A Multi-Sensor Simulation Environment for Autonomous Cars

Rui Song[1], Paul Horridge[1], Simon Pemberton[2], Jon Wetherall[2], Simon Maskell[1], Jason Ralph[1]

[1] *Dept. Electrical Engineering and Electronics University of Liverpool* Liverpool, UK
{rui.song, p.horridge, s.maskell, jfralph}@liverpool.ac.uk
[2] *CGA Simulation,* Liverpool, UK
simon_pemberton@yahoo.co.uk, jon@cgasimulation.com,

***Abstract*—This paper describes a multi-sensor simulation environment. This environment is being used to develop tracking methods to improve the accuracy of environmental perception and obstacle detection for autonomous vehicles. The system is being developed as part of a collaborative project entitled: Artificial Learning Environment for Autonomous Driving (ALEAD). The system currently incorporates a range of different sensor models, such as camera, infrared (IR) camera and LiDAR, with radar and GNSS-aided navigation systems to be added at a later stage. Each sensor model has been developed to be as realistic as possible – incorporating physical defects and other artefacts found in real sensors. This paper describes the environment, sensors and demonstrates the use of a Kalman filter based tracking algorithm to fuse data to predict the trajectories of dynamic obstacles. The multi-sensor tracking system has been tested to track a ball bouncing in a 3D environment constructed using Unity3D software.**

Keywords—multi sensors, autonomous driving, visual tracking, virtual environment

I. Introduction

The move towards autonomous vehicles offers a number of potential benefits; by reducing the number of accidents caused by inattentive drivers, decreasing traffic congestion, lowering emissions, and improving mobility especially for elder and disabled people [1, 2]. Companies such as Tesla and Volvo are actively working on creating fully autonomous vehicles that can plan routes, drive and navigate. This is a difficult problem in carefully controlled environments, but – in the real world – autonomous cars also need to be able to detect unpredictable events and to react appropriately in all circumstances, even when the sensor performance is degraded due to environmental effects (weather, dirt, and possible damage). Such unexpected events could include objects, people or animals entering the roadway unexpectedly, or from behind obstructions. For instance, a fatal crash happened when the self-driving car failed to see a road separator [3]. The ability to deal with real life problems is critical to the safe operation of autonomous vehicles. However, because of the cost and the risks of testing autonomous cars in a real environment, it is common for companies to train algorithms using simulations to improve the autonomy level of cars before putting them into service on the road.

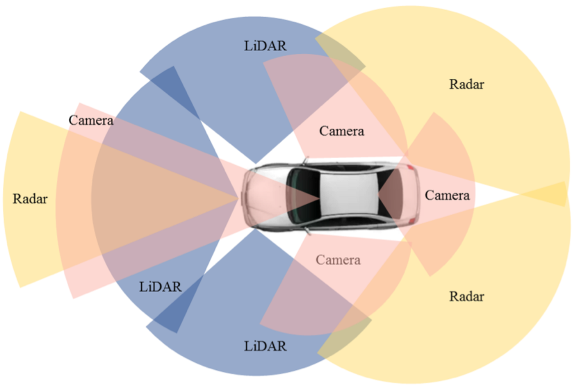


Fig. 1. Typical sensors on autonomous self-driving car

Traditionally, simulations have been developed by importing sensor data that cars have collected in the real world and then using this data to verify that the car's self-driving software is capable of dealing with emergency situations appropriately. For example, a vehicle's cameras can record video of other cars on road and pedestrians crossing the street. The recorded video can be used every time when updating the self-driving software and prove that the software can still detect targets correctly [4, 5].

In recent years, more advanced simulation tools are able to build entire road networks [6, 7], where the autonomous car is surrounded by many other vehicles, bicycles, pedestrians and even animals. Developers can use these constructed virtual worlds to test and retest various scenarios found in everyday life and identify potential problems. In such a way, the stability and robustness of the self-driving software can be improved. Most automotive companies therefore spend most of the time building the simulation environment to be as realistic as possible. However, a highly detailed simulated environment on its own does not provide a complete representation of the real world. Accurate sensor models, visual perception and intelligent guidance systems are also important for sensing surroundings and to avoid collisions. These models should include naturally occurring artefacts and the limitations present in real physical sensors.

