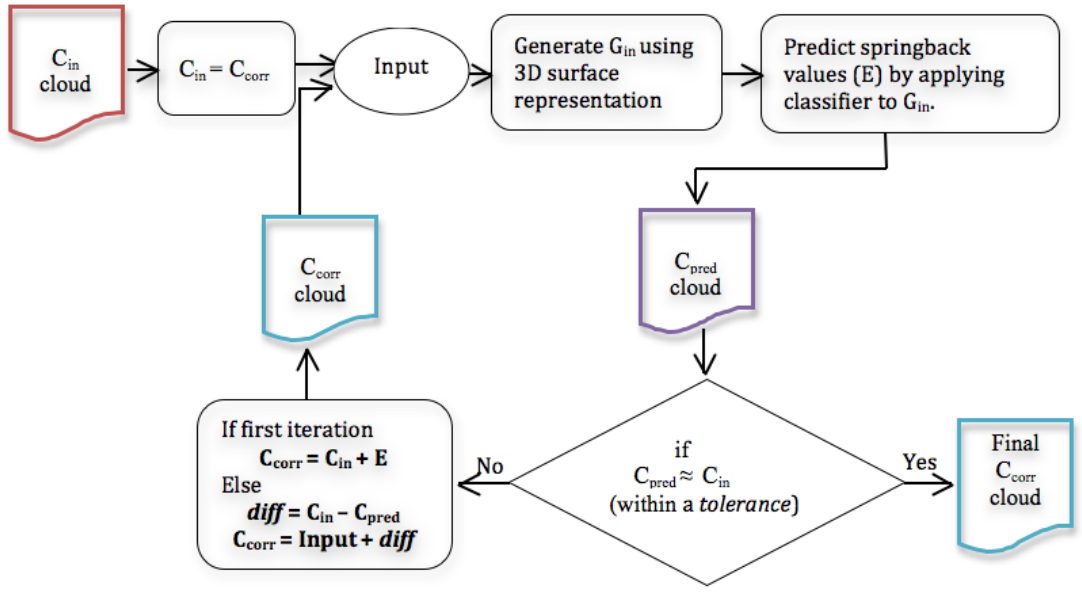


FIGURE 9.2: Iterative IPM.

At start up the process is similar to that for the Single Pass IPM process (see above). The C_{in} cloud is preprocessed using one of the proposed 3D surface representations to give G_{in} and a previously generated classifier is applied to produce the desired set of

predicted springback values $E = \{e_1, e_2, \ldots, e_n\}$ one value per grid centre point in G_{in}. Recall that errors values are given a negative sign if the springback is upwards with respect to G_{out} and a positive sign otherwise (as described previously in Chapter 3). Recall that the springback values are already inverse, so to get the original value of the error the "inverse of the inverse" E are considered and then applied to G_{in} to produce a predicted shape C_{pred} as shown in Equation 9.1.

$$C_{pred} = C_{in} - E \tag{9.1}$$

If, on this first iteration, $C_{pred} \approx C_{in}$ (to a certain level of tolerance) then the process finishes and the original C_{in} is passed on ready for forming. However, this is an unlikely event as this will mean that the predicted springback errors all equated to approximately 0 (or below the tolerance value). In practice the predicted shape will not be the same as the desired shape on the first iteration. As presented previously in Chapter 6, a tolerance of 0.08 mm has been suggested by BS EN ISO 1101:2005 [29] and used with respect to the PS representation to identify the similarity between two curves; this tolerance value can be also used in the context of the iterative IPM in order to identify the maximum difference of error (springback) between C_{pred} and C_{in} that can be acceptable. Thus, as in the case of the Single Pass IPM, the predicted springback errors are applied to the C_{in} cloud to produce a corrected cloud C_{corr} as shown in Equation 9.2.

$$C_{corr} = C_{in} + E \tag{9.2}$$

In the case of the Single Pass IPM the C_{corr} shape was then passed on for manufacture. In the case of the Iterative IPM the process is repeated. The C_{corr} cloud is preprocessed in the same way as before and fed back into the classifier and a new C_{pred} produced. If this new C_{pred} is now near enough to C_{in} the process stops and C_{corr} passed on for manufacture. If it is not the set of differences *diff* between the new C_{pred} and the original C_{in} clouds is determined and this set of difference applied to the *input* cloud to produce C_{corr} which is a new *input* for the iterative IPM:

$$\begin{aligned} \mathit{diff} &= C_{in} - C_{pred} \\ C_{corr} &= \mathit{input} + \mathit{diff} \end{aligned}$$

The *diff* values are described in terms of a magnitude and a direction and calculated in the same manner as springback as described previously in Chapter 3 (Section 3.4). The magnitude is the *distance* along the normal from the point located on C_{in} obtained from the C_{pred} cloud while the direction is determined according the position of the C_{pred} points with respect to the C_{in} points. If the *diff* value is upwards with respect to C_{in} it will assigned a negative sign. Otherwise a positive sign is assigned (as described

previously in Chapter 3). Note that the *diff* values will not necessarily be the same across the shape.

The process continues until either: (i) C_{pred} approximates to C_{in} where the average difference between both clouds satisfy a prescribed tolerance or (ii) a maximum number of iterations has been reached. Algorithm 9.2 presents the iterative IPM process. For experimental purposes $n = 12$ was used. The inputs to the algorithm are the desired (target) cloud C_{in}, a classifier and the max number of iterations n.

The algorithm starts by generating G_{in} using one of the proposed surface representation techniques (line 1). The classifier is then applied to G_{in} so that each grid centre point will be associated with one predicted error value. The set of all the predicted error values define the set E (line 2). C_{in} is assigned to the *input* cloud (line 3). The predicted cloud C_{pred} is then generated using Function *GeneratePredictedCloud* (line 4). If C_{pred} is approximately the same as the desired (target) shape C_{in}, then the C_{corr} is the *input* and the algorithm stops. Otherwise, the C'_{corr} is generated (line 9) using Function *GenerateCorrectedCloud*1 (described earlier for the Single Pass IPM) and used as an *input* for the next iteration (line 9) where Algorithm 9.3 is applied to obtain the best C_{corr} (line 10).

Algorithm 9.2: Main Iterative IPM process

Input: C_{in}, classifier, number of iterations n
Output: C_{corr} cloud

1 $G_{in} \leftarrow$ Process C_{in} using appropriate 3D surface representation ;
2 $E \leftarrow$ Set of springback prediction values obtained from applying classifier to G_{in}, each value in E is correlated to a point p in G_{in};
3 $input \leftarrow C_{in}$;
4 $C_{pred} \leftarrow$ GeneratePredictedCloud (E, G_{in}, $input$);
5 **if** $C_{pred} \approx C_{in}$ **then**
6 | $C_{corr} \leftarrow input$;
7 **end**
8 **else**
9 | $C'_{corr} \leftarrow$ GenerateCorrectedCloud1(E, G_{in}, $input$) ;
10 | $C_{corr} \leftarrow$ IterativeIPM(C'_{corr}, C_{in}, n) ; `// Algorithm 9.3`
11 **end**
12 **return** C_{corr}

The C_{pred} of a given C cloud is generated using Function *GeneratePredictedCloud* where the set of predicted error values E, G_{in} and the C are the inputs to the function. Recall that each centre grid point in G_{in} is associated with an e value, thus e' is the inverse of the e (opposite sign). The C_{pred} of the given C cloud is obtained by apply the inverse of the predicted error values e' to the points of C cloud (Function GeneratePredictedCloud line 6).

Algorithm 9.3 describes the iterative part of the Iterative IPM starting from iteration 2 (as shown in line 1). The inputs are: (i) the original C_{in} cloud, (ii) the maximum number of iterations n and (iii) the current *input* cloud. The *input* cloud is assigned

Function GeneratePredictedCloud(E, G_{in}, C)

Input: E, G_{in}, C
Output: C_{pred} cloud

```
1 C_pred ← C;
2 for all p ∈ C and p_pred ∈ C_pred do
3 |   p ← Grid centre point in G_in corresponding to (x, y) ;
4 |   e ← Value in E corresponding to p ;
5 |   e' ← Value of e with reversed sign ;
6 |   p_pred ← p + e' ;
7 end
8 return C_pred
```

initially to the C'_{corr} resulting from the first iteration with respect to the main iterative process described in Algorithm 9.2, then it assigned to the C_{corr} cloud (Algorithm 9.3 line 2). An iteration counter is used to compare with n as the process proceeds. Again a C_{pred} is generated for each iteration using Function *GeneratePredictedCloud* (line 6) in such a way that if C_{pred} is close enough to the C_{in}, then the C_{corr} is the output. Otherwise, the C_{corr} for the current *input* is generated using Function GenerateCorrectedCloud2 and used as the new *input* for the next iteration (lines 8 and 9 respectively). Thus, the algorithm will be terminated when the iterative IPM is applied for n times (line 3). Finally, the output from Algorithm 9.3 is the final C_{corr} cloud. Function *GenerateCorrectedCloud*2 is used to generate the C_{corr} cloud for the Iterative IPM from the second iteration onwards. The C_{corr} is assigned initially to the *input* cloud (line 1). The differences (*diff*s) between the points located in C_{in} and the points located in C_{pred} are obtained in line 3. The intuition is that if the *diff*s are repeatedly added to the *input* then the C_{pred} for the next input will gradually converge to the desired (target) shape C_{in}.

Algorithm 9.3: Iterative IPM

Input: *input*, C_{in}, number of iterations n
Output: C_{corr} cloud

```
1  counter ← 2;
2  C_corr ← input;
3  while counter < n do
4  |   G_in ← Process C_in using appropriate 3D surface representation;
5  |   E ← Set of springback prediction values obtained from applying classifier to
   |   G_in, each value in E is correlated to a point p in G_in;
6  |   C_pred ← GeneratePredictedCloud (E, G_in, input);
7  |   if C_pred ≠ C_in then
8  |   |   C_corr ← GenerateCorrectedCloud2(C_in, input, C_pred);
9  |   |   input ← C_corr;
10 |   end
11 end
12 return C_corr
```

Function GenerateCorrectedCloud2(C_{in}, *input*, C_{pred})

Input: C_{in}, *input*, C_{pred}
Output: C_{corr} cloud

```
1 C_corr ← input;
2 for all (p_in ∈ C_in), (p_pred ∈ C_pred), (p_corr ∈ C_corr) do
3 |   diff = p_in − p_pred;
4 |   p_corr = input + diff;
5 end
6 return C_corr
```

Table 9.1 presents a detailed example for the Iterative IPM. The iteration ID, the average predicted error (springback) e and the average difference *diff* between the *input* and the C_{pred} for a given shape were recoded for 6 iterations ($n = 6$). From the table it can be seen that the average error equalled the average absolute *diff* on the first iteration where the C_{in} cloud double as the *input*. From the example it can be seen that the difference *diff* gradually decreases and that the minimum *diff* was obtained on the fifth iteration with a difference of 0.016 (bold font). Thus, the *input* cloud arrived at on iteration 5 produce the"best" C_{corr} that could be used to form the given shape with minimum potential effect of springback.

TABLE 9.1: An example on the iterative IPM process for a given shape where the average predicted error (e) and the average absolute difference between the C_{pred} and the C_{in} (*diff*) are recorded for six iterations $n = 6$.

Iteration ID	e	*diff*
1	0.731	0.731
2	0.664	0.067
3	0.735	0.071
4	0.763	0.029
5	**0.747**	**0.016**
6	0.646	0.101

9.4 Experiments and Evaluation

A sequence of experiments were conducted to evaluate the iterative IPM process. The Iterative IPM settings were as follows.

1. To be consistent with the evaluation of the Single Pass IPM: (i) $|L| = |L_E| = 5$ using equal width discretisation and (ii) the LGM (level one) was used to generate a classifier.

2. Two grid sizes $d = 1$ mm and $d = 10$ mm. The first was used because this is the most appropriate resolution for manufacturing purposes and the second because earlier results indicated that this was best suited to the level one LGM technique.

3. The eight Gonzalo and Modified data sets were used: GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2.

4. It was considered that $n = 12$ was sufficient to obtain a deep insight in to the operation of the Iterative IPM in order to: (i) provide a sufficient analysis on the generated C_{pred} and C_{corr} clouds obtained by each iteration of the iterative IPM and (ii) to facilitate the identification of the "best" C_{corr}.

The results of each iteration of the Iterative IPM were recorded in terms of the average of *absolute diff* values between the C_{pred} and C_{in}, and the average springback distribution of the C_{pred} cloud. It should be noted that only the *magnitude* of the *diff* values was considered as the intuition is to see if the C_{pred} and C_{in} converge from each other and the absolute *diff* values converge to zeros as well.

The average absolute *diff* values and the springback distribution of the C_{pred} clouds for the GSV1 data set, using $d = 10$ mm and $d = 1$ mm are presented in Figures 9.3 and 9.4 respectively. Similarly, for the GSV2 data set, the average absolute *diff* values and the springback distribution of the C_{pred} clouds using $d = 10$ mm and $d = 1$ mm are presented in Figures 9.5 and 9.6 respectively. In the same manner the results obtained using GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2 data sets are presented in Figures 9.7, 9.8, 9.9, 9.10, 9.11, 9.12, 9.13, 9.14, 9.15, 9.16, 9.17, 9.18 respectively. From the figures, it can be seen that:

- For the first iteration, the average absolute *diff* values are similar to the average springback distributions since the *input* for the Iterative IPM in the first iteration is the desired shape itself C_{in} so the predicted error (springback) would be the difference between the C_{in} and C_{pred} as shown in the figures.

- The convergence of C_{pred} towards C_{in} can be clearly observed in all the datasets as the average absolute *diff* tends to be zero. Note that the actual springback distribution will not be zero as the error labels are associated with mean values.

- The overall behaviour of the springback distribution is totally related to the operation of the classifier and the nature of the discretisation (equal width discretisation in this case). Recall that the predicted error labels are characterised by their uneven distribution and this may have some implications on the mean values used to replace the labels. For instance, suppose one of the records was labelled with a label L_1 that had a mean value of v_1 in iteration i_1 and this label is changed in the next iteration i_2 to be L_2 which had a mean value of v_2 and if this happened for n records then the springback distribution would be affected by this change either in an increasing or decreasing manner.

- The fluctuations in the average absolute *diff* values (as in the case, for example of Figure 9.6) are explained by: (i) the uneven distribution of the labels, (ii) the dominant error (springback) distributed over the shape, (iii) the differences between the mean values of the error labels and (iv) the changing from one label to another during IPM iterations with respect to a single record which probably will not only affect the average springback distribution but also the overall average

absolute *diff* values. Despite the fluctuating behaviour of the average absolute *diff* values, the overall behaviour was tending to decrease throughout the IPM process.

In practice, it would be of interest to supply the manufacturer with the the "best" C_{corr} cloud rather than the one finally arrived at (there may be local minima). Therefore, Table 9.2 presents a summary related to the "best" iteration ID where the "best" C_{corr} cloud was obtained (from Figures 9.3 to 9.18) with respect to the eight datasets using $d = 10$ and $d = 1$. From the table it can be noted that:

- The maximum number of iterations required to obtain the "best" *input* using $d = 10$ and $d = 1$ was 9 iterations (GTV2) and 8 iterations (MTV2) respectively.
- The minimum number of iterations required to obtain the "best" *input* using $d = 10$ and $d = 1$ was 2 iterations (MSV2, MTV2 and GTV2).
- It was found that $d = 1$ mm required more iterations to obtain the "best" *input* (C_{corr} cloud) than $d = 10$ in five of the eight cases (GSV1, GSV2, GTV1, MSV2 and MTV2). The average number of iterations required to obtain the best *input* for $d = 10$ and $d = 1$ was 4.25 and 4.88 respectively and thus we can argue that there was no significant difference between using different grid sizes to obtain the best *input* with respect to the number of iterations required.
- The final average absolute *diff* obtained when $d = 1$ is always 0. It is therefore suggested $d = 1$ will produce slightly a more accurate end result (by small margin) than when $d = 10$ is used although $d = 10$ required fewer iterations and was more efficient in terms of run time (as will be demonstrated in Section 9.5).
- The material of the shape (steel or titanium) had no significant impact with respect to the iterative IPM process.

Finally, these results show that the iterative IPM would be able to supply manufacturers with a most suitable *input* cloud for the desired shape so as to serve to limit the effect of springback.

9.5 Run Time Analysis

This section presents the run time analysis for the IPM process using the two different grid sizes, $d = 10$ and $d = 1$, for the eight data sets GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1, MTV2. The experiments were carried out using a 2.7 GHz Intel Core i5 PC with 4 GB 1333 MHz DDR3 memory, running OS X 10.8.1 (12B19). The implementation of the IPM was conducted using the Java programming language. The run time for each iteration of the IPM includes the following:

1. All the preprocessing steps required to generate: (i) grids, (ii) the LGM representation and (iii) equal width discretisation using $|L| = |L_E| = 5$ and error calculation with respect to new input (C_{corr} cloud) for each iteration.

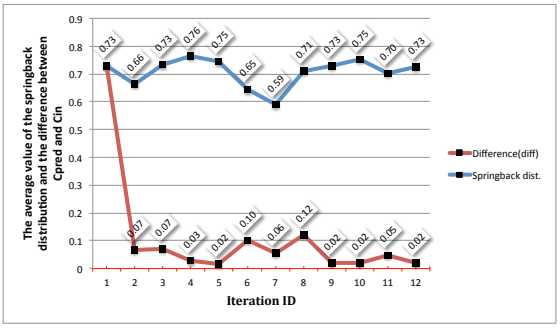


FIGURE 9.3: A average absolute *diff* values and the average springback distribution of C_{pred} for the GSV1 using $d = 10$ mm.

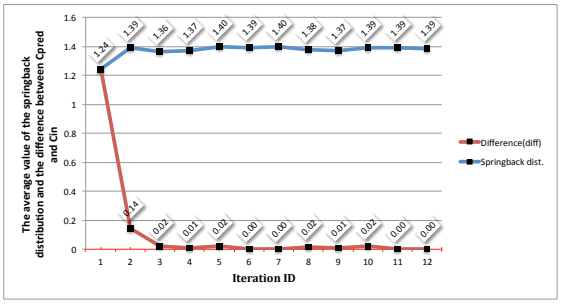


FIGURE 9.4: The average absolute *diff* values and the average springback distribution of C_{pred} for GSV1 using $d = 1$ mm.

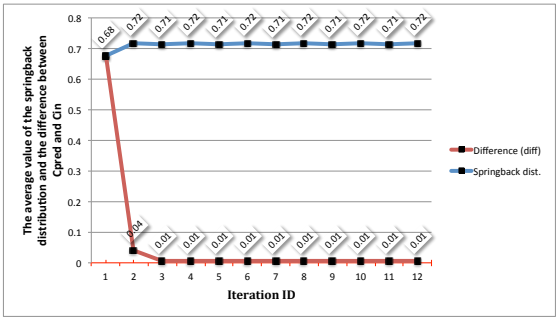


FIGURE 9.5: The average absolute *diff* values and the average springback distribution of C_{pred} for GSV2 using $d = 10$ mm.

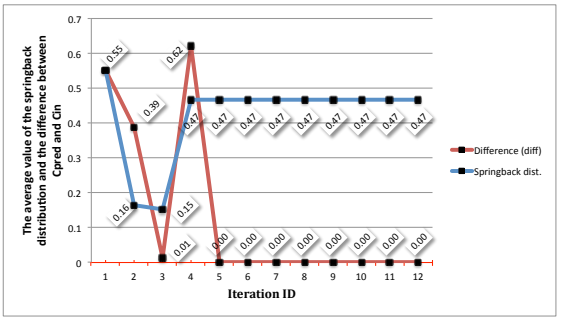


FIGURE 9.6: The average absolute *diff* values and the average springback distribution of C_{pred} for GSV2 using $d = 1$ mm.

TABLE 9.2: The best iteration ID, the average absolute *diff* and the springback distribution for the GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV1 datasets for $d = 10$ mm and $d = 1$ mm.

Datasets	d	best iteration ID	The average absolute *diff*
GSV1	d=10	5	0.02
	d=1	6	0.00
GSV2	d=10	3	0.01
	d=1	5	0.00
GTV1	d=10	3	0.00
	d=1	5	0.00
GTV2	d=10	9	0.00
	d=1	2	0.01
MSV1	d=10	4	0.00
	d=1	3	0.00
MSV2	d=10	2	0.00
	d=1	5	0.00
MTV1	d=10	6	0.01
	d=1	5	0.00
MTV2	d=10	2	0.00
	d=1	8	0.00
Average	d=10	4.25	0.01
	d=1	4.88	0.00

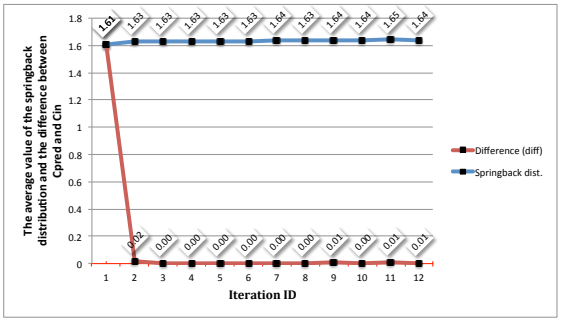


FIGURE 9.7: The average absolute *diff* values and the average springback distribution of C_{pred} for GTV1 using $d = 10$ mm.

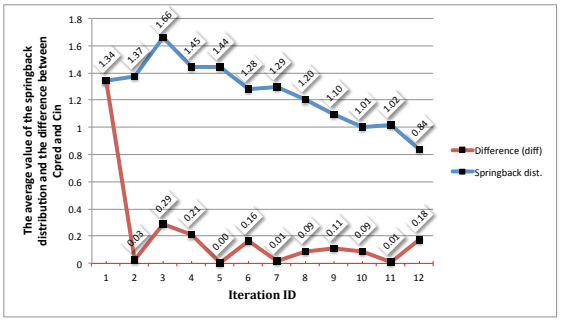


FIGURE 9.8: The average absolute *diff* values and the average springback distribution of C_{pred} for GTV1 using $d = 1$ mm.