As shown in Fig 1, autonomous cars rely on multiple sensors; including cameras, LiDAR, and radar. One of the advantages of using multiple sensors is that it can improve the robustness of the sensing system if one or more sensors malfunction. Some recent work has demonstrated the capabilities of using multiple sensors in 3D simulation environments. [8] used 6 cameras to obtain sparse 3D maps with a full 360^{o} field of view for visual navigation and car localization. [9] integrated cameras and a LiDAR sensor using a mature and open-source simulator called Gazebo. Gazebo, however is mostly supported on Linux and it is not well designed for hybrid simulation. To address these challenges, a unified system for automation and robotics simulator (USARSim) has been gaining the interest of researchers, such as in [10]. The disadvantage of USARSim is that it is not suitable for real-time tests, especially when the 3D environment is complex. Currently, most of the simulation systems that have been developed have yet to be improved to the point where they can be physically realistic. Firstly, the environments need to be modelled to include dynamic artefacts and the 'dirt' associated with the real world (including physical dirt affecting sensor performance and minor damage due to prolonged use). Secondly, sensors, such as cameras and LiDARs, are required to run at a high-frequency and have a high level of fidelity in a dense and harsh environment. Thirdly, for industrial applications, a complete integrated simulation system is required, including different types sensing modelling and navigation algorithms. To address these problems, the Artificial Learning Environment for Autonomous Driving (ALEAD) project is working to train vehicles by developing a hybrid 3D simulator based on Unity3D and ROS. Real world inputs will be used to construct an extensive simulation environment. Representative sensor models for each of the key sensors shown in Fig.1 are being integrated into the virtual environment; including GPS and other satellite navigation systems.

In this paper, the initial design of the ALEAD project is presented. A camera, LiDAR and IR camera have been modelled as part of the multi-sensing system. To verify the sensor modelling performance, a scene with a ball bouncing in a street has been simulated in Unity3D. A tracking algorithm has been developed and implemented to detect the movement of the bouncing ball and to separate it from the surrounding clutter. The paper is organized as follows. Section II describes the details of the ALEAD project and the multi-sensor system. Section III and IV explain the methods used in the sensor simulations and ball tracking, respectively. Results are shown and discussed in Section V. The paper is summarized, and conclusions drawn in Section VI.

II. ALEAD Project

A. Project Overview

ALEAD is a digital environment that provides autonomous vehicles a virtual space to learn in and so saves money and time for the companies developing those systems. The ALEAD project is developing a simulation environment for the development and testing of autonomous vehicles. It is being based around industry standard software components, including the Unity3D graphics engine, and is interfacing with the Robot Operating System (ROS) [11] and autonomous car models, including the Baidu Apollo [12] open driving solution.

Existing computer game simulation technologies developed by the industrial partner CGA are being applied to autonomous vehicle training, using novel improvements on existing simulations systems and applying these systems in a new sector. By using artificial intelligence and machine learning to train vehicles in an extensive simulation environment designed with real world inputs and benefiting from the integration of multiple sensors, ALEAD is combining technologies to create a wholly new environment which could have a significant impact on the time required to get autonomous vehicles on the road. ALEAD focuses on the merging fields of machine learning, virtual reality, augmented reality in the context of realistic simulations of urban environments.

ROS is an open-source framework designed to provide an abstraction layer to complex robotic hardware and software configurations. A variety of libraries and tools are available to help software developers create robot applications and has found wide use in both industry and academia. Moreover, ROS can provide simultaneous operation between hardware devices for actual autonomous vehicles. Unity3D [13] is a more flexible and powerful development platform for creating multiplatform 3D and 2D games and interactive experiences. This game engine, which supports almost every platform, is chosen to be the simulation server. By applying the transmitting interface, a communication module and detailed environment and sensor modelling techniques, the ROS and Unity3D will be incorporated together for real-time autonomous driving simulations.

The ALEAD project will significantly reduce the need for live trials of autonomous vehicles. Using a large number of parallel simulation/training environments, it will be possible to train systems at a rate millions of times faster than running live trials. It is also possible to train for exceptional weather conditions such as fog or ice.

B. Sensor Suite

Current testing systems mainly use video information and live trials. The key to simulating the environment in as realistic way as possible will be the use of physically realistic sensor models and environmental factors. The ALEAD system is developing representative sensor models for each of the key sensors that are likely to be present in future autonomous vehicles, including short range Radar, IR/TV cameras, LiDAR scanners, and GPS. The aim is to identify the factors that determine or limit sensor performance, thereby having an adverse effect on the robustness and safety of an autonomous vehicle: including, precipitation and other atmospheric effects, such as high humidity or fog, bright sources of illumination, such as the sun being low in the sky and reflections from buildings, erratic behavior from other road users, debris in the road, and deliberate jamming of the sensor data. The sensor modelling will make the training physically realistic for computer vision, which operates very differently from human perception.

- Image (Visible Band): A standard visible band camera model, which uses the simple scene as a basis. The angle of the field of view will be defined based on the interface with the coverage of other sensors.

- Image (Infrared Band): A thermal image of the scene based on the three-dimensional geometry of the scene, and requires the objects within the scene to be labelled with temperature information. Also requires some indication of the atmospheric properties to derive path radiance and attenuation properties.
- Lidar (Near Infrared Band): An active near visible band sensor, which measures the time of flight of pulsed light to build up a three-dimensional map of the scene. The scanning processes will be presented while vehicle is in motion and reflection of light from the surfaces of objects in the scene.
- Radar (Short Range mmW or cmW radar): Simple distance measuring device with relatively broad beamwidth and short range. The modelling will be stochastic/statistical in nature and will represent the distribution of radar scatterers in the scene and their statistical properties (standard Radar models).
- GPS/GNSS Satellite Navigation Systems: Basic radio navigation system based on very low power satellite signals. Requires WGS'84 Earth model information for realistic satellite data, including an interface for live GPS/GNSS feed or recorded satellite ephemeris data (e.g. RINEX format).
- Weather effects: to include the effects of different weather conditions on the car and the sensors, e.g. the effect of fog and rain on visibility or ice on the road affecting driving conditions.

III. Sensor Simulation

In this paper, all hardware sensors mounted in the autonomous car are modelled in the simulator, including the camera, IR camera and LiDAR.

A. *Camera*

The main purpose of the camera is to detect any moving target. As Unity3D is a game engine, the development of camera sensor can be used directly from the original camera of Unity3D. To accelerate processing, only objects near the camera are rendered. In such a way, the output of the camera only includes these rendered objects and objects out of range are ignored. In the paper, one camera is mounted statically on the top front of autonomous car. The field of view (FOV) of the camera is set as 36^o, and perspective projection is used. Fig.2 shows the rendering of the camera model in a simulated scene, where roads, houses, trees and street lamps are included.

B. *IR camera*

The infrared scene is based on the 3D scene rendered for the visible band camera, since the physical objects are the same in each case. However, the scene also includes a temperature map that allocates temperature profiles to the different surfaces present in the visible band scene. The IR camera model utilizes this temperature map and converts the temperatures into thermal intensities/photon fluxes [14]. The thermal radiation is propagated through an atmospheric model (including attenuation and path radiance) and then detected using a bespoke infrared camera model with properties representative of a commercial infrared camera (pixel non-uniformities, limited pixel resolution, dead/saturated pixels, etc.) [15]. The position of the IR camera is set beside the camera and the FOV is set as 40^o [15]. Fig.3 shows the temperature map and the corresponding IR image.