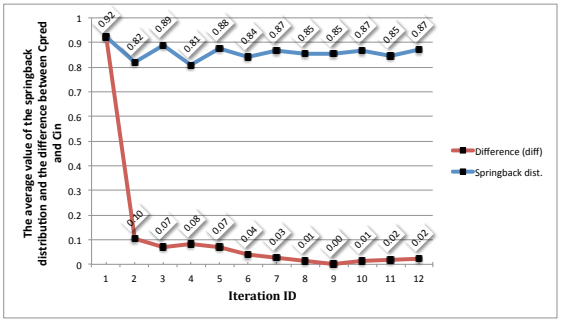


FIGURE 9.9: The average absolute *diff* values and the average springback distribution of C_{pred} for GTV2 using $d = 10$ mm.

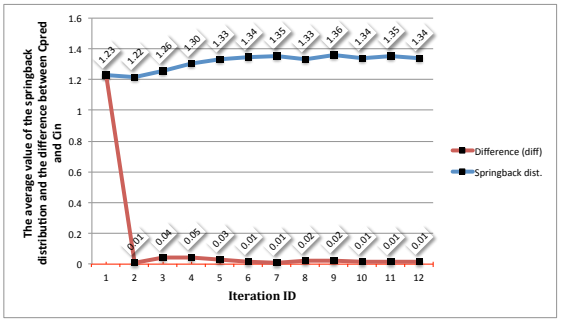


FIGURE 9.10: The average absolute *diff* values and the average springback distribution of C_{pred} for GTV2 using $d = 1$ mm.

2. The prediction of error (springback) values using the predefined classifier.

3. The generation of the C_{pred} and C_{corr} clouds for 12 iterations based on the predicted value of the error (springback).

Figures 9.19 and 9.20 present the run time analysis (in seconds) for the eight data sets using $d = 10$ and $d = 1$ respectively with respect to $n = 12$ iterations. From the figures it can be seen that the IPM required more run time for $d = 1$ than for $d = 10$ and this is clearly because small grid size generates more LGMs and consequently requires more processing than larger grid size. However, since the results presented earlier in Section 9.4 indicated that there was no significant difference between using $d = 10$ and $d = 1$ with respect to the iterative IPM process then the run time may be improved if $d = 10$ is used, although for manufacturing purposes an alternative grid size may be desirable.

9.6 Summary

This chapter has presented two Intelligent Process Models (IPMs) which can be used to reduce the springback effect in the context of AISF. The IPM process assumes the

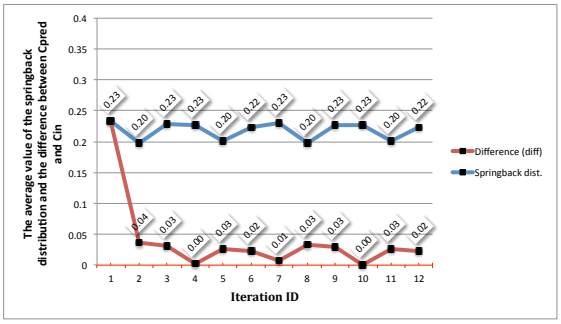


FIGURE 9.11: The average absolute *diff* values and the average springback distribution of C_{pred} for MSV1 using $d = 10$ mm.

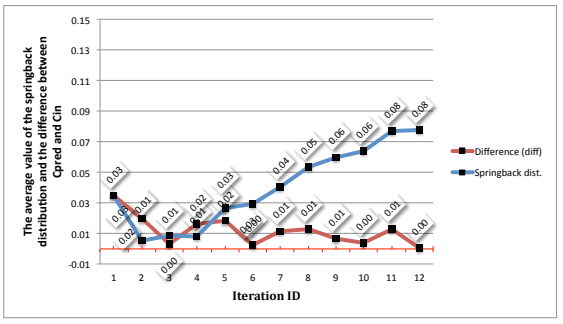


FIGURE 9.12: The average absolute *diff* values and the average springback distribution of C_{pred} for MSV1 using $d = 1$ mm.

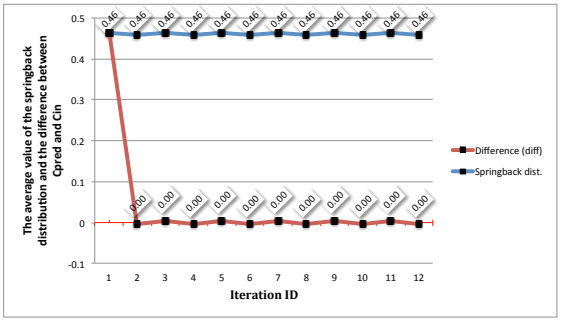


FIGURE 9.13: The average absolute *diff* values and the average springback distribution of C_{pred} for MSV2 using $d = 10$ mm.

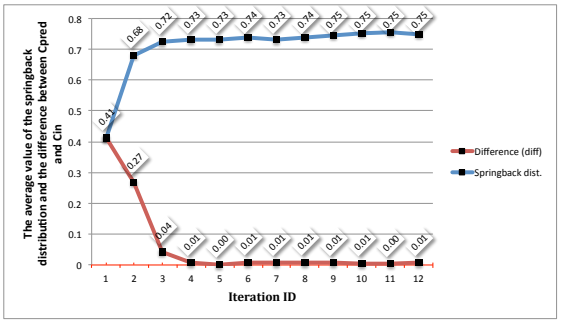


FIGURE 9.14: The average absolute *diff* values and the average springback distribution of C_{pred} for MSV2 using $d = 1$ mm.

existence of an appropriately trained classifier, of the form considered earlier in this thesis, and is designed to take an *input* shape (cloud) C_{in} (A "CAD Shape") and produce a corrected shape (cloud) C_{corr} that mitigates against the effect of springback. Two variations of the IPM process were produced: (i) Single Pass and (ii) Iterative. The first operated in the same manner as the experimental set up presented in Chapter 8. The second, as the name suggested, operated in an iterative manner. The Single Pass IPM was considered in depth in Chapter 8, thus most of the content of this chapter was directed at the Iterative IPM. The conducted evaluation of the Iterative IPM process demonstrated that the process can be successfully applied to a given *input* shape description to produce a corrected description ready for manufacture. Though the Single Pass IPM was able to generate a corrected cloud for a given shape, there is actually no guarantee that this is the "best" corrected cloud (because no further analysis or investigation was conducted). Therefore, using the Iterative IPM can perhaps provide some guarantee that the C_{pred} cloud will gradually converge to C_{in} and it will end by providing the end user with a "best" *input* C_{corr} for the desired shape description. A summary for the entire work described in this thesis, along with some suggested future work, is presented in the following chapter.

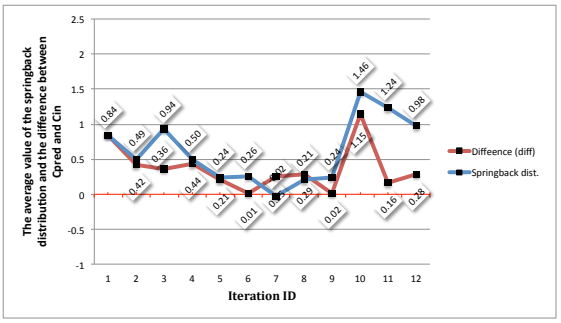


FIGURE 9.15: The average absolute *diff* values and the average springback distribution of C_{pred} for MTV1 using $d = 10$ mm.

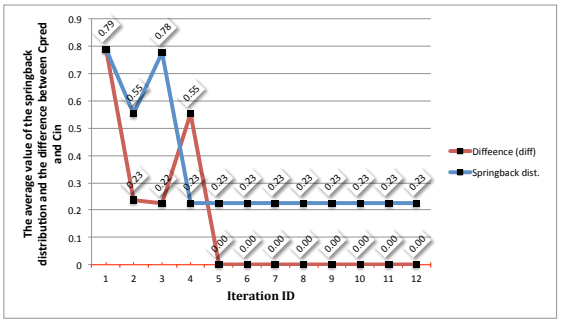


FIGURE 9.16: The average absolute *diff* values and the average springback distribution of C_{pred} for MTV1 using $d = 1$ mm.

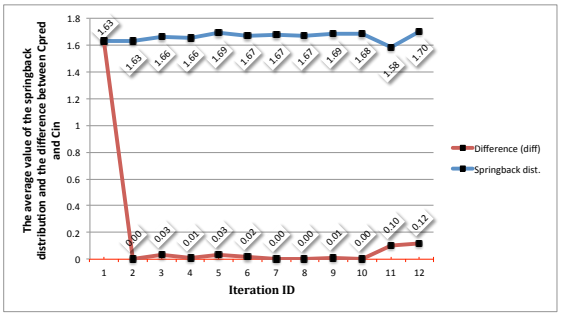


FIGURE 9.17: The average absolute *diff* values and the average springback distribution of C_{pred} for MTV2 using $d = 10$ mm.

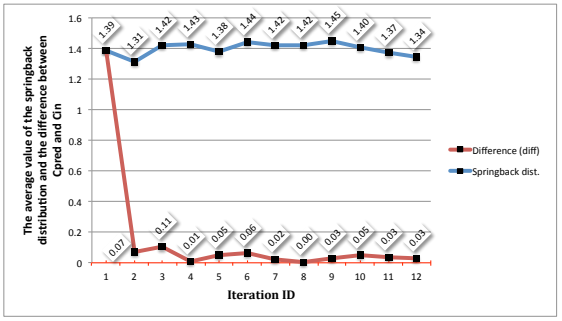


FIGURE 9.18: The average absolute *diff* values and the average springback distribution of C_{pred} for MTV2 using $d = 1$ mm.

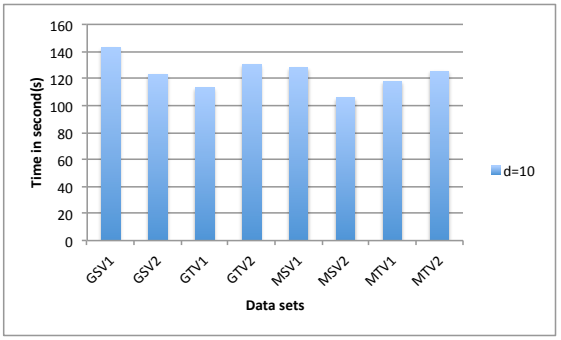


FIGURE 9.19: The run time analysis (in seconds) for the eight data sets using $d = 10$ mm with respect to $n = 12$ iterations.

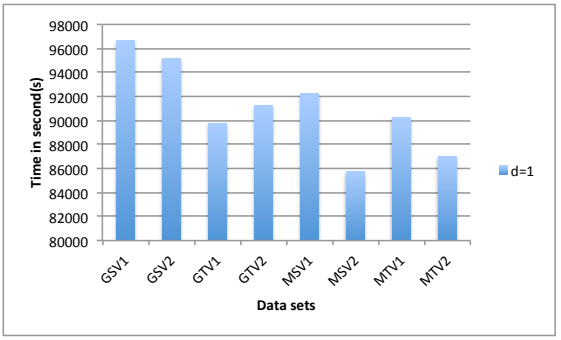


FIGURE 9.20: The run time analysis (in seconds) for the eight data sets using $d = 1$ mm with respect to $n = 12$ iterations.

Chapter 10

Conclusion and Future Research Works

10.1 Inroduction

This concluding chapter presents an overall summary of the work described in this thesis along with the main findings and contributions. This chapter also provides some suggestions for future work. The chapter is organised as follows. In Section 10.2 an overall summary of the thesis is presented. The main findings and contributions are reported in Section 10.3. Finally, suggested ideas for future work are presented Section 10.4 in the context of further potential research based on the work described in this thesis.

10.2 Summary

This section presents an overall summary of the work presented in this thesis. The work was directed at the use of classification techniques for springback detection; the deformation that occurs as a result of the application of a sheet metal forming process (specifically AISF). The main challenge was the nature of the springback phenomena especially the uneven distribution of springback with respect to local geometries. The work can best be described in terms of the following three categories: (i) preprocessing, (ii) 3D surface representation and (iii) classifier generation. With respect to preprocessing a grid representation mechanism was proposed with which to compare point clouds describing the shape of a given object. In the context of AISF the clouds of interest were the C_{in} cloud (describing the desired shape before manufacture) and the C_{out} cloud (describing the shape of the manufactured object). The grid representation was proposed so as to provide for an initial generic representation for both clouds because: (i) it is compatible with many forms of higher level representation, (ii) it facilitates the comparison of the clouds so that springback calculations can be performed easily and (iii) it can be used as a base representation from which higher level representations can

be generated. An important element of the preprocessing was springback calculation, required for classification purposes, which was thus described in detail.

Afterwards, three different surface representation techniques were proposed to capture the local geometric nature for a given shape in a manner compatible with effective classifier generation. The first 3D representation technique, LGM, was founded on the Local Binary Pattern (LBP) concept where the grid center points were described in terms of their local neighbourhoods. Three different variations of the LGM technique were proposed with respect to differing neighbourhood levels: (i) Level one LGM, (ii) Level two LGM and (ii) composite LGM where a combination of level one and two were used. Although the reported results indicated that there was no significant difference between them, the performance of the composite LGM tended to produce a better performance.

The second technique, LDM, was founded on the observation that springback magnitude is influenced by distance from critical features, edges and corners. The reported results showed that the LDM representation was insufficient, on its own, to adequately describe the 3D surfaces of interest in the context of classification. However, the reported results did indicate that the combination of LDM and LGM produced a better performance than when either is used in isolation.

The third technique was the PS technique that was founded on a "space linearisation" concept where the 3D surface was represented in terms of "curves". All the techniques were realised using the proposed RASP framework. The reported performance results indicated that the PS representation was the "best" 3D surface representation technique in terms of classification accuracy and AUC results. Moreover, the conducted statistical evaluation confirmed that the performance of the proposed techniques were statistically significant and also ensured that the performance of the PS was the best overall technique.

For classifier generation a binary valued feature vector representation was used with respect to all the proposed surface representation technique. Individual values were ranged, including the springback values in the case of training sets, using predefined labels sets. For the ranging, equal frequency binning was adopted, as opposed to equal size binning, to eliminate the effect of unbalanced distributions. A wide range of classification algorithms were considered, namely: (i) the C4.5 decision tree algorithm, (ii) Naive Bayes, (iii) the JRIP classification rule learner, (iv) the PART rule based decision tree, (v) a Neural Network (NN) technique and (vi) k-NN. The recorded evaluation indicated that there was no significant difference between the operation of these classifier generators.

The thesis also reported on the practical application of the techniques in the context of generating corrected clouds to be manufactured. By manufacturing pieces using the corrected clouds it was possible to demonstrate that as a result of springback prediction "better" shapes could be manufactured. The prediction process was packaged into what

was termed an Intelligent Process Model (IPM). Two variations of this were suggested, a single pass and an iterative IPM.

10.3 Main Findings and Contributions

This section revisits the research question and associated research issues presented in Chapter 1 (Section 1.3) and describes how each was resolved in terms of a set of "main findings". The section is organised by considering each of the identified research issues in turn as follows.

1. *Training Set Generation.* The challenge of how best to generate the required training data was resolved by first converting the input clouds into a grid representation which allowed for comparison of two surfaces. By comparing the two surfaces, and using normal calculations, it was possible to determine the springback values associated with individual grid squares. This then provided for a generic format in which to present training data. In total eight different raw data sets were used.

   An issue was the optimum value for d the size of the grid representation, to be used. The conducted experimental analysis established that there was no best overall value for d, but that each technique had a specific best value associated with it: $d = 10$ mm was found to be the best grid size for the LGM techniques, (ii) $d = 2.5$ mm was was found to be the best grid size for the LDM technique and (iii) $d = 5$ mm was was found to be the best grid size for the PS techniques. By using different values for d, a large collection of training data were produced, and successfully utilised with respect to the evaluation of the different surface representations. As noted above, the advantage offered by the grid representation was its generic nature and its compatibility with "higher level" representations. It is difficult to conceive of alternative formats for the training data that offer the same advantages and thus it is suggested here that the grid representation is the most appropriate format for training set generation (more details can be found in Chapter 3).

2. *3D Surface Representation.* The main challenge for this thesis was to determine the most appropriate 3D surface representation in order to capture the most important local geometrical features (in the context of sheet metal forming) and consequently facilitate the operation of the classification process. This was addressed through presenting three different surface representation techniques based on three different concepts: (i) the LGM technique was founded on representing the geometrical information in terms of the local surrounding neighbourhood, (ii) the LDM technique was founded on representing a 3D surface in terms of its "critical" features and (iii) the PS technique was founded on the idea of representing a 3D surface in terms of "curves". The three techniques were presented in Chapters 4, 5 and 6 respectively. An extensive evaluation of the proposed techniques

was conducted. Two types of classifier performance experiments were conducted: (i) classifiers trained and tested on the same data and (ii) classifiers trained and tested on different data. The evaluation of the proposed techniques indicated that the PS techniques produced the best overall classification performances in that it outperformed the other techniques in terms of accuracy and AUC. In order to significantly differentiate between the operation of the proposed techniques, a statistical evaluation based on the Friedman and Nemenyi statistical tests was performed. This also confirm the superiority of the PS techniques with respect to the other techniques considered (more details can be found in Chapter 7).

3. *Best Classification Technique.* The challenge of identifying the most suitable classification techniques with respect to each proposed representation technique was resolved by selecting a number of popular classification techniques: (i) C4.5, (ii) Bayes, (iii) JRIP, (iv) PART, (v) Neural Network and (iv) k-Nearest Neighbour (k-NN). The generated classifiers were then incorporated into the IPM concept to predict the errors (springback) and generate corrected clouds. The obtained results indicated that there were no significant difference between the different supervised classification technique considered, but the C4.5 technique was selected to be the most suitable when using the LGM and LDM techniques due to its simplicity and powerful interpretation capabilities, especially for non experts from other fields. The last technique (k-NN) was found to be the most suitable classification technique with respect to the PS technique. Despite the advantages offered by the *labelling* concept, label size was an issue. The uneven distribution of the springback phenomena over shapes was the main reasons behind this issue. However, to eliminate the effect of springback distribution, *equal frequency* discretising was adopted.

4. *Corrected Input Generation.* The challenge of how the predicted errors can best be translated into an acceptable format for the manufacturing process was resolved by applying the predicted errors in the reverse direction to generate a "corrected cloud". Practically, the corrected cloud was successfully applied in a real manufacturing environment and the effect of the springback in the produced parts was minimised (more details can be found in Chapter 8). However, the challenge to obtain the most "optimal" corrected cloud was resolved by the proposed iterative IPM process where the corrected cloud was repeatedly generated until an optimal cloud was obtained (more details can be found in Chapter 9).

Returning to the main research question *"How best can 3D surfaces be represented to reflect local geometrical information according to certain feature(s) of interest so that classification techniques can be applied effectively?"*. The techniques presented in Chapters 4, 5 and 6 respectively, clearly indicated that local geometrical information can be represented effectively using any of these techniques. However, the PS technique was found to outperform the other techniques. The statistical evaluation confirmed that

the PS technique was the "best" techniques. However, all the proposed representations could easily and effectively be used to generate a classifier that served to predict the feature of interest (springback value in our case). It was also demonstrated that the springback predictions could be successfully utilized to manufacture better parts.

The main contributions of the work described in this thesis may be summarised as follows:

1. A grid representation which provides for the comparison of 3D surfaces.
2. A springback calculation mechanism to identify the springback values between the desired and the actual formed shape (given appropriate before and after data).
3. The RASP framework to support the classifier generation process.
4. The LGM surface representation technique that was used to describe 3D surfaces in terms of local neighbourhoods.
5. The LDM surface representation technique that was used to represent 3D surfaces in terms of proximity to the nearest corner or edge.
6. The PS surface representation technique that was used to describe 3D surfaces in terms of point series curves.
7. A statistical comparison to identify the significant difference between the proposed techniques.
8. A mechanism to generate corrected clouds based on the predicted values.
9. The concept of an Intelligent Process Model that combines the proposed techniques with the corrected cloud generation mechanism into a single process with respect to sheet metal forming.

10.4 Future Work

The work presented in this thesis has demonstrated that, in the context of sheet metal forming, springback can be effectively predicted and utilised. Despite the successful results produced, enhancements and improvements can be envisioned. This concluding section suggests some potential areas for future work as follows:

1. **More 3D Representation Techniques.** A standard feature vector format was adopted with respect to the work described in this thesis because of its simplicity and ease of use. It may be interesting to investigate alternative 3D surface representations such as graph or tree based representations. Of course using graph or tree representations would also consequently require the usage of alternative classification techniques possibly founded on ideas concerning graph mining.

2. **Alternative Applications.** The proposed techniques were successfully evaluated in the context of the AISF process. It would be worth investigating the applicability of these techniques with respect to other areas. For instance, the classification of satellite images to identify the diverse topography (mountains, hills, valleys, trees and houses) of a given image. Such an investigation might then serve to illustrate the expected generic nature of the proposed surface representation techniques.

3. **Visualisation and Reasoning.** The ability to generate a visualisation for the shapes resulting from the application of "corrected" clouds, and what it would look like if it was manufactured, would be of great help to practitioners since this would: (i) provide valuable insight on the predicted errors, (ii) provide for a visual evaluation of the actual shapes to be produced so as to a void needless manufacturing and (iii) provide more control over the proposed IPM process so that the process can be stopped at a specific iteration when certain industrial requirements are met, thus providing for a great degree of flexibility with respect to the IPM process. Moreover, if the visualisation is combined with an explanation of why a particular predicted springback occurred in a particular region this would also provide for additional confidence in the technique.

4. **Enhanced Version of Corrected Cloud.** According to [10], there are difficulties in the manufacturing environments on how to utilise the predicted value of the springback effectively to produce the desired shape as accurately as possible. These difficulties with respect to the work described in this thesis may be caused by the generic way that corrections are applied; namely reversing the detected springback along the normal. However, it is conjectured that a better way of doing this might be to weight the springback magnitude. It is also conjectured that this weighting will not be uniform a cross a given shape but will instead be dependent on local geometries. It is therefore suggest that a data mining approach may be used to identify what these weightings might be.

5. **Further Application to Metal Manufacturing Process.** Though the proposed techniques were successfully evaluated in the context of the AISF process using the Gonzalo and the Modified pyramids made of steel and titanium, it would be worthwhile to investigate the applicability of these techniques with respect to: (i) other types of metals such as inconel (a mixture of chromium and iron utilized in the aeroplane industry) and (i) other manufacturing parameters along with the geometry of a given shape such as the size of tool head and the sheet thickness.

Overall the work on the use of 3D representation and prediction techniques in the context of sheet metal forming, as presented in this thesis, has produced some interesting outcomes and provided a sound foundation for future work.