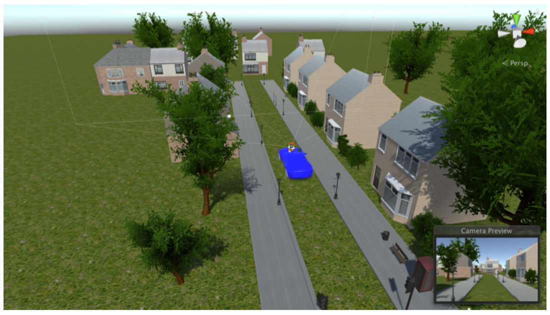


Fig. 2. Rendering of the camera model in a constructed scene.

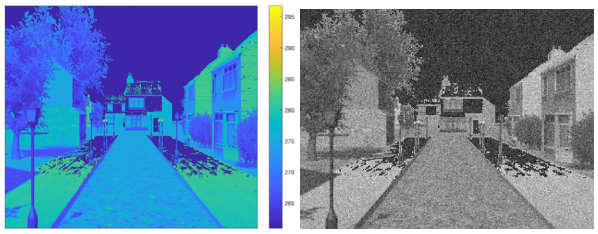


Fig. 3. Left: Rendering of the temperature map cooresponding to the FOV of the IR camera. The colour changes from blue to yellow while temperature increaes. Right: Rendering of the image output from the IR camera.

C. *LiDAR*

A commercial LiDAR sensor is simulated to ensure as realistic a representation as possible. The Velodyne HDL-64E [16], a vertical LiDAR sensor is used because it is the most popular type used in self-driving cars [17]. To simulate this type of LiDAR sensor, parameters such as number of lasers, position of each individual laser and angle, and the rotational speed should be considered. The specification of the Velodyne HDL-64E and its parameters are listed in TABLE I.

In Unity3D, each laser can be represented by using ray-casting. From the mathematical perspective, ray-casting is a directional 3D vector, which checks for intersections with other geometries. The coordinate of the intersected point will be sent back. In such a way, the ray-casting can be considered as a realistic representation of a laser. Note that, it requires to create collider for each object built in the constructed scene. Unity3D uses the physics engine to handle ray-casting. Multiple ray-casts can be executed within a single physics frame. In this way, it can provide simultaneous actions. Fig. 4 shows the LiDAR scanning in the scene.

TABLE I. LIDAR PARAMETERS

Parameter	***Value***
Rotational speed	10 Hz
Number of lasers	64
Max distance	200 m
Angle between scan	0.9^o
Vertical FOV	26.9^o

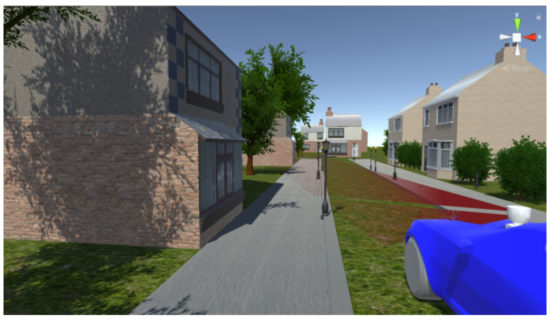


Fig. 4. LiDAR sensor scanning in the scene, laser beams are represented by red lines.

IV. Tracking Algorithm

After receiving the data from camera, IR camera LiDAR respectively, a Kalman filter based algorithm is used to track the trajectory of the dynamic obstacles. The object (or objects) are detected in the respective fields of view using a frame difference method – which will also generate some background detections from static objects as the vehicle moves forwards. Some, but not all, of these static objects can be removed by registering the images before frame differencing. However, some static background clutter can also useful to distinguish moving objects as outliers in the tracking system – i.e. the moving objects are those that have significant differences to the static background. The measurements associated with the detections from each frame are then associated with existing tracks using a standard nearest neighbor approach, or new tracks are created when something has moved into the field of view. Once the measurements have been associated, a standard Kalman filter is used to update and predict the track motion.

A. *Obstacle Detection*

Before the tracking, the positions of moving obstacles are detected from the measured data. For received camera and IR image, the obstacles are detected by using the frame difference method [18]. In this method, the area of moving obstacle (area of significant difference) is enlarged first, and the center of the obstacle (centroid position) is extracted thereafter.

The LiDAR scanning results are represented by point clouds, where the world coordinates, spherical coordinates, and laser number are recorded. When detecting the position of obstacle, a multi-feature extraction method used in [19] is applied to the plotting of the point clouds.

B. *Track Association and Track Maintenance*

Typically, for the results shown here, up to ten objects are detected and tracked by the sensors, and only one of these is in motion. The apparent motion of the other objects comes from the forward motion of the car (assumed to be approximately 10 meters/second for the examples shown here). Each measurement is associated with a track based on the simple Euclidean distance between the two, in the sensor axes. For the low clutter examples considered in this paper, this method is sufficiently robust to allow accurate object tracking.

C. *Obstacle Tracking*

After detection and association, the measured positions of the obstacles obtained from the sensors data are fused into the tracking algorithm. A standard Kalman filter [20] has been used in the results presented below. Several variants of the standard particle (bootstrap) filter [21, 22] were also tested but little benefit was found in using these rather than the Kalman filter.

The state vector for this model contains position, velocity and acceleration, and is given by,

$$\underline{X}(t) = (x \quad v_x \quad a_x \quad y \quad v_y \quad a_y \quad z \quad v_z \quad a_z)^T \tag{1}$$

where the co-ordinates (x, y, z) are in an earth-stabilized reference frame centered on the vehicle (along (x) – across, right (y) – down (z)). We use a standard linear kinematic model with nearly-constant acceleration, and a standard process noise model to allow for small variations in the actual acceleration of the target. The linear kinematics are represented by the matrix,

$$\underline{F} = \begin{pmatrix} F_{3\times3} & I_{3\times3} & I_{3\times3} \\ I_{3\times3} & F_{3\times3} & I_{3\times3} \\ I_{3\times3} & I_{3\times3} & F_{3\times3} \end{pmatrix} \tag{2}$$

where $I_{3\times3}$ is the 3×3 identity matrix and,

$$F_{3\times3} = \begin{pmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$\underline{X}(t + \Delta t) = \underline{F} \cdot \underline{X}(t) + \underline{v}(t) \tag{4}$$

where $v(t)$ is an acceleration noise source (continuous Wiener process acceleration model [20]) with a process noise covariance given by,

$$\underline{Q} = \begin{bmatrix} Q_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & Q_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & Q_{3\times3} \end{bmatrix} \tag{5}$$

$$Q_{3\times3} = \begin{bmatrix} \frac{1}{20}dt^5 & \frac{1}{8}dt^4 & \frac{1}{6}dt^3 \\ \frac{1}{8}dt^4 & \frac{1}{3}dt^3 & \frac{1}{2}dt^2 \\ \frac{1}{6}dt^3 & \frac{1}{2}dt^2 & dt \end{bmatrix} \cdot S_w \tag{6}$$

where $0_{3\times3}$ is a 3×3 zero matrix, and S_w is the power spectral density of a continuous white noise process noise representing the 'jerk' (time derivative of the target acceleration).

The measurements from the three sensors are represented by periodic measurements of image position in the visible and infrared bands, giving (y, z) information, and image location and range for the LiDAR, providing (x, y, z) information. The measurement frequency and measurement errors are dictated by the corresponding fields of view of each sensor (horizontal $\theta_{FoV,y}$ and vertical $\theta_{FoV,z}$) and corresponding numbers of pixels ($N_{pix,y}$ and $N_{pix,z}$) when combined with the range to the object, and the range accuracy in the case of the LiDAR.