Appendix A

Error Visualisation for Gonzalo and Modified Pyramids

This appendix presents a visualisation of the error (springback) distribution for the Gonzalo and Modified pyramids with respect to the eight data sets used in this thesis: GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2; using a range of grid size $d = 2.5, 5, 10, 15, 20$. The label set size, used for the purpose of the visualisation, was $|L| = 7$. The distribution was reported in term of: (i) the *absolute* springback (error) and (i) the *directed* springback (error). The springback distribution for each of the data sets for the *steel* manufactured shapes are presented first with respect to absolute and directed springback distribution respectively, then for each of the data sets for the *titanium* manufactured shapes. The figures are as follows.

1. Figures A.2 and A.3 present the *absolute* springback distribution and the *directed* springback distribution respectively for the GSV1 shape using different grid sizes.
2. Figures A.4 and A.5 present the *absolute* springback distribution and the *directed* springback distribution respectively for the GSV2 shape using different grid sizes respectively.
3. Figures A.6 and A.7 present the *absolute* springback distribution and the *directed* springback distribution respectively for the MSV1 shape using different grid sizes respectively.
4. Figures A.8 and A.9 present the *absolute* springback distribution and the *directed* springback distribution respectively for the MSV2 shape using different grid sizes respectively.
5. Figures A.10 and A.11 present the *absolute* springback distribution and the *directed* springback distribution respectively for the GTV1 shape using different grid sizes respectively.

6. Figures A.12 and A.13 present the *absolute* springback distribution and the *directed* springback distribution respectively for the GTV2 shape using different grid sizes respectively.

7. Figures A.14 and A.15 present the *absolute* springback distribution and the *directed* springback distribution respectively for the MTV1 shape using different grid sizes respectively.

8. Figures A.16 and A.17 present the *absolute* springback distribution and the *directed* springback distribution respectively for the MTV2 shape using different grid sizes respectively.

The colour code for the error distribution with respect to both the *directed* and undirected error visualisation is presented in Figure A.1. In the case of the *absolute* error distribution the colour turns from yellow to green as the error becomes larger; while in the case of the *directed* error visualisation the largest error values occur when the $errorscale = 1$ and 7 (the two ends) and the minimum error value occurs when the error scale = 4 (in the centre of the error range).

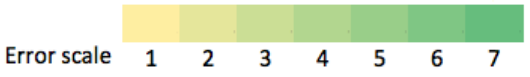
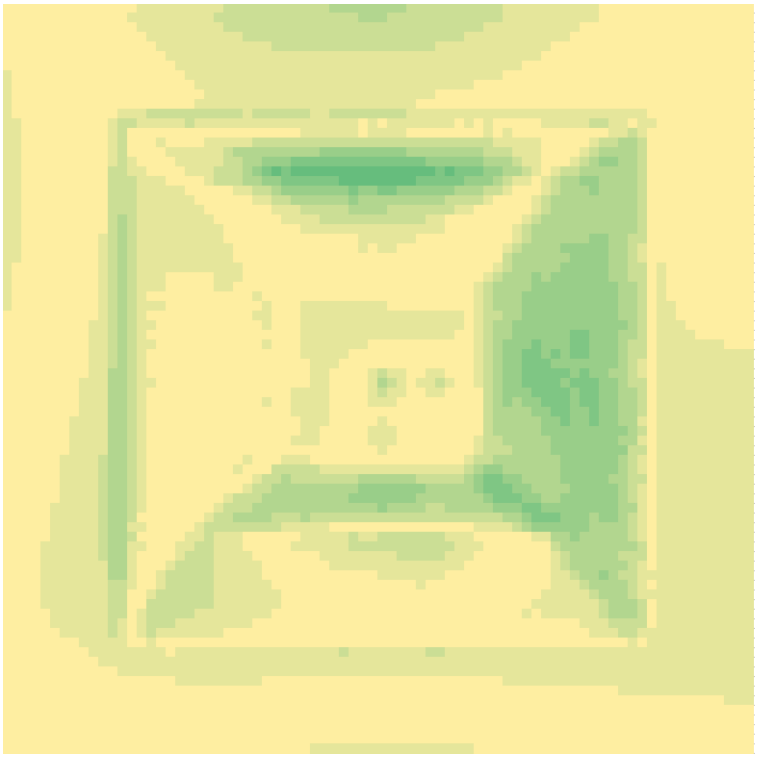
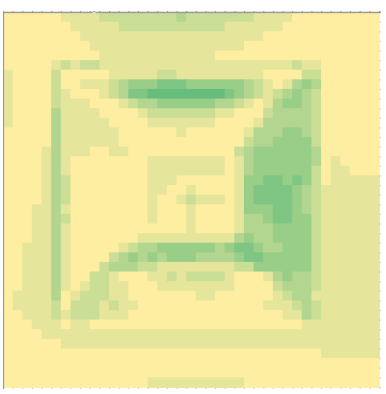


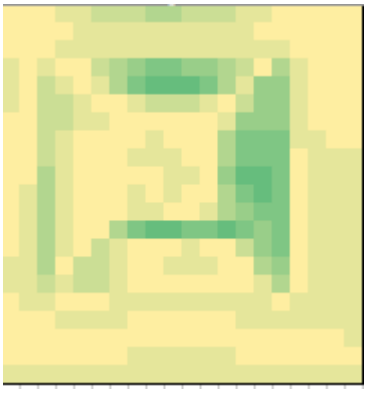
FIGURE A.1: The Error scale used to describe both the *absolute* and the *directed* error (springback) distribution for the Gonzalo and Modified pyramid shapes.



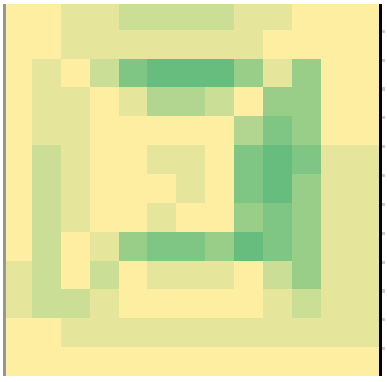
(a) $GSV1$ $d = 2.5$



(b) $GSV1$ $d = 5$



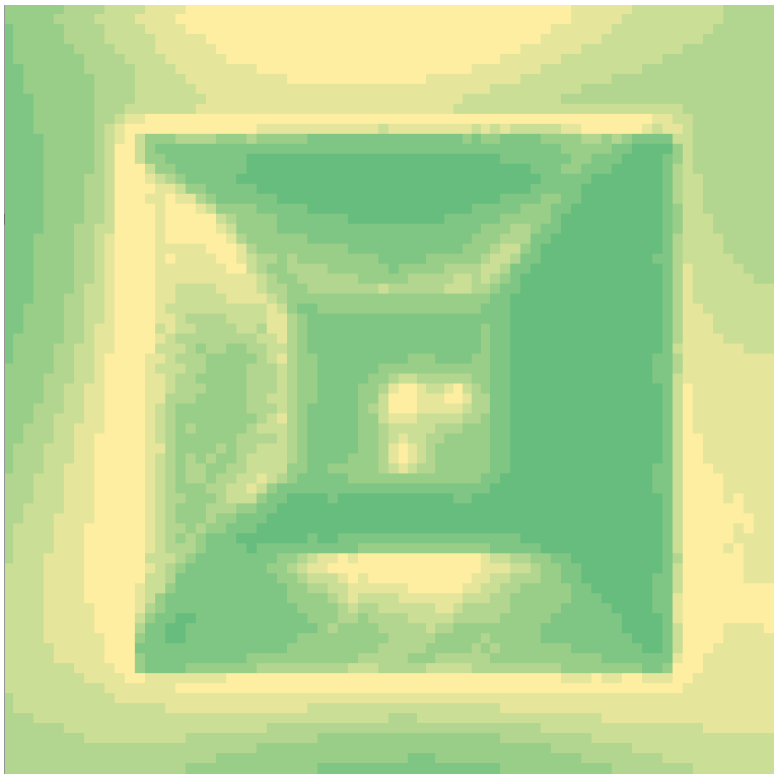
(c) $GSV1$ $d = 10$



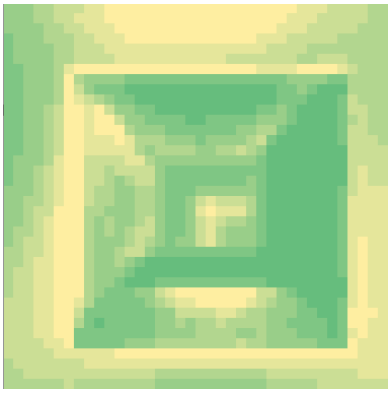
(d) $GSV1$ $d = 15$

(e) $GSV1$ $d = 20$

FIGURE A.2: The *absolute* error visualisation for the Gonzalo Steel V1 (GSV1) pyramid for different grid sizes (d).

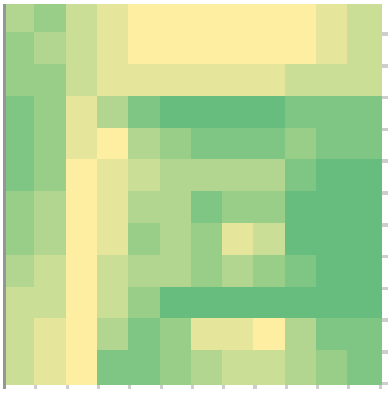


(a) $GSV1$ $d = 2.5$



(b) $GSV1$ $d = 5$

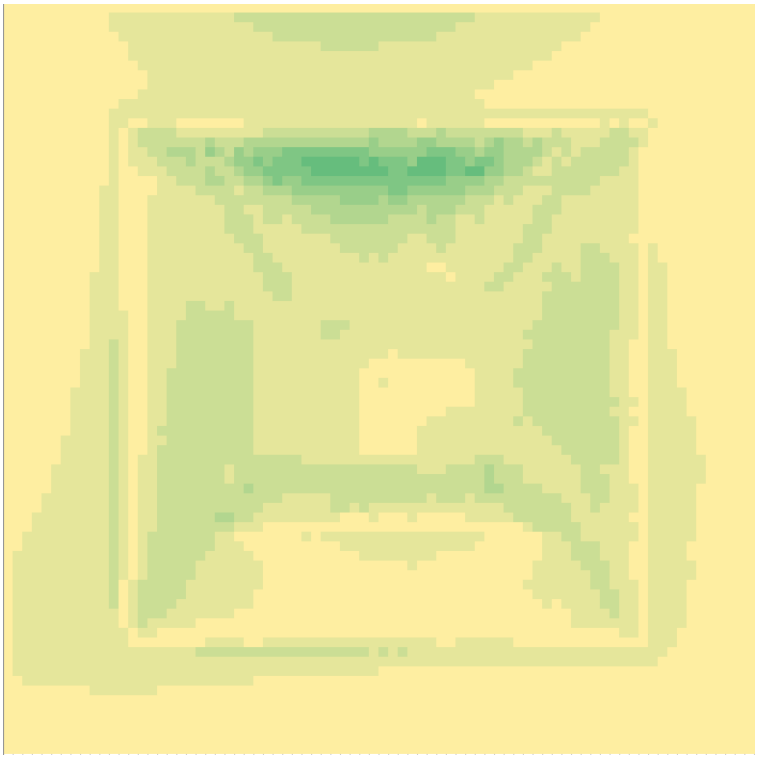
(c) $GSV1$ $d = 10$



(d) $GSV1$ $d = 15$

(e) $GSV1$ $d = 20$

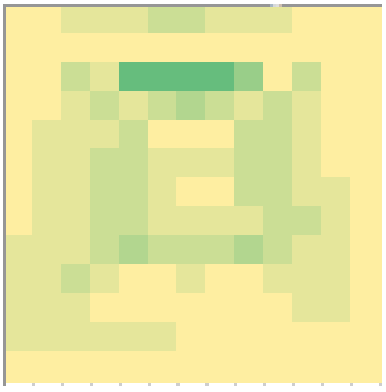
FIGURE A.3: The *directed* error visualisation results for Gonzalo Steel V1 (GSV1) pyramid for different grid size (d).



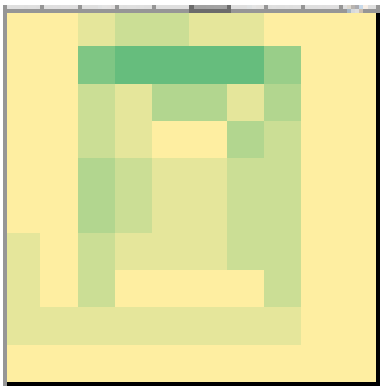
(a) $GSV2$ $d = 2.5$

(b) $GSV2$ $d = 5$

(c) $GSV2$ $d = 10$

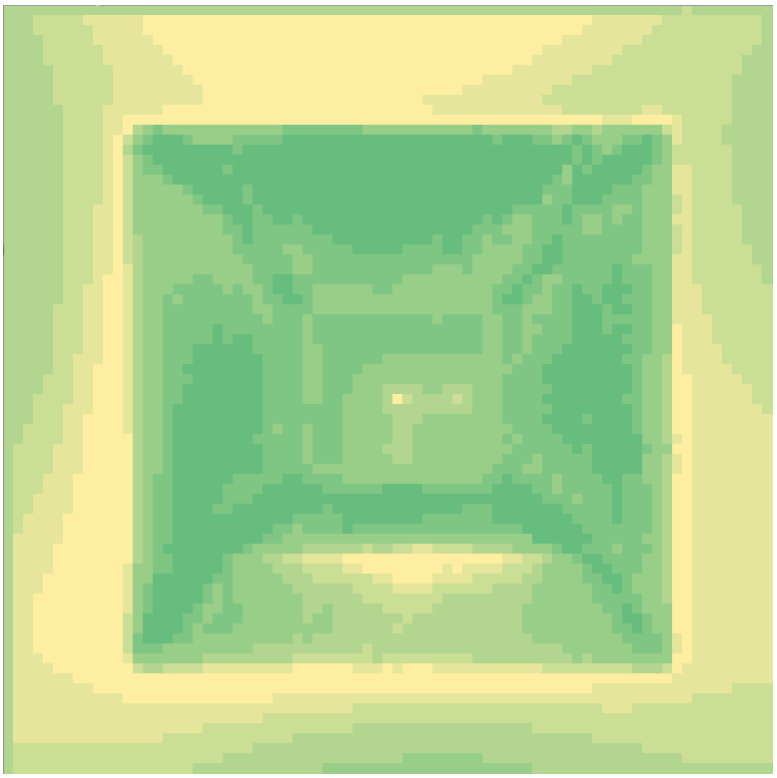


(d) $GSV2$ $d = 15$

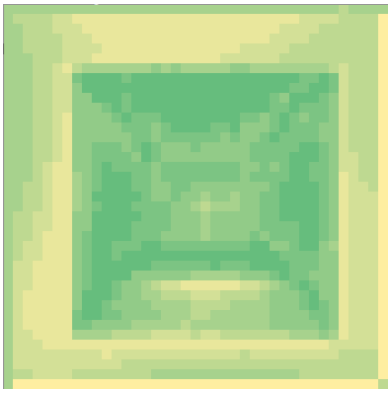


(e) $GSV2$ $d = 20$

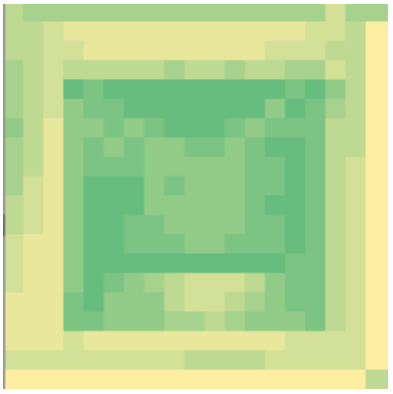
FIGURE A.4: The *absolute* error visualisation results for the Gonzalo Steel V2 (GSV2) pyramid for different grid sizes (d).



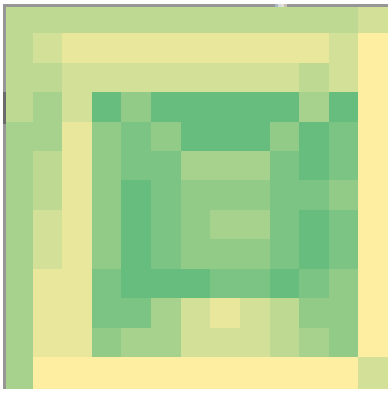
(a) $GSV2$ $d = 2.5$



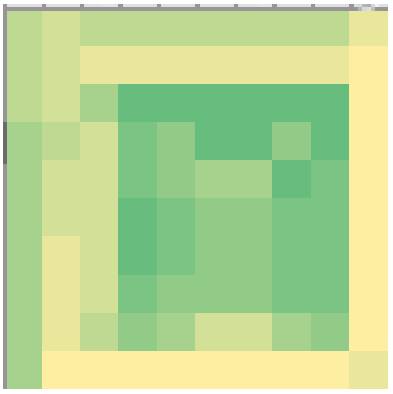
(b) $GSV2$ $d = 5$



(c) $GSV2$ $d = 10$

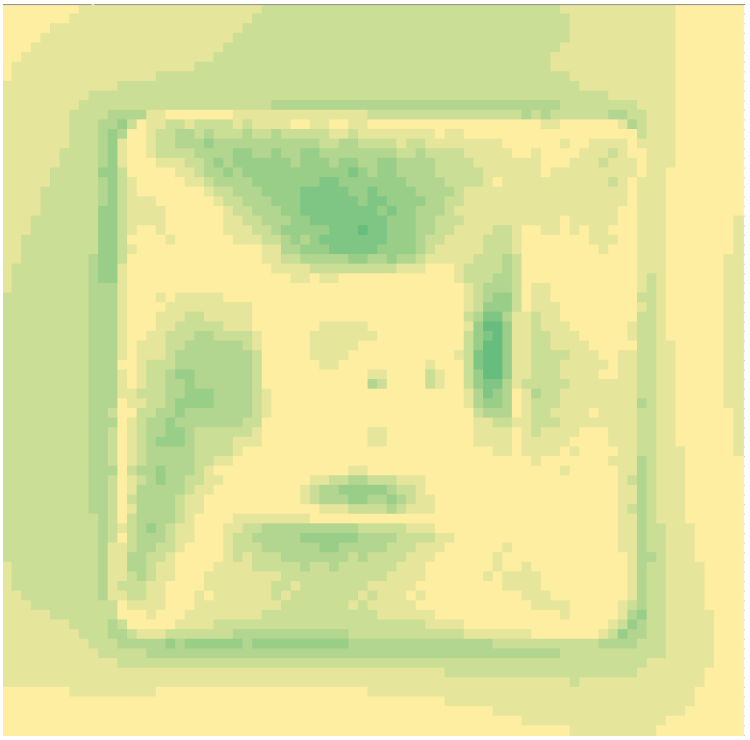


(d) $GSV2$ $d = 15$

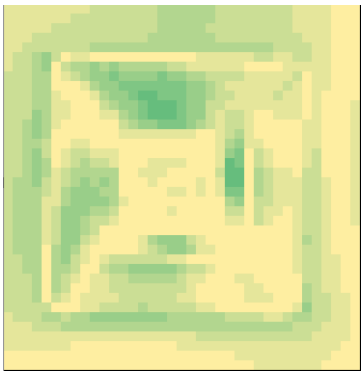


(e) $GSV2$ $d = 20$

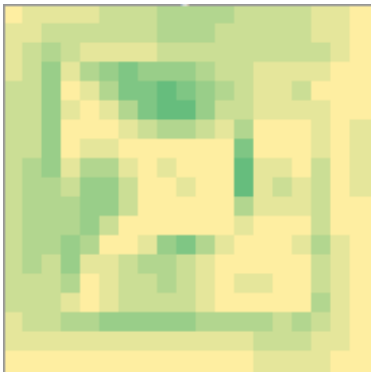
FIGURE A.5: The *directed* error visualisation results for the Gonzalo Steel V2 (GSV2) pyramid for different grid sizes (d).



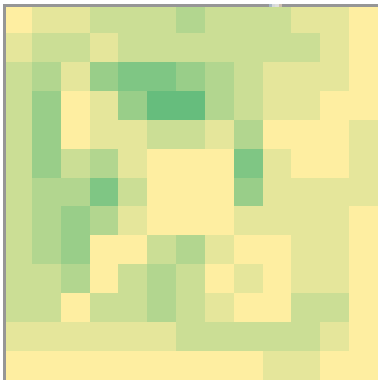
(a) $MSV1$ $d = 2.5$



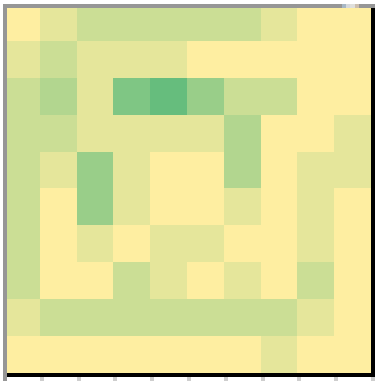
(b) $MSV1$ $d = 5$



(c) $MSV1$ $d = 10$

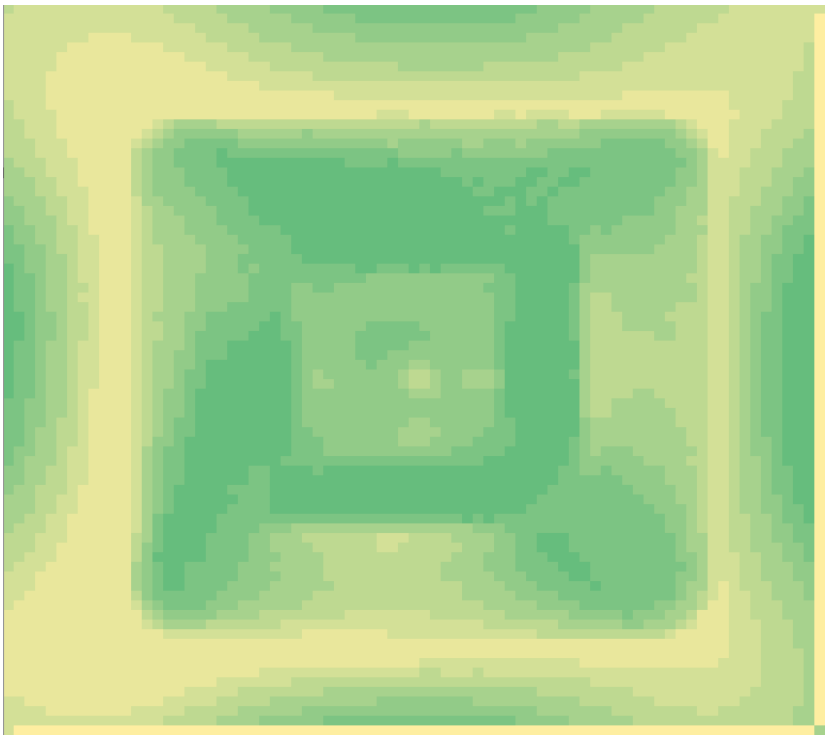


(d) $MSV1$ $d = 15$

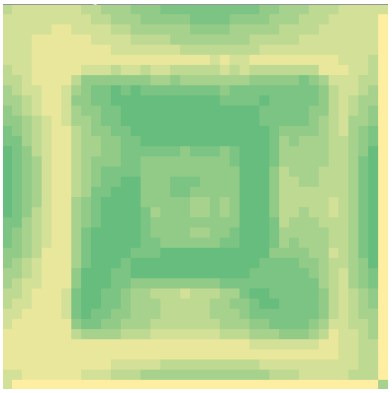


(e) $MSV1$ $d = 20$

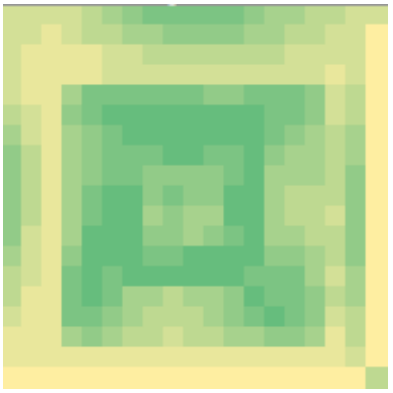
FIGURE A.6: The *absolute* error visualisation results for the Modified Steel V1 (MSV1) pyramid for different grid sizes (d).