$$H_{vis,ir}(\underline{X}) = \begin{pmatrix} \left(\frac{N_{pix,y}}{\theta_{FoV,y}}\right)\tan^{-1}\left(\frac{y}{x}\right) + \frac{N_{pix,y}}{2} \\ \left(\frac{N_{pix,z}}{\theta_{FoV,z}}\right)\tan^{-1}\left(-\frac{z}{x}\right) + \frac{N_{pix,z}}{2} \end{pmatrix} \tag{7}$$

$$H_{LiDAR}(\underline{X}) = \begin{pmatrix} \left(\frac{N_{pix,y}}{\theta_{FoV,y}}\right)\tan^{-1}\left(\frac{y}{x}\right) + \frac{N_{pix,y}}{2} \\ \left(\frac{N_{pix,z}}{\theta_{FoV,z}}\right)\tan^{-1}\left(-\frac{z}{x}\right) + \frac{N_{pix,z}}{2} \\ \sqrt{x^2 + y^2 + z^2} \end{pmatrix} \tag{8}$$

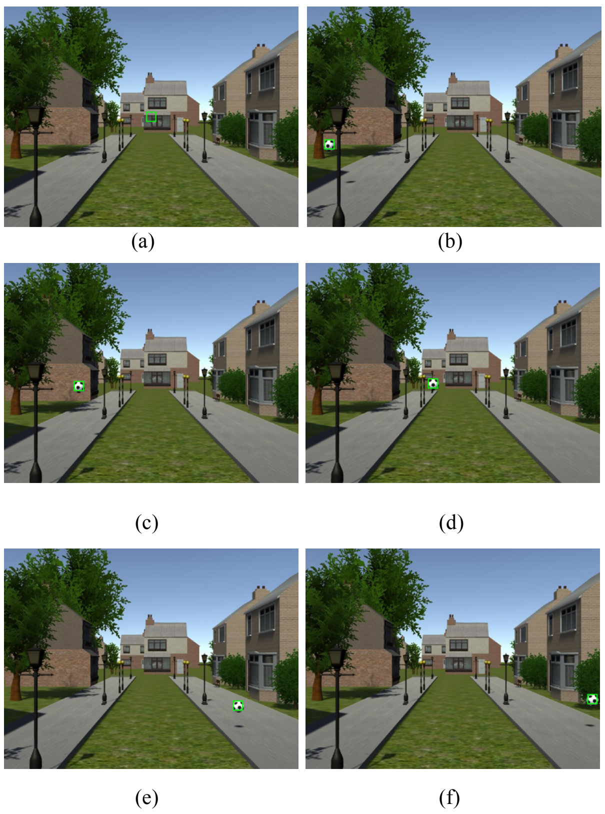


Fig. 5. Detection of the bouncing ball, the center position of the ball is captured by a green box.

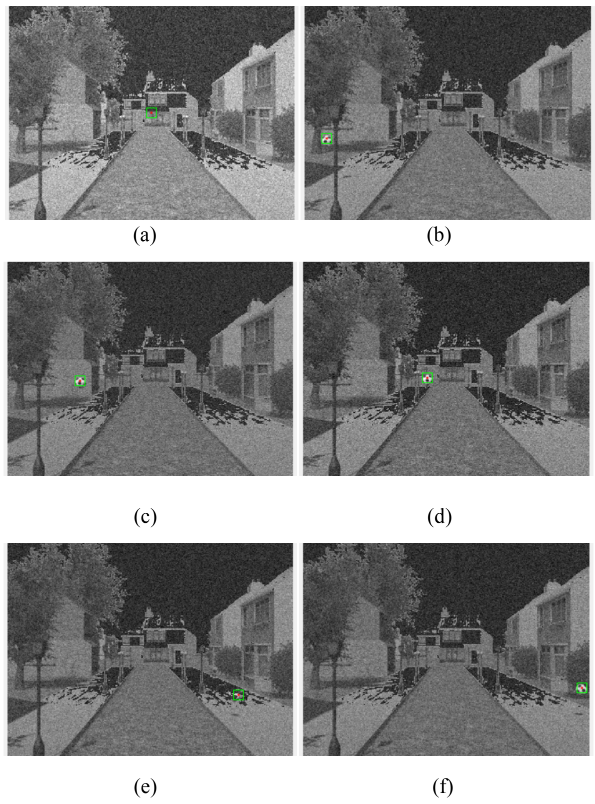


Fig. 6. Resulted IR images and detection of the bouncing ball, the centroid position of the ball is captured by a green box and the point that with the highest temperature are marked by red star.

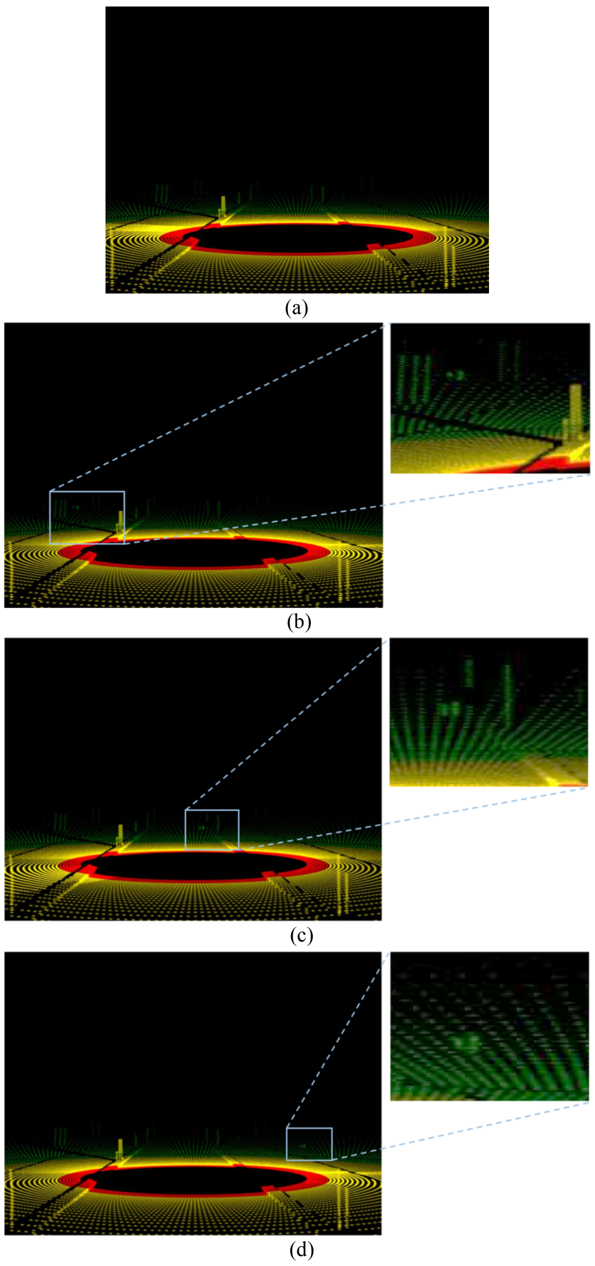


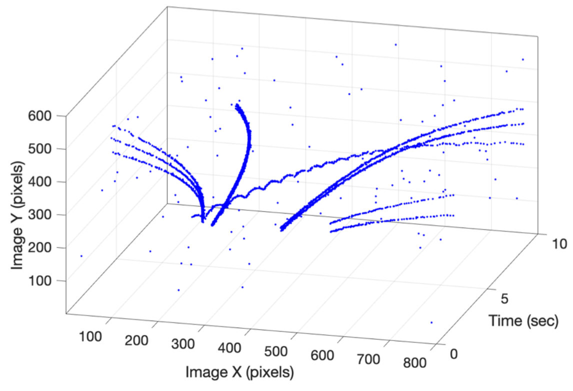
Fig. 7. LiDAR plotting of the point cloud. (a) LiDAR scanning results of the entire scene with no appearance of the ball. (b), (c) and (d) detect the ball, where the feature of the ball is shown in the enlarged window. The colour changes from red to yellow and green as the distance increases.