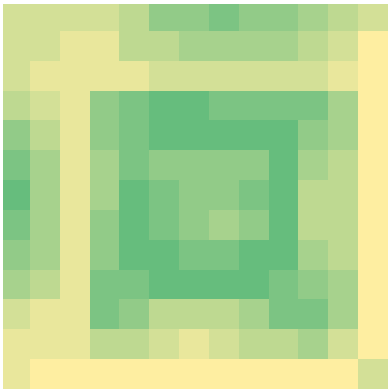
(a) $MSV1$ $d = 2.5$



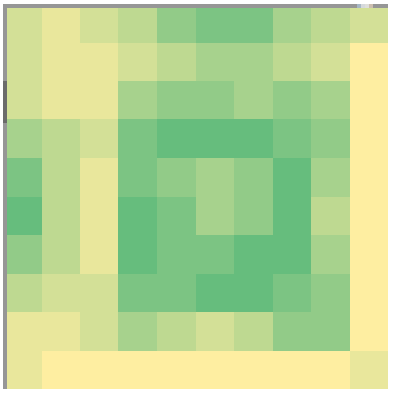
(b) $MSV1$ $d = 5$



(c) $MSV1$ $d = 10$

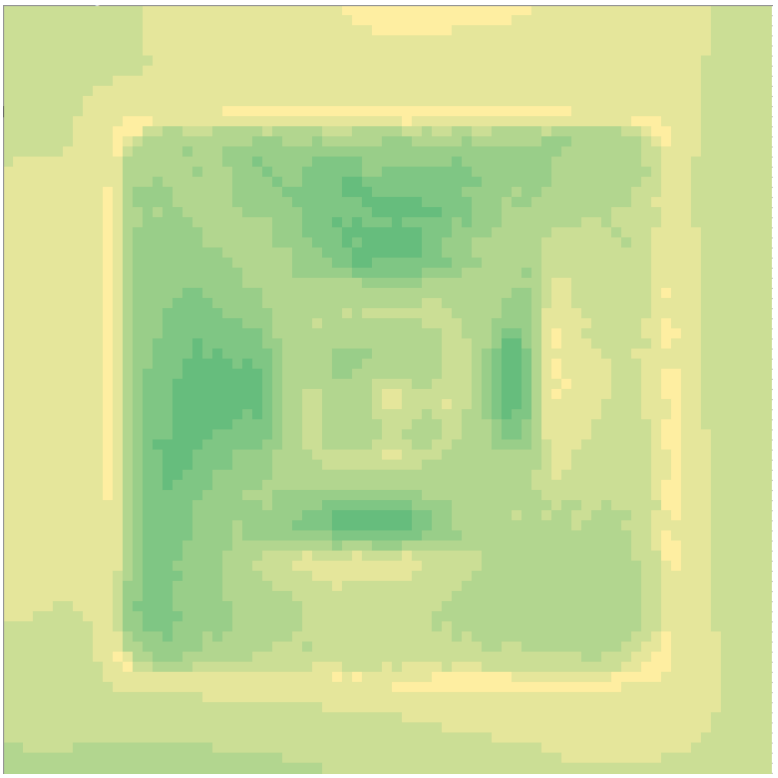


(d) $MSV1$ $d = 15$

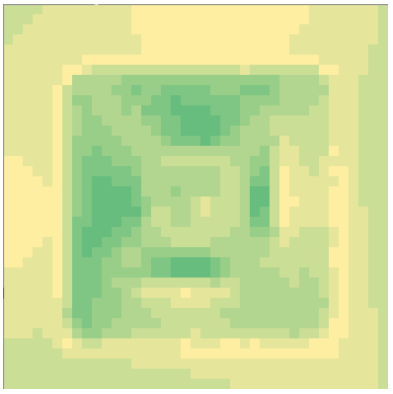


(e) $MSV1$ $d = 20$

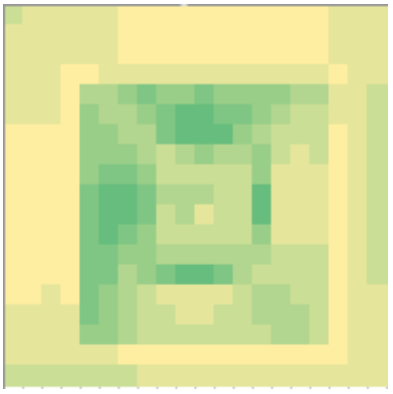
FIGURE A.7: The *directed* error visualisation results for the Modified Steel V1 (MSV1) pyramid for different grid sizes (d).



(a) $MSV2$ $d = 2.5$

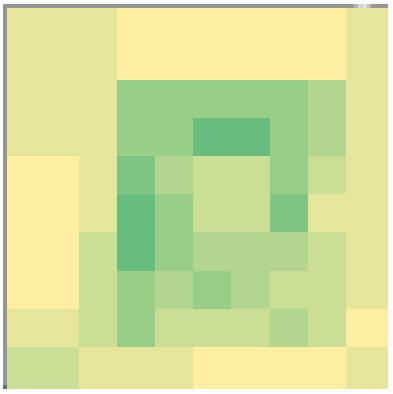


(b) $MSV2$ $d = 5$



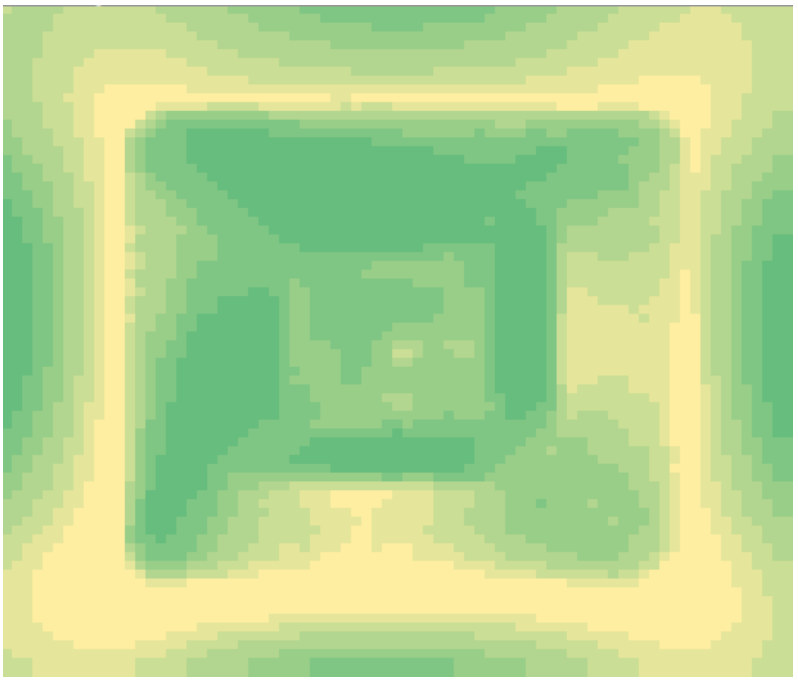
(c) $MSV2$ $d = 10$

(d) $MSV2$ $d = 15$

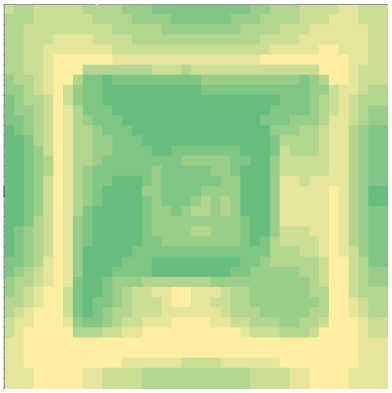


(e) $MSV2$ $d = 20$

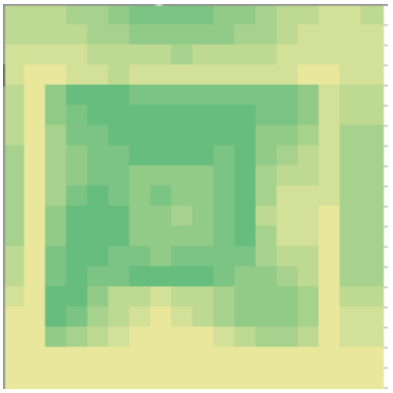
FIGURE A.8: The *absolute* error visualisation results for the Modified Steel V2 (MSV2) pyramid for different grid sizes (d).



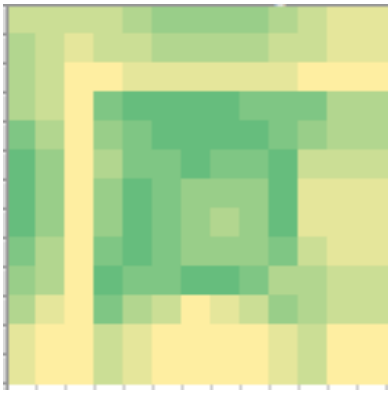
(a) $MSV2$ $d = 2.5$



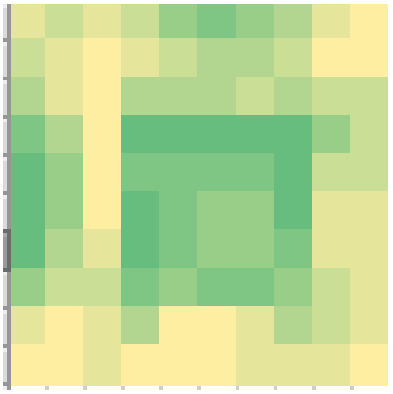
(b) $MSV2$ $d = 5$



(c) $MSV2$ $d = 10$

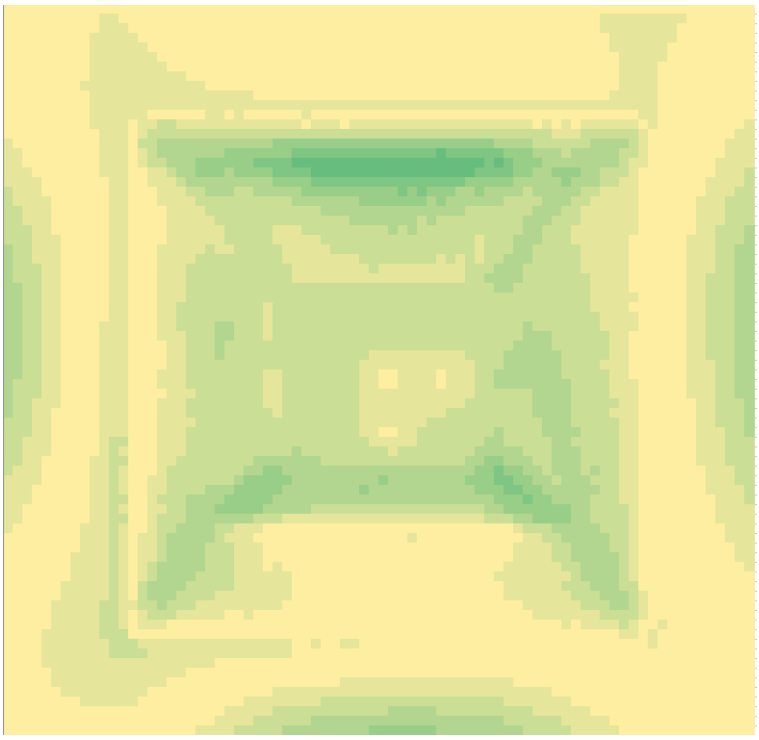


(d) $MSV2$ $d = 15$

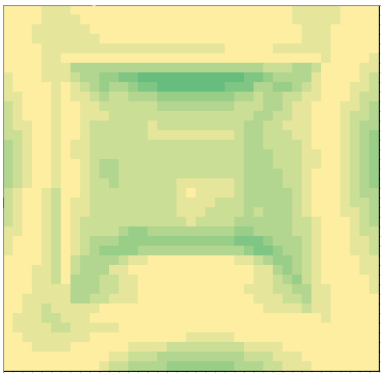


(e) $MSV2$ $d = 20$

FIGURE A.9: The *directed* error visualisation results for the Modified Steel V2 (MSV2) pyramid for different grid sizes (d).

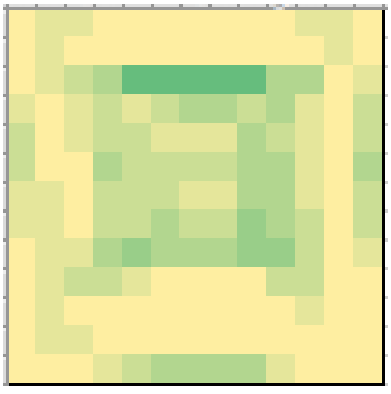


(a) $GTV1$ $d = 2.5$

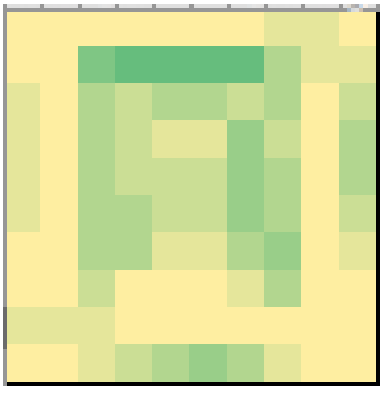


(b) $GTV1$ $d = 5$

(c) $GTV1$ $d = 10$

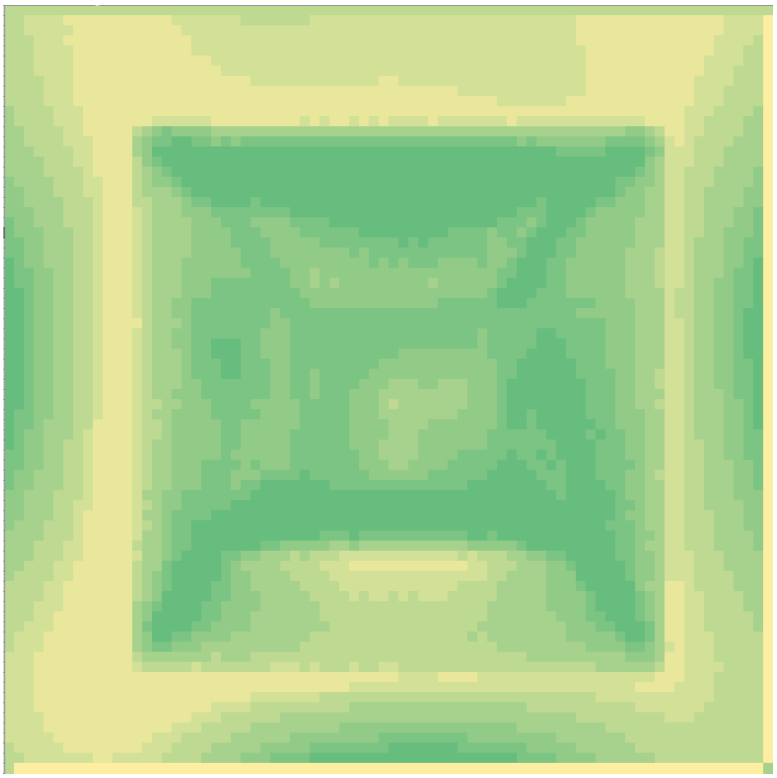


(d) $GTV1$ $d = 15$

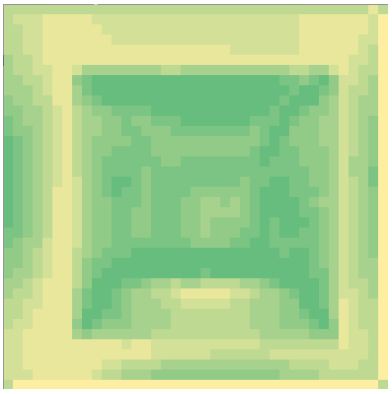


(e) $GTV1$ $d = 20$

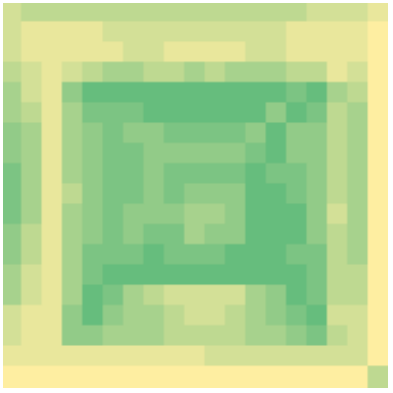
FIGURE A.10: The *absolute* error visualisation results for the Gonzalo Titanium V1 (GTV1) pyramid for different grid sizes (d).



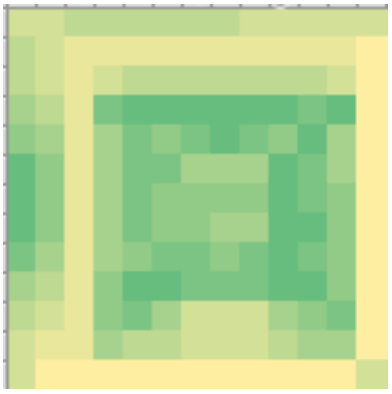
(a) $GTV1$ $d = 2.5$



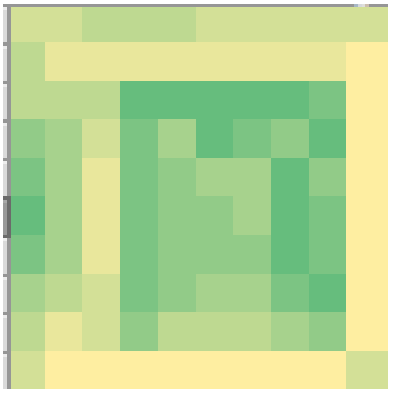
(b) $GTV1$ $d = 5$



(c) $GTV1$ $d = 10$

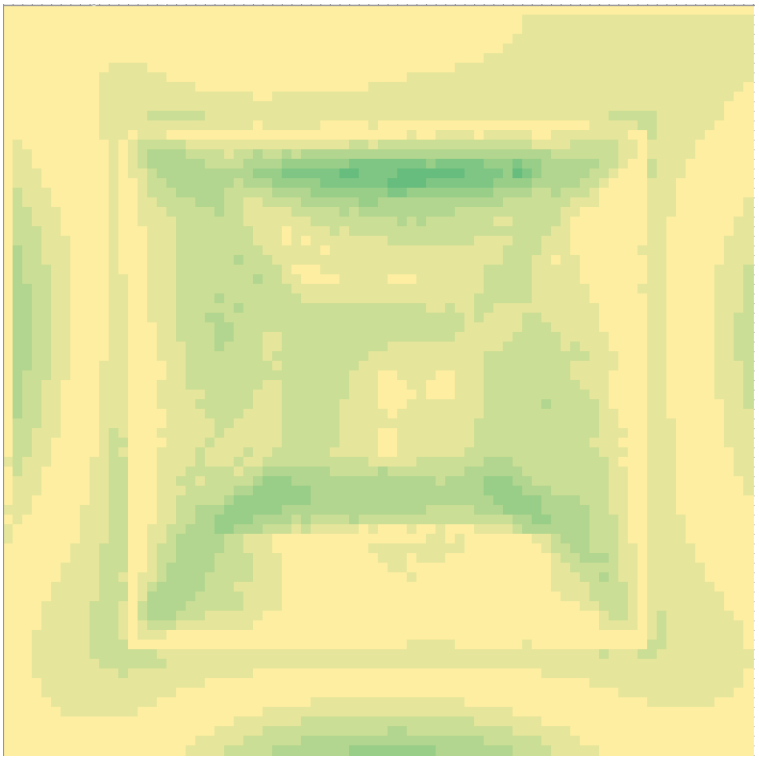


(d) $GTV1$ $d = 15$

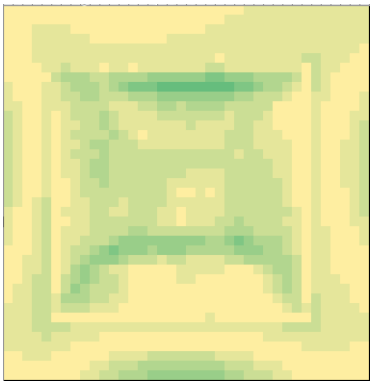


(e) $GTV1$ $d = 20$

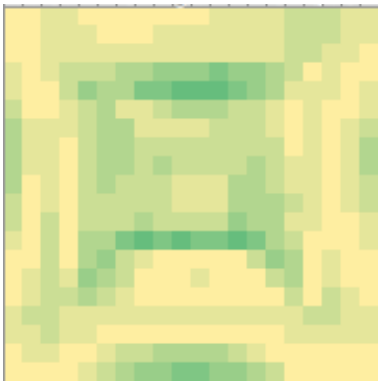
FIGURE A.11: The *directed* error visualisation results for the Gonzalo Titanium V1 (GTV1) pyramid for different grid sizes (d).



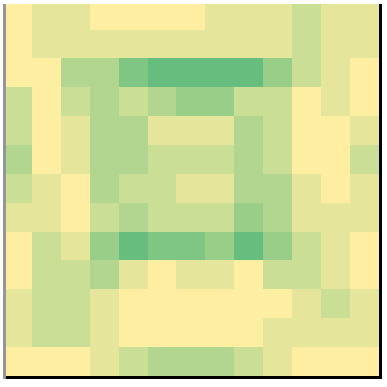
(a) $GTV2\ d = 2.5$



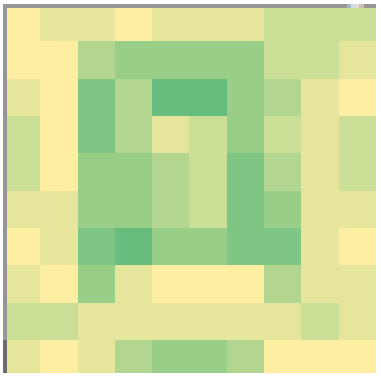
(b) $GTV2\ d = 5$



(c) $GTV2\ d = 10$

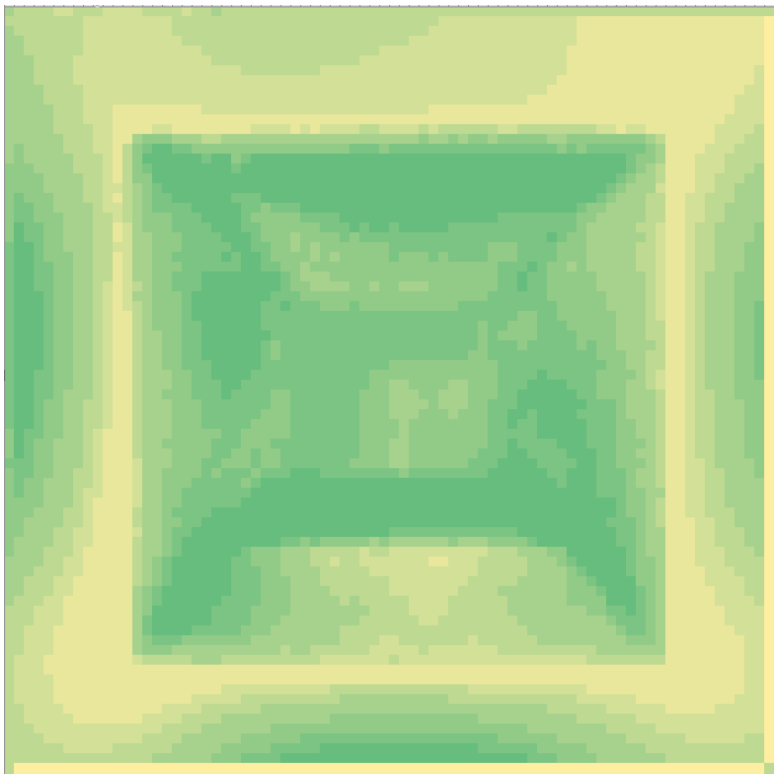


(d) $GTV2\ d = 15$

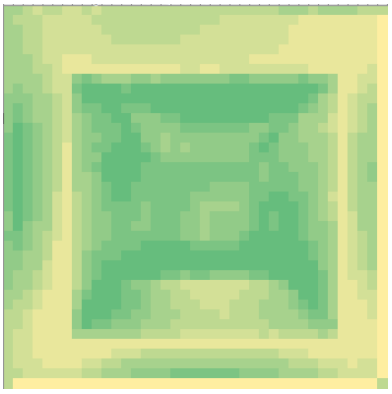


(e) $GTV2\ d = 20$

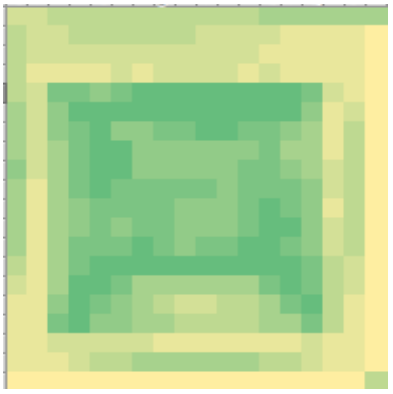
FIGURE A.12: The *absolute* error visualisation results for the Gonzalo Titanium V2 (GTV2) pyramid for different grid sizes (d).



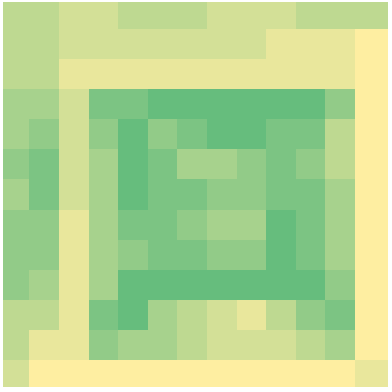
(a) $GTV2$ $d = 2.5$



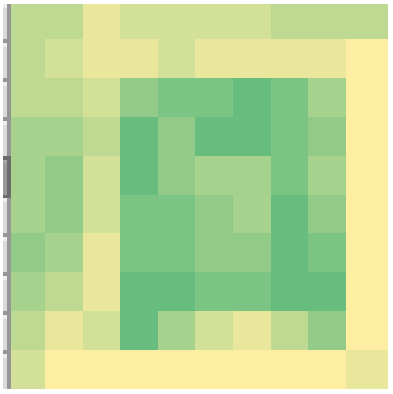
(b) $GTV2$ $d = 5$



(c) $GTV2$ $d = 10$

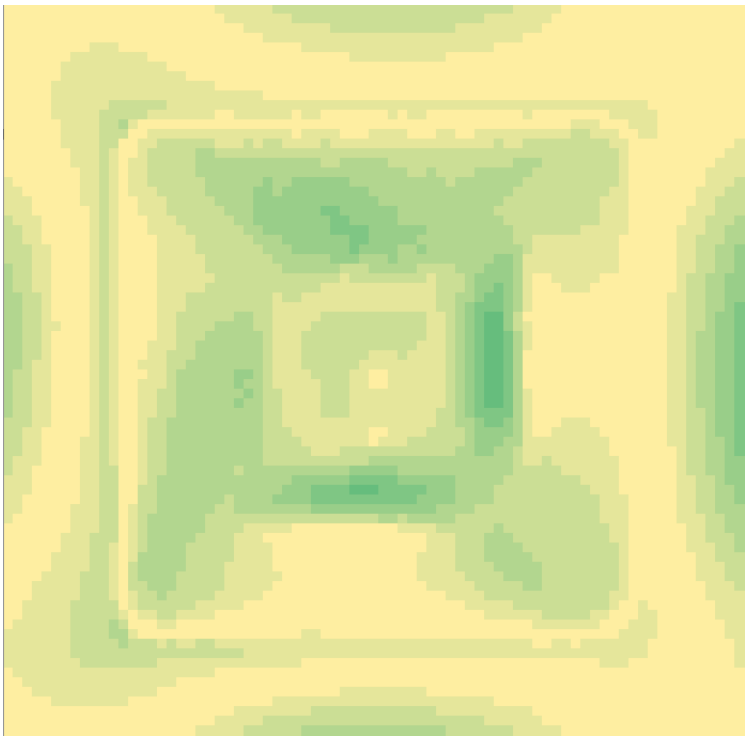


(d) $GTV2$ $d = 15$

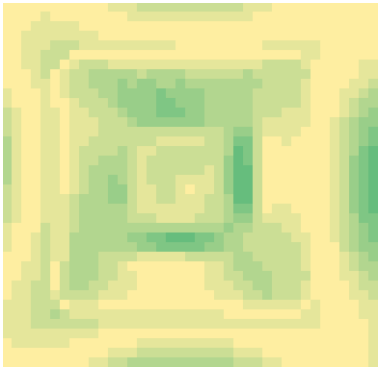


(e) $GTV2$ $d = 20$

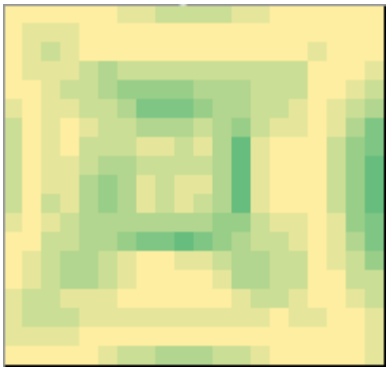
FIGURE A.13: The *directed* error visualisation results for the Gonzalo Titanium V2 (GTV2) pyramid for different grid sizes (d).



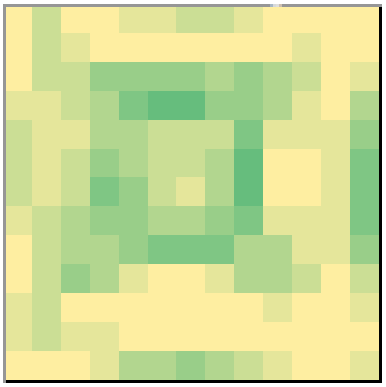
(a) $MTV1$ $d = 2.5$



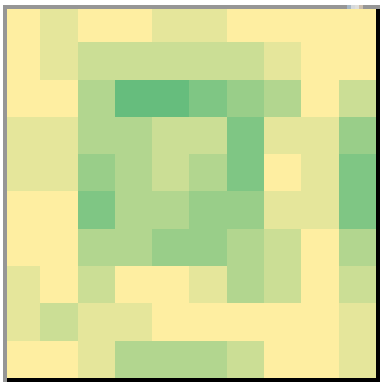
(b) $MTV1$ $d = 5$



(c) $MTV1$ $d = 10$

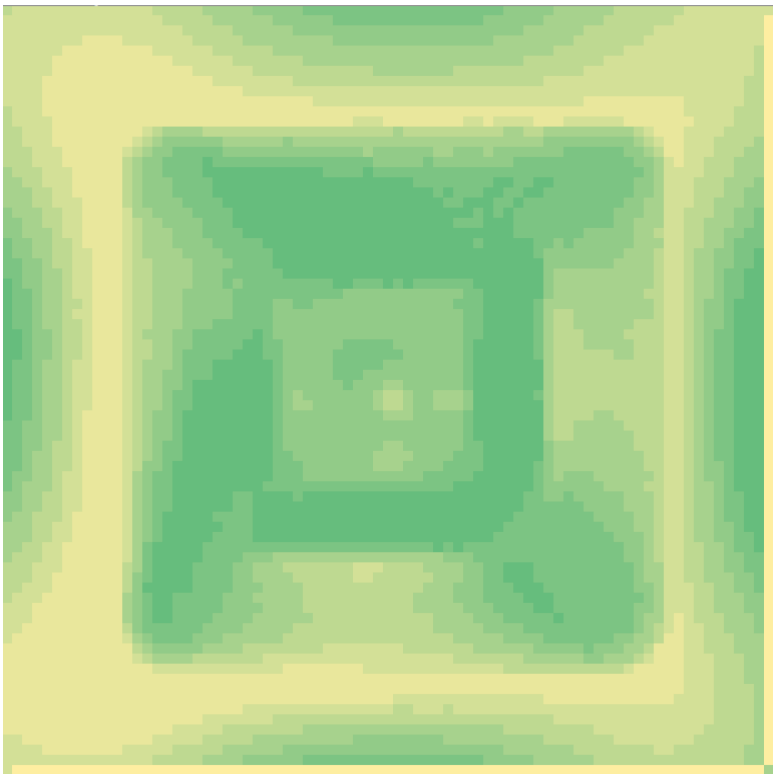


(d) $MTV1$ $d = 15$

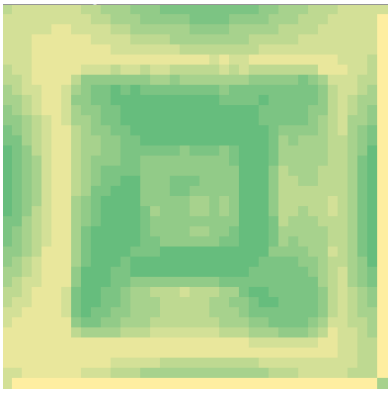


(e) $MTV1$ $d = 20$

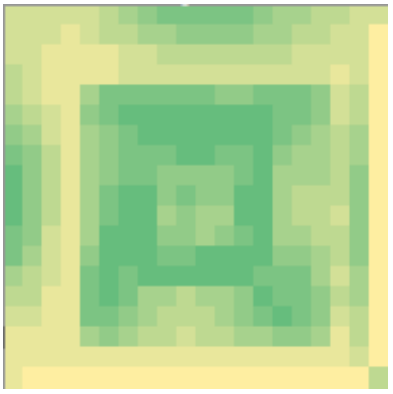
FIGURE A.14: The *absolute* error visualisation results for the Modified Titanium V1 (MTV1) pyramid for different grid sizes (d).



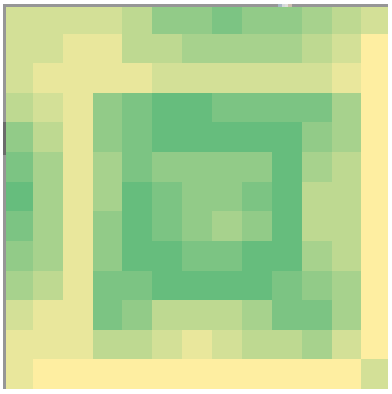
(a) $MTV1$ $d = 2.5$



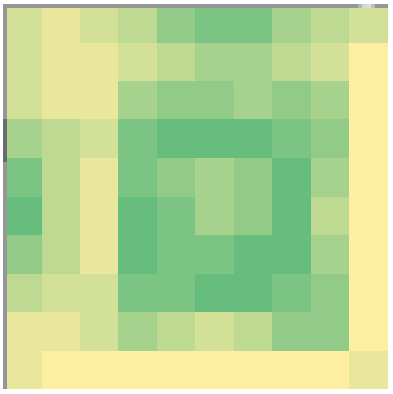
(b) $MTV1$ $d = 5$



(c) $MTV1$ $d = 10$

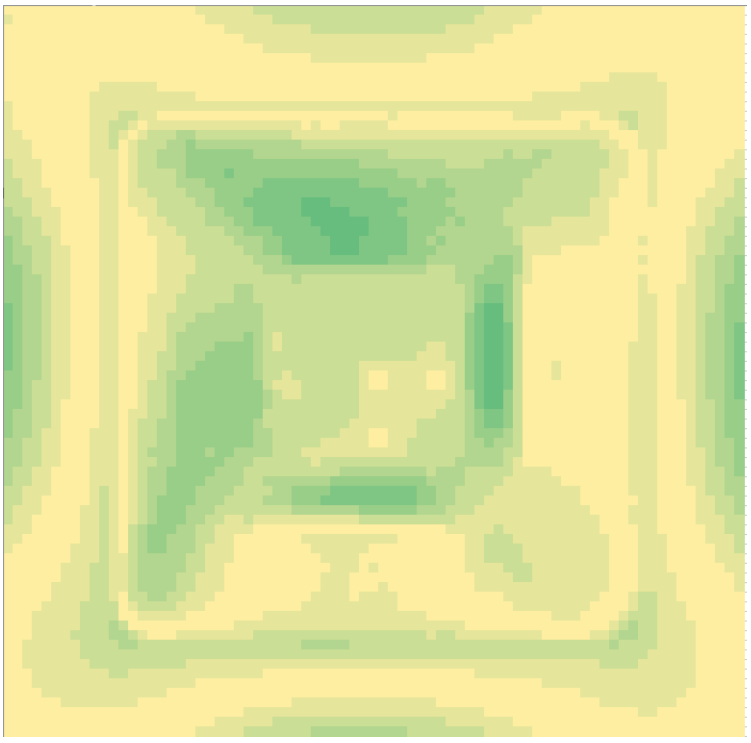


(d) $MTV1$ $d = 15$

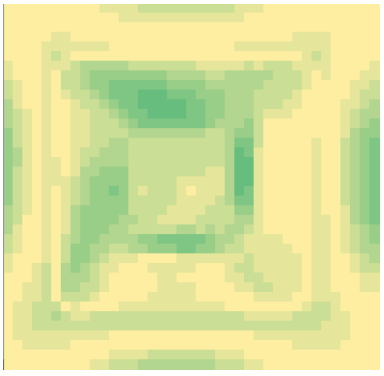


(e) $MTV1$ $d = 20$

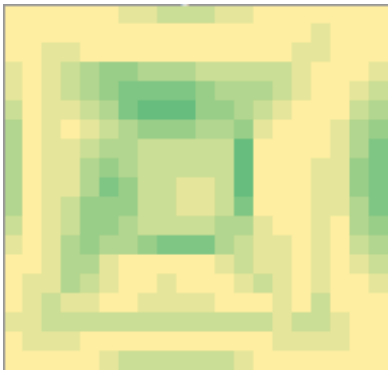
FIGURE A.15: The *directed* error visualisation results for the Modified Titanium V1 (MTV1) pyramid for different grid sizes (d).



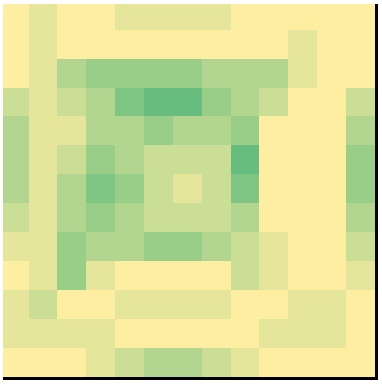
(a) $MTV2$ $d = 2.5$



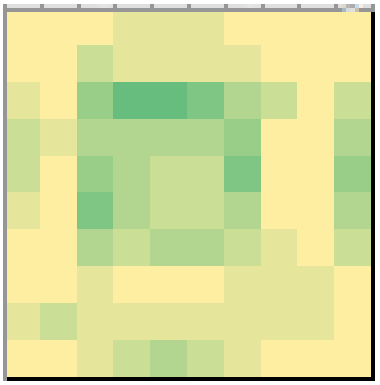
(b) $MTV2$ $d = 5$



(c) $MTV2$ $d = 10$

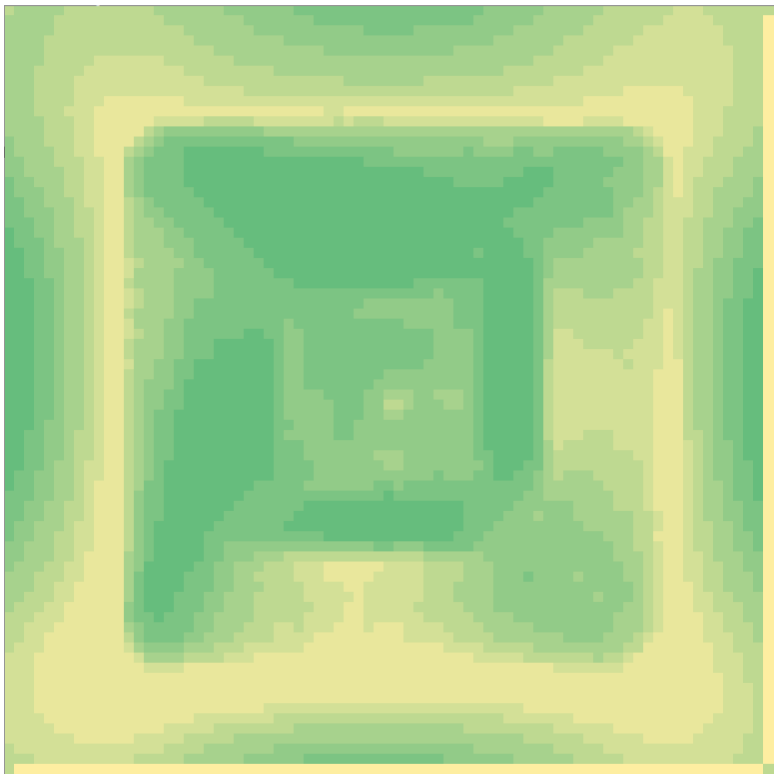


(d) $MTV2$ $d = 15$

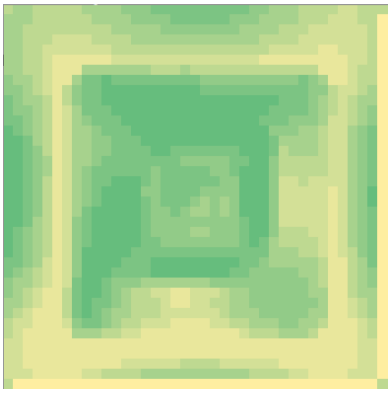


(e) $MTV2$ $d = 20$

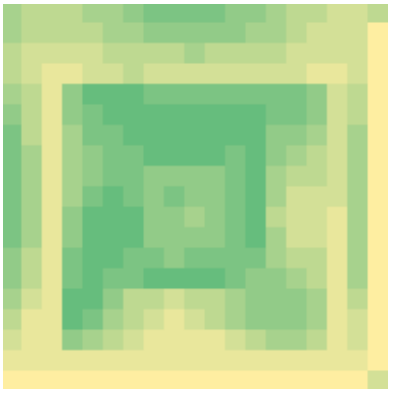
FIGURE A.16: The *absolute* error visualisation results for the Modified Titanium V2 (MTV2) pyramid for different grid sizes (d).



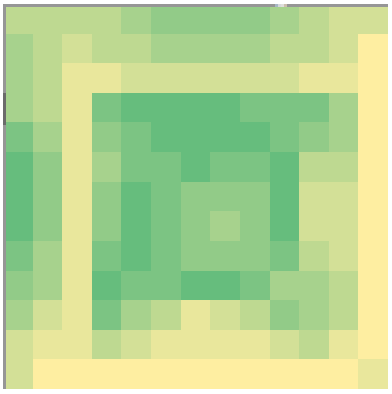
(a) $MTV2$ $d = 2.5$



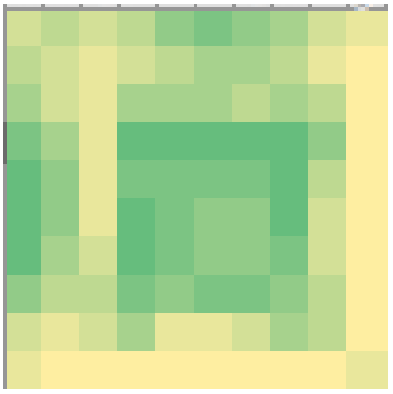
(b) $MTV2$ $d = 5$



(c) $MTV2$ $d = 10$



(d) $MTV2$ $d = 15$



(e) $MTV2$ $d = 20$

FIGURE A.17: The *directed* error visualisation results for the Modified Titanium V2 (MTV2) pyramid for different grid sizes (d).

Appendix B

Discretised Results for PS Representation Technique

This appendix presents the results of the preliminary experiments conducted to evaluate the *discretised* version of the Point Series Representation. The results were evaluated in terms of accuracy and AUC measurements. TCV was also used. In Chapter 6, results of experiments demonstrated that using *discretised* error values with the PS representation was *less* effective than when using *real* error values. Thus, for reasons of readability and succinctness, only the results using real error (springback) values were further considered. In this appendix the results from some additional experiments using *discretised* error (springback) are presented. The aim is to reinforce the results presented in Chapter 6. The additional experiments were conducted using a range of label sizes ($|L| = \{3, 5, 7, 9, 11, 13\}$) coupled with a range of grid size ($d = \{2.5, 5, 10, 15, 20\}$) for both the Gonzalo and Modified pyramids; GSV1, GSV2, GTV1, GTV2, MSV1, MSV2, MTV1 and MTV2. The neighbourhood size of $n = 5$ was chosen because this is the mid way value in the range of $n = \{3, 5, 7\}$ values (considered in Chapter 6). Only the key points representation was selected so as to limit the number of experiments undertaken and because there was no expectation, in the context of discretised versus real error comparison, that there will be any difference in operation.

Two groups of experiments were conducted: (i) using the same data set to generate a classifier and (ii) using different data sets to generate a classifier where a data set is used to train the classifier while the other one used to test it. Tables B.1 and B.2 present the results of the Gonzalo and Modified pyramids respectively for the first group. Tables B.3 and B.4 present the results for the second group where the classifier trained and tested on different data sets. The obtained results from both groups confirms that the *discretised* PS is not an appropriate alternative for the PS representation.

TABLE B.1: Discretised attributes with descretised error labels for Gonzalo pyramid using 5×5 *key* PS technique.

		d=2.5		d=5		d=10		d=15		d=20	
	$\|L\|$	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
GSV1	3	0.87	0.50	0.87	0.50	0.83	0.57	0.72	0.48	0.69	0.66
	5	0.85	0.25	0.84	0.25	0.81	0.38	0.78	0.27	0.82	0.37
	7	0.84	0.17	0.83	0.17	0.84	0.27	0.83	0.26	0.84	0.34
	9	0.85	0.13	0.85	0.13	0.87	0.25	0.83	0.18	0.82	0.16
	11	0.87	0.10	0.87	0.14	0.87	0.21	0.87	0.18	0.83	0.22
	13	0.88	0.08	0.88	0.11	0.89	0.16	0.86	0.10	0.86	0.22
GSV2	3	0.92	0.50	0.89	0.64	0.86	0.73	0.92	0.62	0.91	0.88
	5	0.91	0.37	0.88	0.35	0.87	0.47	0.81	0.70	0.86	0.62
	7	0.85	0.21	0.85	0.22	0.85	0.36	0.84	0.59	0.85	0.54
	9	0.87	0.19	0.85	0.15	0.84	0.17	0.78	0.36	0.84	0.46
	11	0.88	0.13	0.87	0.16	0.89	0.30	0.81	0.53	0.83	0.32
	13	0.89	0.10	0.89	0.16	0.91	0.25	0.85	0.36	0.85	0.35
GTV1	3	0.86	0.67	0.87	0.69	0.78	0.61	0.89	0.67	0.82	0.65
	5	0.84	0.34	0.83	0.34	0.81	0.34	0.82	0.66	0.89	0.57
	7	0.85	0.23	0.84	0.22	0.84	0.37	0.84	0.47	0.88	0.55
	9	0.85	0.16	0.86	0.19	0.86	0.27	0.87	0.51	0.88	0.36
	11	0.87	0.15	0.86	0.12	0.87	0.27	0.84	0.30	0.87	0.32
	13	0.88	0.11	0.88	0.12	0.90	0.27	0.88	0.40	0.90	0.31
GTV2	3	0.84	0.55	0.84	0.57	0.89	0.68	0.84	0.62	0.82	0.71
	5	0.84	0.31	0.83	0.30	0.79	0.30	0.83	0.55	0.81	0.38
	7	0.84	0.24	0.83	0.24	0.84	0.24	0.84	0.29	0.87	0.39
	9	0.85	0.17	0.84	0.15	0.86	0.19	0.85	0.27	0.87	0.29
	11	0.87	0.15	0.86	0.14	0.87	0.20	0.88	0.21	0.89	0.30
	13	0.88	0.12	0.87	0.09	0.89	0.21	0.89	0.23	0.87	0.08

TABLE B.2: Discretised attributes with descretised error labels for Modified pyramid using 5×5 *key* PS technique.

		d=2.5		d=5		d=10		d=15		d=20	
	$\|L\|$	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
MSV1	3	0.80	0.50	0.74	0.50	0.72	0.49	0.75	0.57	0.81	0.70
	5	0.78	0.27	0.77	0.29	0.74	0.29	0.74	0.30	0.77	0.26
	7	0.81	0.19	0.81	0.20	0.77	0.24	0.78	0.18	0.80	0.18
	9	0.84	0.15	0.84	0.15	0.84	0.17	0.84	0.32	0.83	0.14
	11	0.85	0.11	0.86	0.13	0.86	0.14	0.86	0.25	0.84	0.11
	13	0.87	0.09	0.88	0.12	0.87	0.10	0.87	0.21	0.88	0.16
MSV2	3	0.84	0.50	0.81	0.50	0.82	0.52	0.84	0.69	0.86	0.54
	5	0.80	0.26	0.74	0.23	0.81	0.33	0.81	0.42	0.77	0.27
	7	0.80	0.15	0.81	0.17	0.82	0.32	0.83	0.27	0.83	0.19
	9	0.83	0.11	0.83	0.12	0.83	0.12	0.85	0.32	0.87	0.21
	11	0.85	0.11	0.86	0.10	0.85	0.13	0.89	0.28	0.84	0.15
	13	0.87	0.08	0.88	0.11	0.87	0.10	0.88	0.20	0.88	0.23
MTV1	3	0.83	0.64	0.82	0.62	0.84	0.62	0.81	0.65	0.81	0.62
	5	0.83	0.36	0.82	0.35	0.78	0.34	0.84	0.60	0.82	0.50
	7	0.84	0.25	0.82	0.23	0.84	0.44	0.84	0.35	0.84	0.38
	9	0.86	0.22	0.86	0.21	0.84	0.16	0.87	0.35	0.84	0.17
	11	0.87	0.16	0.87	0.14	0.87	0.20	0.87	0.23	0.87	0.22
	13	0.89	0.15	0.88	0.13	0.89	0.26	0.89	0.16	0.89	0.15
MTV2	3	0.76	0.51	0.80	0.61	0.77	0.51	0.78	0.54	0.87	0.76
	5	0.80	0.30	0.80	0.29	0.80	0.27	0.82	0.43	0.84	0.53
	7	0.81	0.18	0.81	0.19	0.81	0.18	0.84	0.28	0.82	0.39
	9	0.84	0.14	0.84	0.15	0.84	0.15	0.85	0.17	0.83	0.32
	11	0.86	0.12	0.86	0.15	0.87	0.15	0.87	0.13	0.84	0.22
	13	0.88	0.11	0.88	0.12	0.88	0.15	0.90	0.24	0.85	0.19

TABLE B.3: Real attribute values with discretised error labels for Gonzalo pyramid using 5×5 *key* PS technique and by using *different* data sets.

Train-Test	$\|L\|$	d=2.5		d=5		d=10		d=15		d=20	
		Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
GSV1-GSV2	3	0.86	0.52	0.80	0.59	0.82	0.54	0.85	0.49	0.84	0.47
	5	0.85	0.43	0.81	0.42	0.77	0.29	0.82	0.52	0.84	0.41
	7	0.88	0.33	0.85	0.33	0.84	0.24	0.85	0.37	0.80	0.27
	9	0.89	0.28	0.86	0.27	0.86	0.20	0.86	0.27	0.86	0.21
	11	0.89	0.22	0.87	0.21	0.87	0.16	0.87	0.22	0.88	0.23
	13	0.91	0.21	0.89	0.19	0.90	0.16	0.90	0.20	0.86	0.10
GSV2-GSV1	3	0.83	0.51	0.83	0.53	0.80	0.51	0.87	0.55	0.81	0.47
	5	0.82	0.42	0.80	0.38	0.79	0.37	0.80	0.50	0.86	0.55
	7	0.84	0.39	0.83	0.35	0.81	0.33	0.80	0.45	0.79	0.39
	9	0.86	0.32	0.85	0.26	0.84	0.26	0.86	0.39	0.84	0.31
	11	0.86	0.27	0.87	0.20	0.85	0.23	0.85	0.33	0.86	0.33
	13	0.88	0.24	0.88	0.20	0.90	0.19	0.90	0.26	0.86	0.19
GTV1-GTV2	3	0.85	0.61	0.83	0.59	0.82	0.70	0.86	0.57	0.88	0.56
	5	0.84	0.46	0.82	0.41	0.83	0.51	0.86	0.45	0.85	0.62
	7	0.85	0.33	0.83	0.32	0.83	0.34	0.84	0.48	0.86	0.45
	9	0.88	0.31	0.86	0.27	0.86	0.30	0.89	0.42	0.84	0.36
	11	0.89	0.28	0.88	0.24	0.87	0.27	0.88	0.29	0.85	0.26
	13	0.90	0.25	0.89	0.20	0.90	0.25	0.90	0.30	0.87	0.31
GTV2-GTV1	3	0.86	0.60	0.84	0.63	0.79	0.64	0.90	0.83	0.86	0.57
	5	0.84	0.46	0.82	0.44	0.80	0.52	0.85	0.67	0.84	0.50
	7	0.86	0.38	0.85	0.41	0.83	0.36	0.88	0.55	0.86	0.39
	9	0.88	0.35	0.87	0.32	0.86	0.32	0.89	0.46	0.87	0.31
	11	0.89	0.32	0.88	0.30	0.87	0.31	0.89	0.44	0.88	0.21
	13	0.90	0.28	0.89	0.26	0.90	0.27	0.90	0.37	0.90	0.26

TABLE B.4: Discretised error labels for Modified pyramid using 5×5 *key* PS technique and by using *diferent* data sets.

Train-Test		d=2.5		d=5		d=10		d=15		d=20	
	$\|L\|$	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
MSV1-MSV2	3	0.86	0.75	0.85	0.75	0.84	0.72	0.89	0.76	0.90	0.74
	5	0.84	0.58	0.82	0.54	0.82	0.50	0.83	0.57	0.83	0.47
	7	0.86	0.50	0.84	0.45	0.86	0.51	0.85	0.35	0.84	0.43
	9	0.87	0.45	0.86	0.42	0.86	0.41	0.86	0.42	0.86	0.36
	11	0.88	0.40	0.87	0.35	0.87	0.34	0.87	0.34	0.88	0.31
	13	0.89	0.36	0.89	0.31	0.90	0.26	0.90	0.29	0.90	0.28
MSV2-MSV1	3	0.86	0.81	0.86	0.82	0.85	0.80	0.89	0.84	0.90	0.90
	5	0.84	0.56	0.82	0.56	0.82	0.53	0.84	0.60	0.84	0.53
	7	0.86	0.46	0.84	0.43	0.87	0.47	0.85	0.42	0.85	0.51
	9	0.87	0.39	0.87	0.38	0.86	0.37	0.86	0.41	0.87	0.48
	11	0.88	0.33	0.87	0.30	0.87	0.29	0.88	0.34	0.88	0.36
	13	0.89	0.28	0.89	0.26	0.90	0.24	0.90	0.28	0.89	0.38
MTV1-MTV2	3	0.87	0.79	0.87	0.79	0.88	0.79	0.83	0.74	0.86	0.76
	5	0.89	0.70	0.88	0.67	0.88	0.72	0.84	0.63	0.84	0.61
	7	0.88	0.63	0.87	0.52	0.87	0.55	0.85	0.53	0.85	0.54
	9	0.89	0.51	0.88	0.82	0.87	0.48	0.86	0.48	0.86	0.46
	11	0.88	0.39	0.88	0.37	0.89	0.42	0.88	0.46	0.88	0.41
	13	0.89	0.32	0.89	0.32	0.90	0.30	0.90	0.38	0.88	0.31
MTV2-MTV1	3	0.86	0.78	0.87	0.78	0.87	0.78	0.84	0.762	0.86	0.73
	5	0.89	0.64	0.88	0.64	0.88	0.68	0.85	0.62	0.85	0.65
	7	0.88	0.51	0.87	0.48	0.87	0.49	0.85	0.48	0.85	0.52
	9	0.89	0.40	0.88	0.42	0.87	0.40	0.86	0.41	0.86	0.48
	11	0.89	0.34	0.89	0.32	0.88	0.34	0.88	0.38	0.89	0.44
	13	0.90	0.28	0.89	0.27	0.90	0.26	0.90	0.26	0.89	0.33

Appendix C

AUC Calculation based on Mann-Whitney-Wilcoxon.

C.1 Introduction

This Appendix presents a full example to demonstrate the AUC estimation based on the Mann-Whitney-Wilcoxon (MWW) statistical method. The Mann-Whitney-Wilcoxon (MWW) statistical method[1] was proposed in [95] to estimate AUC values based on a *Ranking* concept as shown in Equation C.1 where c is the number of classes. In Equation C.1 $A_{i,j}$ is calculated based on the MWW values obtained with respect to class i and class j as shown in Equation C.2. The MWW value is calculated based on the Man-Whitney-Wilcoxon (MWW) statistic (or rank sum) [26]. The ranking concept used in this context is mainly based on the signal detection theory [95], and hence the actual value of an attribute is denoted by (*signal "S"*) and the predicted value is denoted by (*response "R"*) with respect to a *binary* classification. Thus the MWW value of class i with respect to class j is calculated as shown in Equation C.3 where n_1 denotes number of positive examples[2] while n_2 is the number of negative examples[3] with respect to c_i and r_i is the sum rank for the positive examples with respect to class i. The rank for a given record of class i and with respect to class j is obtained from a value obtained according the combinations of R and S. Table C.1 presents the values (denoted as a group ID in this example) of the different potential combinations of R and S. The group ID value associated with each record is used to order the records sequentially starting from group ID 1 to group ID 4 and consequently all the records of group ID 1 will be followed by all the records of group ID 2 and so on. Then a sequential number (rank) is given to records starting from the first record in group ID 1 to the last record in group ID 4. An example of estimating the AUC values using the ranking process for 51 records

[1]The Wilcoxon rank sum test makes no assumption about the probability distributions and is completely based on the scores (rank) of the tuples.

[2]Positive examples are the number of records that are actually labelled with class i with respect to j and this means that $S = 1$.

[3]Negative examples are the number of records that are actually *not* labelled with class i with respect to j and this means that $S = 0$.

using three classes c_1, c_2 and c_3 is presented below. The actual and the predicted label of the data set, that are denoted using the S and R variables respectively, are presented as 1 value with respect to the set of classes c_1, c_2 and c_3 as shown in Table C.2. The MWW value of c_1 with respect to class $c2$ is presented in Table C.3 and the MWW value of c_2 with respect to class c_1 is presented in Table C.4. Both tables shows the associated group ID and the rank values. The numbering starts with the first record in group ID 1 to last record in group ID 4 sequentially. According to the given ranks, the last rows indicates the number of positive and negative records, the sum of ranks and the MWW values with respect to class c_1 and c_2 in Tables C.3 and C.4 respectively. Similarly, Tables C.5 and C.6 presents the MWW value of class c_1 with respect to class c_3 and the MWW values of class c_3 with respect to class c_1, while Tables C.7 and C.8 presents the MWW values of class c_2 with respect to class c_3 and the MWW values of class c_3 with respect to class c_2. Finally Table C.9 presents the calculation of the overall AUC using Equation C.3 which is based on the MWW values calculated with respect to classes c_1, c_2 and c_3.

$$AUC = \frac{2}{c(c-1)} \sum_{i<j} A_{i,j} \tag{C.1}$$

$$A_{i,j} = \frac{MWW(i|j) + MWW(j|i)}{2} \tag{C.2}$$

$$MWW(i|j) = \frac{\sum r_i - \frac{n_1(n_1+1)}{2}}{n_1 n_2} \tag{C.3}$$

TABLE C.1: The values (Group ID) of different combinations of R and S based on Hand et al. [95].

R	S	Group ID
0	1	1
0	0	2
1	0	3
1	1	4

TABLE C.2: Data sets example.

Records	S			R		
	$c1$	$c2$	$c3$	$c1$	$c2$	$c3$
1	1	0	0	1	0	0
2	1	0	0	1	0	0
3	1	0	0	1	0	0
4	0	1	0	0	1	0
5	0	1	0	0	1	0
6	0	1	0	0	1	0
7	0	1	0	0	1	0
8	0	1	0	0	1	0
9	0	1	0	0	1	0
10	0	1	0	0	1	0
11	0	1	0	0	1	0
12	0	1	0	0	1	0
13	0	1	0	0	1	0
14	0	1	0	0	1	0
15	0	1	0	0	1	0
16	0	1	0	0	0	1
17	0	1	0	0	0	1
18	0	1	0	0	0	1
19	0	0	1	0	0	1
20	0	0	1	0	0	1
21	0	0	1	0	0	1
22	0	0	1	0	0	1
23	0	0	1	0	0	1
24	0	0	1	0	0	1
25	0	0	1	0	0	1
26	0	0	1	0	0	1
27	0	0	1	0	0	1
28	0	0	1	0	0	1
29	0	0	1	0	0	1
30	0	0	1	0	0	1
31	0	0	1	0	0	1
32	0	0	1	0	0	1
33	0	0	1	0	0	1
34	0	0	1	0	0	1
35	0	0	1	0	0	1
36	0	0	1	0	0	1
37	0	0	1	0	0	1
38	0	0	1	0	0	1
39	0	0	1	0	0	1
40	0	0	1	0	0	1
41	0	0	1	0	0	1
42	0	0	1	0	0	1
43	0	0	1	0	0	1
44	0	0	1	0	0	1
45	0	0	1	0	0	1
46	0	0	1	0	0	1
47	0	0	1	0	0	1
48	0	0	1	0	0	1
49	0	0	1	0	0	1
50	0	0	1	0	0	1
51	0	0	1	0	0	1

TABLE C.3: The $MWW(c_1|c_2)$ value.

S	R	group ID	ranks
1	1	4	**16**
1	1	4	**17**
1	1	4	**18**
0	0	2	1
0	0	2	2
0	0	2	3
0	0	2	4
0	0	2	5
0	0	2	6
0	0	2	7
0	0	2	8
0	0	2	9
0	0	2	10
0	0	2	11
0	0	2	12
0	0	2	13
0	0	2	14
0	0	2	15
$n_1 = 3$ $n_2 = 15$ $\sum r_i = 51$		$MWW(c_1\|c_2) = \frac{51-6}{45} = 1$	

TABLE C.4: The $MWW(c_2|c_1)$ value

S	R	group ID	ranks
0	0	2	4
0	0	2	5
0	0	2	6
1	1	4	**7**
1	1	4	**8**
1	1	4	**9**
1	1	4	**10**
1	1	4	**11**
1	1	4	**12**
1	1	4	**13**
1	1	4	**14**
1	1	4	**15**
1	1	4	**16**
1	1	4	**17**
1	1	4	**18**
1	0	1	**1**
1	0	1	**2**
1	0	1	**3**
$n_1 = 15$ $n_2 = 3$ $\sum r_i = 156$		$MWW(c_2\|c_1) = \frac{156-120}{45} = 0.80$	

TABLE C.5: The $MWW(c_1|c_3)$ value.

S	R	group ID	ranks
1	1	4	**34**
1	1	4	**35**
1	1	4	**36**
0	0	2	1
0	0	2	2
0	0	2	3
0	0	2	4
0	0	2	5
0	0	2	6
0	0	2	7
0	0	2	8
0	0	2	9
0	0	2	10
0	0	2	11
0	0	2	12
0	0	2	13
0	0	2	14
0	0	2	15
0	0	2	16
0	0	2	17
0	0	2	18
0	0	2	19
0	0	2	20
0	0	2	21
0	0	2	22
0	0	2	23
0	0	2	24
0	0	2	25
0	0	2	26
0	0	2	27
0	0	2	28
0	0	2	29
0	0	2	30
0	0	2	31
0	0	2	32
0	0	2	33
$n_1 = 3$ $n_2 = 33$ $\sum r_i = 105$		$MWW(c_1\|c_3) = \frac{105-6}{99} = 1$	

TABLE C.6: The $MWW(c_3|c_1)$ value

S	R	group ID	ranks
0	0	2	1
0	0	2	2
0	0	2	3
1	1	4	**4**
1	1	4	**5**
1	1	4	**6**
1	1	4	**7**
1	1	4	**8**
1	1	4	**9**
1	1	4	**10**
1	1	4	**11**
1	1	4	**12**
1	1	4	**13**
1	1	4	**14**
1	1	4	**15**
1	1	4	**16**
1	1	4	**17**
1	1	4	**18**
1	1	4	**19**
1	1	4	**20**
1	1	4	**21**
1	1	4	**22**
1	1	4	**23**
1	1	4	**24**
1	1	4	**25**
1	1	4	**26**
1	1	4	**27**
1	1	4	**28**
1	1	4	**29**
1	1	4	**30**
1	1	4	**31**
1	1	4	**32**
1	1	4	**33**
1	1	4	**34**
1	1	4	**35**
1	1	4	**36**
$n_1 = 33$ $n_2 = 3$ $\sum r_i = 660$		$MWW(c_3\|c_1) = \frac{660-561}{99} = 1$	

TABLE C.7: The $MWW(c_2|c_3)$ value.

S	R	group ID	ranks
1	1	4	**37**
1	1	4	**38**
1	1	4	**39**
1	1	4	**40**
1	1	4	**41**
1	1	4	**42**
1	1	4	**43**
1	1	4	**44**
1	1	4	**45**
1	1	4	**46**
1	0	1	**47**
1	0	1	**48**
1	0	2	**1**
1	0	2	**2**
1	0	2	**3**
0	0	2	4
0	0	2	5
0	0	2	6
0	0	2	7
0	0	2	8
0	0	2	9
0	0	2	10
0	0	2	11
0	0	2	12
0	0	2	13
0	0	2	14
0	0	2	15
0	0	2	16
0	0	2	17
0	0	2	18
0	0	2	19
0	0	2	20
0	0	2	21
0	0	2	22
0	0	2	23
0	0	2	24
0	0	2	25
0	0	2	26
0	0	2	27
0	0	2	28
0	0	2	29
0	0	2	30
0	0	2	31
0	0	2	32
0	0	2	33
0	0	2	34
0	0	2	35
0	0	2	36
$n_1 = 15$ $n_2 = 33$ $\sum r_i = 516$		$MWW(c_2\|c_3) = \frac{516-120}{495} = 0.8$	

TABLE C.8: The $MWW(c_3|c_2)$ value

S	R	group ID	ranks
0	0	2	1
0	0	2	2
0	0	2	3
0	0	2	4
0	0	2	5
0	0	2	6
0	0	2	7
0	0	2	8
0	0	2	9
0	0	2	10
0	0	2	11
0	0	2	12
0	1	3	13
0	1	3	14
0	1	3	15
1	1	4	**16**
1	1	4	**17**
1	1	4	**18**
1	1	4	**19**
1	1	4	**20**
1	1	4	**21**
1	1	4	**22**
1	1	4	**23**
1	1	4	**24**
1	1	4	**25**
1	1	4	**26**
1	1	4	**27**
1	1	4	**28**
1	1	4	**29**
1	1	4	**30**
1	1	4	**31**
1	1	4	**32**
1	1	4	**33**
1	1	4	**34**
1	1	4	**35**
1	1	4	**36**
1	1	4	**37**
1	1	4	**38**
1	1	4	**39**
1	1	4	**40**
1	1	4	**41**
1	1	4	**42**
1	1	4	**43**
1	1	4	**44**
1	1	4	**45**
1	1	4	**46**
1	1	4	**47**
1	1	4	**48**
$n_1 = 33$ $n_2 = 15$ $\sum r_i = 1056$		$MWW(c_3\|c_2) = \frac{1056-561}{495} = 1$	

TABLE C.9: The overall AUC value for the given data sets.

MWW (i,j)	MWW(j,i)	A(i,j)
MWW $(c_1, c_2) = 1$	MWW $(c_2, c_1) = 0.8$	A$(c_1\|c_2) = \frac{1+0.8}{2} = 0.9$
MWW $(c_1, c_3) = 1$	MWW $(c_3, c_1) = 1$	A$(c_1\|c_3) = \frac{1+1}{2} = 1$
MWW $(c_2, c_3) = 0.8$	MWW $(c_3, c_2) = 1$	A$(c_2\|c_3) = \frac{0.8+1}{2} = 0.9$
AUC$= \frac{2\times 2.8}{3\times 2} = 0.93$		

References

[1] A. Aamodt and E. Plaza, *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*, AI Communications **7** (1994), no. 1, 39–59.

[2] R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning, and T. Bollinger, *The Quest Data Mining System*, In Proc. of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, 1996, pp. 244–249.

[3] R. Agrawal and R. Srikant, *Fast Algorithms for Mining Association Rules in Large Databases*, Proceedings of the 20th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[4] T. Ahonen, A. Hadid, and M. Pietikinen, *Face recognition with local binary patterns*, Computer Vision - ECCV 2004, vol. 3021, Springer, 2004, pp. 469–481.

[5] A. Albarrak, F. Coenen, and Y. Zheng, *Classification of Volumetric Retinal Images using Overlapping Decomposition and Tree Analysis.*, CBMS, IEEE, 2013, pp. 11–16.

[6] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, *Point Set Surfaces*, Proceedings of the Conference on Visualization, IEEE Computer Society, 2001, pp. 21–28.

[7] A. Amresh, G. Farin, and A. Razdan, *Adaptive subdivision schemes for triangular meshes*, Hierarchical and Geometric Methods in Scientific Visualization, Springer, 2002, pp. 319–327.

[8] S. Ando, *Image Field Categorization and Edge/Corner Detection from Gradient Covariance*, IEEE Transactions on Pattern Analysis and Machine Intelligences **22** (2000), no. 2, 179–190.

[9] E. Angelopoulou and L.B. Wolff, *Sign of gaussian curvature from curve orientation in photometric space*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 10, 1056–1066.

[10] A.D. Anggono, W.A. Siswanto, and B. Omar, *Combined method of spring-forward and spring-back for die compensation acceleration*, Modeling, Simulation and Applied Optimization (ICMSAO), 2011 4th International Conference on, April 2011, pp. 1–6.

[11] H. Anton, *Elementary Linear Algebra*, 8th ed., John Wiley & Sons, 2000.

[12] B. T. Araghi, A. Göttmann, M. Bambach, G. Hirt, G. Bergweiler, J. Diettrich, M. Steiners, and A. Saeed-Akbari, *Review on the Development of a Hybrid Incremental Sheet Forming System for Small Batch Sizes and Individualized Production.*, Production Engineering **5** (2011), no. 4, 393–404.

[13] B. T. Araghi, G.L. Manco, M. Bambach, and G. Hirt, *Investigation into a New Hybrid Forming Process: Incremental Sheet Forming Combined with Stretch Forming*, CIRP Annals - Manufacturing Technology **58** (2009), no. 1, 225–228.

[14] N. Asnafi, *Springback and fracture in v-die air bending of thick stainless steel sheets*, Materials & Design **21** (2000), no. 3, 217–236.

[15] K. Ataman, W. Street, and Y. Zhang, *Learning to Rank by Maximizing AUC with Linear Programming*, In IEEE International Joint Conference on Neural Networks, 2006, pp. 123–129.

[16] L. Baoli, L. Qin, and Y. Shiwen, *An Adaptive K-Nearest Neighbor Text Categorization Strategy*, ACM Transactions on Asian Language Information Processing (TALIP) **3** (2004), no. 4, 215–226.

[17] A. K. Behera, J. Verbert, B. Lauwers, and J. Duflou, *Tool Path Compensation Strategies for Single Point Incremental Sheet Forming using Multivariate Adaptive Regression Splines*, Computer-Aided Design **45** (2013), no. 3, 575–590.

[18] A. Behrouzi, B. M. Dariani, and M. Shakeri, *A One-step Analytical Approach for Springback Compensation in Channel Forming Process*, Proceedings of the World Congress on Engineering 2009, 2009, pp. 1757–1762.

[19] F. Bernardini, C. Bajaj, J. Chen, and D. Schikore, *Automatic Reconstruction of 3D CAD Models from Digital Scans*, International Journal of Computational Geometry and Applications **9** (1999), 327–369.

[20] D. J. Berndt and J. Clifford, *Using Dynamic Time Warping to Find Patterns in Time Series*, KDD Workshop, 1994, pp. 359–370.

[21] M.J. Berry and G.S. Linoff, *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*, Wiley, 2004.

[22] M. H. Bharati, J. J. Liu, and J. F. MacGregor, *Image Texture Analysis: Methods and Comparisons*, Chemometrics and Intelligent Laboratory Systems **72** (2004), no. 1, 57–71.

[23] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[24] C. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 2006.

[25] M. Botsch, M. Pauly, C. Rossl, S. Bischoff, and L. Kobbelt, *Geometric Modeling Based on Triangle Meshes*, ACM SIGGRAPH 2006 Courses, ACM, 2006.

[26] A. Bradley, *The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms*, Pattern Recognition **30** (1997), 1145–1159.

[27] G. W. Brown, *Errors, types i and ii*, American Journal of Diseases of Children **137** (1983), no. 6, 586–591.

[28] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers, *Combined Region and Motion-Based 3D Tracking of Rigid and Articulated Objects*, IEEE Transactions on Pattern Analysis and Machine Intelligence **32** (2010), no. 3, 402–415.

[29] *BS ISO 1101:2005 Geometrical Product Specifications (GPS) - Geometrical tolerancing - Tolerances of form, orientation, location and run-out- Generalities, definitions, symbols, indications on drawings*, 2005.

[30] R. J. Campbell, , and P. J. Flynn, *A Survey of Free-form Object Representation and Recognition Techniques*, Computer Vision and Image Understanding **81** (2001), no. 2, 166–210.

[31] J. Canny, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence **8** (1986), no. 6, 679–698.

[32] J. Cao, S. Wushour, X. Yao, N. Li, J. Liang, and X. Liang, *Sharp Feature Extraction in Point Clouds*, Image Processing **6** (2012), no. 7, 863 –869.

[33] Q. Cao and M. E. Parry, *Neural network earnings per share forecasting models: A comparison of backward propagation and the genetic algorithm*, Decision Support Systems **47** (2009), no. 1, 32–41.

[34] W.D Carden, L.M Geng, D.K Matlock, and R.H Wagoner, *Measurement of Springback*, International Journal of Mechanical Sciences **44** (2002), no. 1, 79–101.

[35] J. Catlett, *On Changing Continuous Attributes into Ordered Discrete Attributes*, Proceedings of the European Working Session on Learning on Machine Learning, Springer, 1991, pp. 164–178.

[36] S. Chatti and N. Hermi, *The Effect of Non-linear Recovery on Springback Prediction*, Computers and Structures **89** (2011), no. 13-14, 1367–1377.

[37] M. Chen, J. Han, and P. S. Yu, *Data Mining: An Overview from Database Perspective*, IEEE Transactions on Knowledge and Data Engineering **8** (1996), 866–883.

[38] H. S. Cheng, J. Cao, and Z. C. Xia, *An Accelerated Springback Compensation Method*, International Journal of Mechanical Sciences **49** (2007), no. 3, 267–279.

[39] S. Chu, E. Keogh, D. Hart, M. Pazzani, and M. Pazzani, *Iterative Deepening Dynamic Time Warping for Time Series*, In Proc 2 nd SIAM International Conference on Data Mining, 2002.

[40] R.M Cleveland and A.K Ghosh, *Inelastic Effects on Springback in Metals*, International Journal of Plasticity **18** (2002), no. 5-6, 769–785.

[41] W. Cohen, *Fast effective rule induction*, Twelfth International Conference on Machine Learning, Morgan Kaufmann, 1995, pp. 115–123.

[42] E.W. Collings, *Materials Properties Handbook: Titanium Alloys*, ASM International, 1994.

[43] R.D. Cook, *Concepts and Applications of Finite Element Analysis*, Wiley, 2001.

[44] M. W. Craven and J. W. Shavlik, *Using Neural Networks for Data Mining*, Future Generation Computer Systems **13** (1997), no. 2-3, 211–229.

[45] S. L. Culp and University of Michigan, *Warping Methods for Means and Variances in Functional Data*, University of Michigan, 2008.

[46] M. Das and J. Anand, *Robust Edge Detection in Noisy Images using an Adaptive Stochastic Gradient Technique*, Image Processing, 1995. Proceedings., International Conference on, vol. 2, Oct 1995, pp. 149–152 vol.2.

[47] L.J. de Vin, A.H. Streppel, U.P. Singh, and H.J.J. Kals, *A Process Model for Air Bending*, Journal of Materials Processing Technology **57** (1996), no. 1-2, 48–54.

[48] K. Delibasis, A. Kechriniotis, and I. Maglogiannis, *A Novel Tool for Segmenting 3D Medical Images Based on Generalized Cylinders and Active Surfaces*, Computer Methods and Programs in Biomedicine **111** (2013), no. 1, 148–165.

[49] K. Demarsin, D. Vanderstraeten, T. Volodine, and D. Roose, *Detection of closed sharp edges in point clouds using normal estimation and graph theory*, Computer-Aided Design **39** (2007), no. 4, 276–283.

[50] Janez Demšar, *Statistical comparisons of classifiers over multiple data sets*, Journal of machine learning research **7** (2006), 1–30.

[51] M. Diligenti, M. Maggini, and L. Rigutini, *Learning Similarities for Text Documents Using Neural Networks*, In Artificial Neural Networks in Pattern Recognition (ANNPR), Florence - Italy, 2003, pp. 12–13.

[52] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, *Querying and mining of time series data: Experimental comparison of representations and distance measures*, Proceedings of the VLDB Endowment **1** (2008), no. 2, 1542–1552.

[53] K. Dittakan, F. Coenen, and R. Christley, *Satellite image mining for census collection: A comparative study with respect to the ethiopian hinterland*, Machine Learning and Data Mining in Pattern Recognition, vol. 7988, Springer, 2013, pp. 260–274.

[54] J. Dougherty, R. Kohavi, and M. Sahami, *Supervised and Unsupervised Discretization of Continuous Features*, ICML, Morgan Kaufmann, 1995, pp. 194–202.

[55] M.H. Dunham, *Data mining: Introductory and advanced topics*, Pearson Education, 2006.

[56] P. S. Dunston, S. R. Ranjithan, and L. E. Bernold, *Neural Network Model for the Automated Control of Springback in Rebars.*, IEEE Expert **11** (1996), no. 4, 45–49.

[57] A. Efrat, Q. Fan, and S. Venkatasubramanian, *Curve Matching, Time Warping, and Light Fields: New Algorithms for Computing Similarity Between Curves*, Journal of mathematical imaging and vision **27** (2007), no. 3, 203–216.

[58] J. Egan, *Signal detection theory and roc-analysis*, Academic Press, 1975.

[59] P. Egerton and William W. Hall, *Computer Graphics: Mathematical First Steps*, 1st ed., Simon & Schuster International, 1998.

[60] C. Eisenacher, Q. Meyer, and C. Loop, *Real-time view-dependent rendering of parametric surfaces*, Proceedings of the 2009 symposium on Interactive 3D graphics and games, ACM, 2009, pp. 137–143.

[61] T. Fawcett, *An Introduction to ROC Analysis*, Pattern recognition letters **27** (2006), no. 8, 861–874.

[62] U. Fayyad, G. Piatetsky-shapiro, and P. Smyth, *From Data Mining to Knowledge Discovery in Databases*, AI Magazine **17** (1996), 37–54.

[63] U. Fayyad, G. Piatetsky-shapiro, and P. Smyth, *Knowledge Discovery and Data Mining: Towards a Unifying Framework*, Proceedings of the 2nd international conference on Knowledge Discovery and Data mining (KDD'96), AAAI Press, August 1996, pp. 82–88.

[64] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, American Association for Artificial Intelligence, 1996.

[65] M. Firat, B. Kaftanoglu, and O. Eser, *Sheet Metal Forming Analyses With An Emphasis On the Springback Deformation*, Journal of Materials Processing Technology **196** (2008), no. 1-3, 135–148.

[66] R.A. Fisher and F. Yates, *Statistical tables for biological,agricultural,and medical research: 6th ed rev and enl*, Hafoor, 1970.

[67] E. Fix and J.L Hodges, *Discriminatory Analysis, Non-parametric Discrimination: Consistency Properties*, Tech. report, USAF Scholl of aviation and medicine, Randolph Field, 1951, 4.

[68] P. Flach, *The Geometry of ROC space: Understanding Machine Learning Metrics through ROC Isometrics*, Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA, AAAI Press, 2003, pp. 194–201.

[69] P. Flach, *ROC analysis*, Encyclopedia of Machine Learning, Springer, 2010, pp. 869–875.

[70] S. Fleishman, D. Cohen-Or, and C. Silva, *Robust Moving Least-Squares Fitting With Sharp Features*, ACM transactions on graphics **24** (2005), no. 3, 544–552.

[71] J.D. Foley, A. Van Dam, J.F. Hughes, S.K. Feiner, and D.F. Sklar, *Computer Graphics: Principles and Practice*, ADDISON WESLEY Publishing Company Incorporated, 2013.

[72] E. Frank and I. Witten, *Generating Accurate Rule Sets Without Global Optimization*, ICML, 1998, pp. 144–151.

[73] E. Frank and I. H. Witten, *Making Better Use of Global Discretization*, Proc. of the Sixteenth International Conference on Machine Learning, 1999, pp. 115–123.

[74] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, *Knowledge Discovery in Databases: An Overview*, Knowledge Discovery in Databases, AAAI/MIT Press, 1991, pp. 1–30.

[75] J. H. Friedman, *Multivariate Adaptive Regression Splines*, Annals of Statistics (1991).

[76] M. Friedman, *A Comparison of Alternative Tests of Significance for the Problem of m Rankings*, The Annals of Mathematical Statistics **11** (1940), no. 1, 86–92.

[77] L. Fritsche, A. Schlaefer, K. Budde, K. Schrter, and H. Neumayer, *Research Paper: Recognition of Critical Situations from Time Series of Laboratory Results by Case-Based Reasoning.*, Journal of the American Medical Informatics Association **9** (2002), no. 5, 520–528.

[78] J. Fu, S. Joshi, and T. Simpson, *Shape differentiation of freeform surfaces using a similarity measure based on an integral of gaussian curvature*, Computer-Aided Design **40** (2008), no. 3, 311–323.

[79] J. Fürnkranz and A. Flach, *ROC 'n' Rule Learning: Towards A better Understanding of Covering Algorithms*, Machine learning **58** (2005), no. 1, 39–77.

[80] R. Gal and D. Cohen-or, *Salient Geometric Features for Partial Shape Matching and Similarity*, ACM Transactions on Graphics **25** (2006), 130–150.

[81] S. García, A. Fernández, J. Luengo, and F. Herrera, *A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability*, Soft Computing **13** (2009), no. 10, 959–977.

[82] S. García, D. Molina, M. Lozano, and F. Herrera, *A study on the use of non-parametric tests for analysing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimisation*, Journal of Heuristics **15** (2009), no. 6, 617–644.

[83] R. Giegerich, C. Meyer, and P. Steffen, *A Discipline of Dynamic Programming over Sequence Data*, Science of Computer Programming **51** (2004), no. 3, 215–263.

[84] T. Giorgino, *Computing and Visualizing Dynamic Time Warping Alignments in R: The DTW Package*, Journal of Statistical Software (2009).

[85] D. Gordon and R. A. Reynolds, *Image Space Shading of 3-Dimensional Objects*, Computer Vision, Graphics, and Image Processing **29** (1985), no. 3, 361–376.

[86] A. Göttmann, J. Diettrich, G. Bergweiler, M. Bambach, G. Hirt, P. Loosen, and R. Poprawe, *Laser-assisted Asymmetric Incremental Sheet Forming of Titanium Sheet Metal Parts.*, Production Engineering **5** (2011), no. 3, 263–271.

[87] S. Gumhold, X. Wang, and R. MacLeod, *Feature Extraction From Point clouds*, Proceedings of 10th International Meshing Roundtable, October 2001, pp. 293–305.

[88] Z. Guo, L. Zhang, and D. Zhang, *A Completed Modeling of Local Binary Pattern Operator for Texture Classification*, IEEE Transactions on Image Processing **19** (2010), no. 6, 1657 –1663.

[89] E. Guresen, G. Kayakutlu, and T. U. Daim, *Using Artificial Neural Network Models in Stock Market Index Prediction*, Expert Systems with Applications **38** (2011), no. 8, 10389–10397.

[90] A. Hadoush and A.H. van den Boogaard, *Efficient Implicit Simulation of Incremental Sheet Forming*, International Journal for Numerical Methods in Engineering **90** (2012), no. 5, 597–612.

[91] A. M. S. Hamouda, F. Abu Khadra, M. M. Hamadan, R. M. Imhemed, and E. Mahdi , *Springback in v-bending: a finite element approach*, International Journal of Materials and Product Technology **21** (2004), no. 1, 124–136.

[92] F. Han, J. Mo, H. Qi, R. Long, X. Cui, and Z. Li, *Springback Prediction for Incremental Sheet Forming Based on FEM-PSONN Technology*, Transactions of Nonferrous Metals Society of China **23** (2013), no. 4, 1061–1071.

[93] J. Han, *Data mining: Concepts and techniques*, Morgan Kaufmann Publishers Inc., 2005.

[94] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed., Morgan Kaufmann Publishers Inc., 2011.

[95] D. Hand and R. Till, *A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems*, Machine learning **45** (2001), no. 2, 171–186.

[96] D. J. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, 2001.

[97] J. Hanley and B. McNeil, *The Meaning and Use of the Area Under a Receiver Operating Characteristic(ROC) Curve*, Radiology **143** (1982), no. 1, 29–36.

[98] W. Hao and S. Duncan, *Optimization of Tool Trajectory for Incremental Sheet Forming Using Closed Loop Control*, Automation Science and Engineering (CASE), 2011 IEEE Conference on, 2011, pp. 779 –784.

[99] J.R. Hauser, *Numerical methods for nonlinear engineering models*, Springer, 2009.

[100] S.S. Haykin, *Neural networks: a comprehensive foundation*, Macmillan, 1994.

[101] G. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projection*, 2nd ed., Springer, 2009.

[102] D. Hilbert, *Ueber Die Stetige Abbildung Einer Line Auf Ein Flächenstück*, Mathematische Annalen **38** (1891), no. 3, 459–460.

[103] K. Hildebrandt, K. Polthier, and M. Wardetzky, *Smooth Feature Lines on Surface Meshes*, Proceedings of the third Eurographics symposium on Geometry processing, Eurographics Association, 2005.

[104] G. Hirt, J. Ames, M. Bambach, R. Kopp, and R. Kopp, *Forming Strategies and Process Modelling for {CNC} Incremental Sheet Forming*, CIRP Annals - Manufacturing Technology **53** (2004), no. 1, 203–206.

[105] C. Hoffmann and J. Rossignac, *A road map to solid modeling*, IEEE Transactions on Visualization and Computer Graphics **2** (1996), no. 1, 3–10.

[106] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Surface Reconstruction From Unorganized Points*, Computer graphics **26** (1992), no. 2, 71–78.

[107] A.S.M. Hoque, P.K. Halder, M.S. Parvez, and T. Szecsi, *Integrated Manufacturing Features and Design for Manufacture Guidelines for Reducing Product cost under CAD/CAM Environment*, Computers & Industrial Engineering **66** (2013), no. 4, 988–1003.

[108] H. Hsu, A. C. Yang, and M. Lu, *KNN-DTW Based Missing Value Imputation for Microarray Time Series Data*, Journal of Computers **6** (2011), no. 3, 418–425.

[109] J. Huang and C. Ling, *Using auc and accuracy in evaluating learning algorithms*, IEEE Transactions on Knowledge and Data Engineering **17** (2005), no. 3, 299–310.

[110] M. Inamdar, P. P. Date, K. Narasimhan, S. K. Maiti, and U. P. Singh, *Development of an artificial neural network to predict springback in air vee bending*, The International Journal of Advanced Manufacturing Technology **16** (2000), no. 5, 376–381.

[111] *Tecnalia. Inspiring Business*, `http://www.tecnalia.com/`, January 2014, [Online; accessed Feb. 08, 2014].

[112] *The Innovative MAnufacturing of complex Ti sheet components (INMA)*, `http://www.inmaproject.eu/`, January 2014, [Online; accessed 27-Jan-2014].

[113] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, McGraw-Hill, Inc., 1995.

[114] J. Jeswiet, M. Geiger, U. Engel, M. Kleiner, M. Schikorra, J. Duflou, R. Neugebauer, P. Bariani, and S. Bruschi, *MetalFforming Progress Since 2000*, CIRP Journal of Manufacturing Science and Technology **1** (2008), no. 1, 2–17.

[115] J. Jeswiet, F. Micari, G. Hirt, A. Bramley, J. Duflou, and J. Allwood, *Asymmetric Single Point Incremental Forming of Sheet Metal*, CIRP Annals - Manufacturing Technology **54** (2005), no. 2, 88–114.

[116] G. John and P. Langley, *Estimating Continuous Distributions in Bayesian Classifiers*, Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1995, pp. 338–345.

[117] N.L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, no. 2, Wiley & Sons, 1995.

[118] H. Kaganami and Z. Beiji, *Region-Based Segmentation versus Edge Detection*, Proceedings of the 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE Computer Society, 2009, pp. 1217–1221.

[119] A. P. Karafillis and M. C. Boyce, *Tooling and Binder Design for Sheet Metal Forming Processes Compensating Springback Error*, International Journal of Machine Tools and Manufacture **36** (1996), no. 4, 503–526.

[120] A. E. Kaufman, *Voxels as a Computational Representation of Geometry*, in The Computational Representation of Geometry. SIGGRAPH '94 Course Notes, 1994, p. 45.

[121] J.F. Kenney and E.S. Keeping, *Mathematics of statistics*, no. pt. 2, Van Nostrand, 1947.

[122] E. Keogh and C. A. Ratanamahatana, *Exact Indexing of Dynamic Time Warping*, Knowledge and Information Systems **7** (2005), no. 3, 358–386.

[123] E. J. Keogh and M. J. Pazzani, *Scaling Up Dynamic Time Warping for Data Mining Applications*, KDD, 2000, pp. 285–289.

[124] E. J. Keogh and M. J. Pazzani, *Derivative Dynamic Time Warping*, In First SIAM International Conference on Data Mining (SDM2001, 2001.

[125] M. Khan, F. Coenen, C. Dixon, and S. El-Salhi, *Finding Correlations Between 3-d Surfaces: A study in Asymmetric Incremental Sheet Forming*, Proc. Machine Learning and Data Mining in Pattern Recognition (MLDM'12), Springer, 2012, pp. 336–379.

[126] D. J. Kim and B. M. Kim, *Application of Neural Network and FEM for Metal Forming Processes*, International Journal of Machine Tools and Manufacture **40** (2000), no. 6, 911–925.

[127] L. Kobbelt and M. Botsch, *A survey of Point-Based Techniques in Computer Graphics*, Computers and graphics **28** (2004), no. 6, 801–814.

[128] S. Kotsiantis and D. Kanellopoulos, *Discretization techniques: A recent survey*, **32** (2006), 47–58.

[129] V. Krishnamurthy and M. Levoy, *Fitting Smooth Surfaces to Dense Polygon Meshes*, Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, ACM, 1996, pp. 313–324.

[130] J.-F. Lalonde, R. Unnikrishnan, N. Vandapel, and M. Hebert, *Scale Selection for Classification of Point-Sampled 3D Surfaces*, 3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on, 2005, pp. 285–292.

[131] D. Lay, *Linear Algebra and Its Applications*, 4th ed., Pearson, 2010.

[132] E. Leszak, *Apparatus and Process for IncrementalDieless Forming*, September 1967.

[133] F. W. B. Li, *Parametric surface rendering*, Wiley Encyclopedia of Computer Science and Engineering, 2008.

[134] K.P. Li, W.P. Carden, and R.H. Wagoner, *Simulation of Springback*, International Journal of Mechanical Sciences **44** (2002), no. 1, 103–122.

[135] C. Ling, J. Huang, and H. Zhang, *AUC: A better Measure Than Accuracy in Comparing Learning Algorithms*, Proceedings of the 16th Canadian society for computational studies of intelligence conference on Advances in artificial intelligence, Springer, 2003, pp. 329–341.

[136] R. Lingbeek, J. Hutink, S. Ohnimus, M. Petzoldt, and J. Weiher, *The development of a finite elements based springback compensation tool for sheet metal products*, Journal of Materials Processing Technology **169** (2005), no. 1, 115–125.

[137] R.A. Lingbeek, W. Gan, R.H. Wagoner, T. Meinders, and J. Weiher, *Theoretical verification of the displacement adjustment and springforward algorithms for springback compensation*, International Journal of Material Forming **1** (2008), no. 3, 159–168.

[138] H. Liu, F. Hussain, C. Tan, and M. Dash, *Discretization: An Enabling Technique*, Data Mining and Knowledge Discovery **6** (2002), no. 4, 393–423 (English).

[139] J. Liu, A. Jakas, A. Al-Obaidi, and Y. Liu, *A comparative study of different corner detection methods*, Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on, dec. 2009, pp. 509 –514.

[140] W. Liu, Z. Liang, T. Huang, Y. Chen, and J. Lian, *Process Optimal Ccontrol of Sheet Metal Forming Springback Based on Evolutionary Strategy*, Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on, June 2008, pp. 7940 –7945.

[141] Y. C. Liu, *The effect of restraining force on shape deviations in flanged channels*, Journal of Engineering Materials and Technology **110** (1988), no. 4, 389–394.

[142] N. Loménie, D. Racoceanu, and G. Stamon, *Point set analysis: An image analysis point of view for rapid prototyping technologies*, ed. m.e. hoque ed., vol. 1, InTech, 07/2011 2011.

[143] W. E. Lorensen and H. E. Cline, *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, SIGGRAPH Computer Graphics **21** (1987), no. 4, 163–169.

[144] G. Lu and A. Sajjanhar, *Region-based Shape Representation and Similarity Measure Suitable for Content-based Image Retrieval*, Multimedia Systems **7** (1999), no. 2, 165–174.

[145] D. J. Maguire, *The Raster GIS Design Model – a profile of ERDAS*, Computers & Geosciences **18** (1992), no. 4, 463–470.

[146] J. Mah, C. Samson, S. D. McKinnon, and D. Thibodeau, *3D Laser Imaging for Surface Roughness Analysis*, International Journal of Rock Mechanics and Mining Sciences **58** (2013), 111–117.

[147] O. Maimon and L. Rokach (eds.), *Data Mining and Knowledge Discovery Handbook, 2nd ed*, Springer, 2010.

[148] K. Manabe, M. Yang, and S. Yoshihara, *Artificial Intelligence Identification of Process Parameters and Adaptive Control System for Deep-Drawing Process*, Journal of Materials Processing Technology **80 - 81** (1998), no. 0, 421–426.

[149] Z. Marciniak, J. L. Duncan, and S.J. Hu, *Mechanics of sheet metal forming*, Butterworth-Heinemann, 2002.

[150] T. Meinders, I.A. Burchitz, M.H.A. Bonte, and R.A. Lingbeek, *Numerical Product Design: Springback Prediction, Compensation and Optimization*, International Journal of Machine Tools and Manufacture **48** (2008), no. 5, 499–514, Advances in Sheet Metal Forming Applications.

[151] Z. Meng, J. Pan, K. Tseng, and C. Hsu, *Corner detection based on normal vector of boundary fitting line*, International Conference on Genetic and Evolutionary Computing (2010), 756–759.

[152] M. Müller, *Information Retrieval for Music and Motion*, Springer, 2007.

[153] N. Narasimhan and M. Lovell, *Predicting Springback in Sheet Metal Forming: An Explicit to Implicit Sequential Solution Procedure*, Finite Elements in Analysis and Design **33** (1999), no. 1, 29–42.

[154] V. Nasrollahi and B. Arezoo, *Prediction of Springback in Sheet Metal Components With Holes on the Bending Area, Using Experiments, Finite Element and Neural Networks*, Materials and Design **36** (2012), 331–336.

[155] P. Nemenyi, *Distribution-free multiple comparisons*, Ph.D. thesis, 1963.

[156] V. Niennattrakul and C. A. Ratanamahatana, *Learning DTW Global Constraint for Time Series Classification*, Computing Research Repository **abs/0903.0041** (2009).

[157] A. Nilsson, L. Melin, and C. Magnusson, *Finite-Element Simulation of V-die Bending: A Comparison with Experimental Results*, Journal of Materials Processing Technology **65** (1997), no. 1-3, 52–58.

[158] P. Ning and J. Bloomenthal, *An Evaluation of Implicit Surface Tilers*, Computer Graphics and Applications **13** (1993), no. 6, 33–41.

[159] T. Ojala, M. Pietikainen, and T. Maeenpaa, *Multiresolution Gray-scale and Rotation Invariant Texture Classification with Local Binary Patterns*, IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002), no. 7, 971–987.

[160] U. Orguner and F. Gustafsson, *Statistical Characteristics of Harris Corner Detector*, IEEE Workshop on Statistical Signal Processing, Aug 2007, pp. 571–575.

[161] C. Oztireli, G. Guennebaud, and M. Gross, *Feature preserving point set surfaces based on non-linear kernel regression*, Eurographics, 2009.

[162] D. L. Page, A. F. Koschan, and M. A. Abidi, *Perception-based 3D Triangle Mesh Segmentation Using Fast Marching Watersheds*, in Proceedings of the International Conference on Computer Vision and Pattern Recognition, II, 2003, pp. 27–32.

[163] D. L. Page, Y. Sun, A. F. Koschan, J. Paik, and M. A. Abidi, *Normal Vector Voting: Crease Detection and Curvature Estimation on Large, Noisy Meshes*, Graphical Models **64** (2002), no. 3-4, 199–229.

[164] S.M. Pandit and S.M. Wu, *Time Series and System Analysis With Applications*, Krieger Publishing Company, 2000.

[165] M. Pauly, R. Keiser, and M. Gross, *Multi-Scale Feature Extraction on Point-Sampled Surfaces*, Computer Graphics Forum, vol. 22, Blackwell Synergy, 2003, pp. 281–289.

[166] G. Peano, *Sur Une Courbe, Qui Remplit Toute Une Aire Plane*, Mathematische Annalen **36** (1890), no. 1, 157–160 (French).

[167] G. Piatetsky-Shapiro and W. J. Frawley, *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991.

[168] T. Pinto, C. Kohler, and A. Albertazzi, *Regular Mesh Measurement of Large Free Form Surfaces Using Stereo Vision and Fringe Projection*, Optics and Lasers in Engineering **50** (2012), no. 7, 910–916.

[169] F. Pourboghrat and E. Chu, *Springback in plane strain stretch/draw sheet forming*, International Journal of Mechanical Sciences **37** (1995), no. 3, 327–341.

[170] J. M. S. Prewitt, *Object Enhancement and Extraction*, Picture Processing and Psychopictorics (B. S. Lipkin and A. Rosenfeld, eds.), Academic Press, 1970, pp. 75–149.

[171] K. K. Pringle, *Visual Perception by a Computer*, Automatic Interpretation and Classification of Images (A. Grasselli, ed.), Academic Press, New York, 1969, pp. 277–284.

[172] S. Pulla, A. Razdan, and G. Farin, *Improved curvature estimation for watershed segmentation of 3-dimensional meshes*, IEEE Transactions on Visualization and Computer Graphics **2002** (2001).

[173] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[174] A. Ricci, *A Constructive Geometry for Computer Graphics*, The Computer Journal **16** (1973), no. 2, 157–160.

[175] J. Ringenberg, M. Deo, V. Devabhaktuni, O. Berenfeld, B. Snyder, P. Boyers, and J. Gold, *Accurate Reconstruction of 3D Cardiac Geometry from Coarsely-Sliced MRI*, Computer Methods and Programs in Biomedicine **113** (2014), no. 2, 483–493.

[176] A. Rockwood, *The displacement method for implicit blending surfaces in solid models*, ACM Transactions on Graphics **8** (1989), no. 4, 279–297.

[177] R. Rojas, *Neural networks: A systematic introduction*, Springer, 1996.

[178] A Rosochowski, *Die compensation procedure to negate die deflection and component springback*, Journal of Materials Processing Technology **115** (2001), no. 2, 187–191.

[179] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Representations by Back-Propagating Errors*, Nature **323** (1986), no. 6088, 533–536.

[180] *IBF institute of metal forming*, `http://www.ibf.rwth-aachen.de/en/`, January 2014, [Online; accessed 27-Jan-2014].

[181] H. Sakoe and S. Chiba, *Dynamic Programming Algorithm Optimization for Spoken Word Recognition*, IEEE Transactions on Acoustics, Speech and Signal Processing **26** (1978), no. 1, 43–49.

[182] H. Sakoe and S. Chiba, *Readings in speech recognition*, Morgan Kaufmann Publishers Inc., 1990, pp. 159–165.

[183] D. Salomon, *Curves and Surfaces for Computer Graphics*, Springer, 2006.

[184] S. Salvador and P. Chan, *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space*, Intelligent Data Analysis **11** (2007), no. 5, 561–580.

[185] Z. Shaozuo, L. Chao, P. Yinghong, L. Dayong, and Y. Hongbo, *Study on Factors Affecting Springback and Application of Data Mining in Springback Analysis*, Journal of Shanghai Jiaotong University **2** (2003), 192–196.

[186] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*, 4 ed., Chapman & Hall/CRC, 2007.

[187] B. W. Silverman and M. C. Jones, *E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951)*, International Statistical Review / Revue Internationale de Statistique **57** (1989), no. 3, 233–238.

[188] P. Steffen, R. Giegerich, and M. Giraud, *GPU Parallelization of Algebraic Dynamic Programming*, Proceedings of the 8th International Conference on Parallel Processing and Applied Mathematics: Part II, Springer, 2010, pp. 290–299.

[189] M. Stone, *Cross-Validatory Choice and Assessment of Statistical Predictions*, Journal of the Royal Statistical Society. Series B (Methodological) **36** (1974), no. 2, pp. 111–147.

[190] M. Strano, *Technological Representation of Forming Limits for Negative Incremental Forming of Thin Aluminum Sheets*, Journal of Manufacturing Processes **7** (2005), no. 2, 122–129.

[191] S. Sumathi and S.N. Sivanandam, *Introduction to Data Mining and Its Applications*, Springer, 2006.

[192] W. Sun, C. Bradley, Y. Zhang, and H. Loh, *Cloud Data Modelling Employing a Unified, Non-redundant Triangular Mesh*, Computer-Aided Design **33** (2001), no. 2, 183–193.

[193] Y. Sun, D.L. Page, J.K. Paik, A. Koschan, and M.A. Abidi, *Triangle mesh-based edge detection and its application to surface segmentation and adaptive surface smoothing*, IEEE International Conference on Image Processing, vol. 3, June 2002, pp. 825–828 vol.3.

[194] M. Sunseri, J. Cao, A. P. Karafillis, and M. C. Boyce, *Accommodation of springback error in channel forming using active binder force control: Numerical simulations and experiments*, Journal of Engineering Materials and Technology **118** (1996), no. 3, 426–435.

[195] P. Tan, M.Steinbach, and V. Kumar, *Introduction to Data Mining, (first edition)*, Addison-Wesley, May 2005.

[196] C. Tang and G. Medioni, *Inference of Integrated Surface, Curve, and Junction Descriptions From Sparse 3D Data*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 11, 1206–1223.

[197] L. Tapie, B. Mawussi, and A. Bernard, *Topological Model for Machining of Parts with Complex Shapes*, Computers & Industrial Engineering **63** (2012), no. 5, 528–541.

[198] G. Taubin, *Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **13** (1991), no. 11, 1115–1138.

[199] A. Taylan and A.E. Tekkaya, *Sheet Metal Forming: Processes and Applications*, ASM International, 2012.

[200] A. Telea, *Data Visualization - Principles and Practice.*, A K Peters, 2008.

[201] G. Tognola, M. Parazzini, G. Pedretti, P. Ravazzani, C. Svelto, M. Norgia, and F. Grandori, *Three-Dimensional Reconstruction and Image Processing in Mandibular Distraction Planning*, IEEE Transactions on Instrumentation and Measurement **55** (2006), no. 6, 1959 –1964.

[202] S. Tsumoto, *Contingency matrix theory: Statistical dependence in a contingency table*, Information Sciences **179** (2009), no. 11, 1615–1627.

[203] A. Udomchaiporn, F. Coenen, M. Garca-Fiana, and V. Sluming, *3-D MRI Brain Scan Feature Classification Using an Oct-Tree Representation*, Advanced Data Mining and Applications, vol. 8346, Springer, 2013, pp. 229–240.

[204] A. H. van den Boogaard, T. Meinders, and J. Hutink, *Efficient Implicit Finite Element Analysis of Sheet Forming Processes*, International Journal for Numerical Methods in Engineering **56** (2003), no. 8, 1083–1107.

[205] T. Varady, R. Martin, and J. Cox, *Reverse Engineering of Geometric Models - An Introduction.*, Computer-Aided Design **29** (1997), no. 4, 255–268.

[206] D. Varberg, E. Purcell, and S. Rigdon, *Calculus*, 9th ed., Pearson, 2005.

[207] N. Vasconcelos and A. Lippman, *Feature representations for image retrieval: beyond the color histogram*, Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on, vol. 2, 2000, pp. 899–902 vol.2.

[208] J. Vince, *Introduction to the Mathematics for Computer Graphics*, 3rd ed., Springer, 2010.

[209] J.F Wang, R.H Wagoner, W.D Carden, D.K Matlock, and F Barlat, *Creep and anelasticity in the springback of aluminum*, International Journal of Plasticity **20** (2004), no. 12, 2209–2232.

[210] C. Weber, S. Hahmann, and H. Hagen, *Methods for Feature Detection in Point Clouds*, Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop), March, 2010, vol. 19, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, pp. 90–99.

[211] L. Wei and E. Keogh, *Semi-supervised Time Series Classification*, Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 748–753.

[212] K. Q. Weinberger and L. K. Saul, *Distance Metric Learning for Large Margin Nearest Neighbor Classification*, Journal of machine learning research **10** (2009), 207–244.

[213] M. A. Wesley, T. Lozano-Pérez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman, *A Geometric Modeling System for Automated Mechanical Assembly*, IBM Journal of Rsearch and Development **24** (1980), no. 1, 64–74.

[214] B. Widrow, D. E. Rumelhart, and M. A. Lehr, *Neural Networks: Applications in Industry, Business and Science*, Communications of the ACM **37** (1994), no. 3, 93–105.

[215] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2 ed., Elsevier Science, 2005, (software downloaded from http://www.cs.waikato.ac.nz/ml/weka/).

[216] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motodai, G. McLachlan, A. Ng, B. Liu, P. Yu, Z. Zhou, M. Steinbach, D. Hand, and D. Steinberg, *Top 10 Algorithms in Data Mining*, Knowledge and information systems **14** (2007), no. 1, 1–37.

[217] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. Ratanamahatana, *Fast Time Series Classification Using Numerosity Reduction*, Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006, pp. 1033–1040.

[218] W.L. Xu, C.H. Ma, C.H. Li, and W.J. Feng, *Sensitive Factors in Springback Simulation for Sheet Metal Forming*, Journal of Materials Processing Technology **151** (2004), 217–222.

[219] J. Yin, D. Li, Y. Wang, and Y. Peng, *Knowledge Discovery from Finite Element Simulation Data*, Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, vol. 3, Aug 2004, pp. 1335–1340.

[220] A. Yogeswaran and P. Payeur, *Features Extraction from Point Clouds for Automated Detection of Deformations on Automotive Body Parts*, IEEE International Workshop on Robotic and Sensors Environments, 2009, pp. 122 –127.

[221] D. Yoo, *Three-Dimensional Surface Reconstruction of Human Bone Using a B-Spline Based Interpolation Approach*, Computer-Aided Design **43** (2011), no. 8, 934–947.

[222] J. Yoon, F. Pourboghrat, K. Chung, and D. Yang, *Springback Prediction For Sheet Metal Forming Process Using a 3d Hybrid Membrane/Shell Method*, International Journal of Mechanical Sciences **44** (2002), no. 10, 2133–2153.

[223] D. Yu, X. Yu, Q. Hu, J. Liu, and A. Wu, *Dynamic Time Warping Constraint Learning for Large Margin Nearest Neighbor Classification*, Information Sciences **181** (2011), no. 13, 2787–2796.

[224] J. Yu and J. Amores, *Algorithmic Aspects of Treewidth*, Journal of Algorithms **7** (1986), 309–322.

[225] J. Yu, J. Amores, N. Sebe, P. Radeva, and Q. Tian, *Distance Learning for Similarity Estimation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **30** (2008), no. 3, 451–462.

[226] W. Yu, F. Coenen, M. Zito, and S. El-Salhi, *Minimal Vertex Unique Labelled Subgraph Mining*, International Conference on Data Warehousing and Knowledge Discovery, 2013, pp. 317–326.

[227] W. Yu, F. Coenen, M. Zito, and S. El-Salhi, *Vertex Unique Labelled Subgraph Mining*, International Conference of the British Computer Society's Specialist Group on Artificial Intelligence, 2013, pp. 21–37.

[228] W. Yu, F. Coenen, M. Zito, and S. El-Salhi, *Vertex unique labelled subgraph mining for vertex label classification*, International Conference on Advanced Data Mining and Applications, vol. 8346, Springer, 2013, pp. 542–553.

[229] C. Zhang and T. Chen, *Efficient feature extraction for 2d/3d objects in mesh representation*, Image Processing, 2001. Proceedings. 2001 International Conference on, vol. 3, 2001, pp. 935–938 vol.3.

[230] D. Zhang, Z. Cui, X. Ruan, and Y. Li, *An Analytical Model for Predicting Springback and Side Wall Curl of Sheet After u-bending*, Computational Materials Science **38** (2007), no. 4, 707–715.

[231] W. Zhao, C. Zhao, Y. Wen, and S. Xiao, *An Adaptive Corner Extraction Method of Point Cloud for Machine Vision Measuring System*, Proceedings of the 2010 International Conference on Machine Vision and Human-machine Interface, IEEE Computer Society, 2010, pp. 80–83.

[232] E. Zuckerberger, A. Tal, and S. Shlafman, *Polyhedral surface decomposition with applications*, Computers & Graphics **26** (2002), no. 5, 733–743.