V. Simulation Results

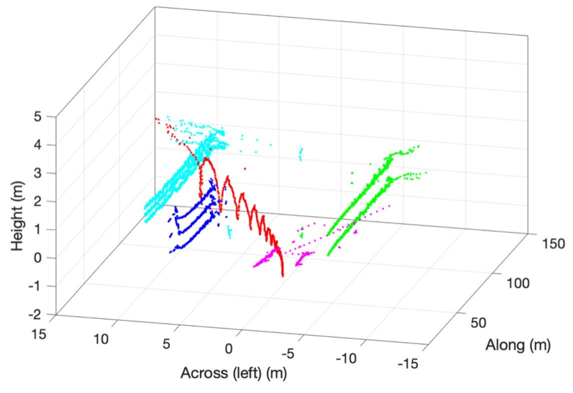
To test the proposed multi-sensor tracking system, an animation of a ball bouncing in the street is created in the scene. The true positions while the football is bouncing can be directly retrieved from the transform function of the football and sent to the tracking algorithm to calculate the trajectory. The three sensors, camera, IR camera and LiDAR, generate their measurements at their own operating frequency, namely 60 Hz, 20 Hz, and 10 Hz, respectively. All measurements received at local coordinate system are converted into the car's global coordinate system.

The image captured from the camera and the detection of the ball are represented in Fig. 5. The center position of the ball is marked using a green box. Fig. 6 shows the detection of the ball from the IR camera imaging results at corresponding time as Fig. 5. As images from IR camera have lower resolution than the camera images, the detected centers of the ball deviate from Fig. 5.

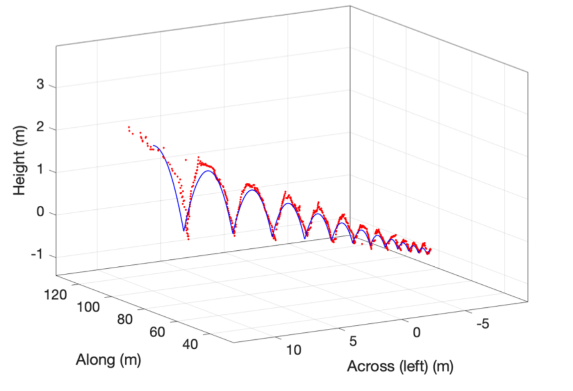
The point clouds generated from the LiDAR simulation are shown in Fig.7. Static obstacles, like street lamps are featured as line shape. While the ball is featured as a circle. Yellow points represent object near the car, while green points represent object with distance further away from the car.



(a)



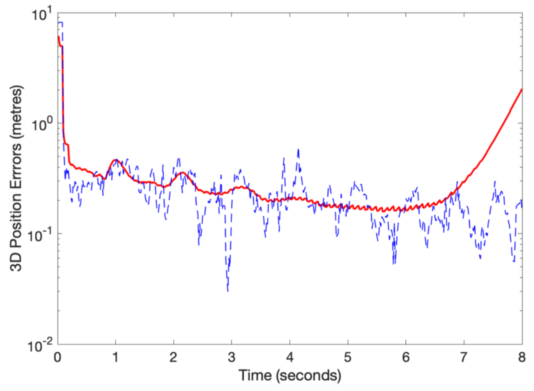
(b)



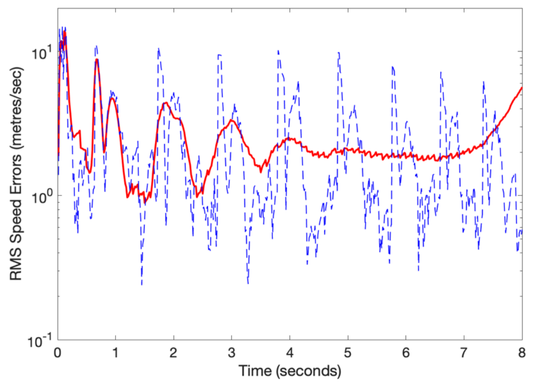
(c)

Fig. 8. Example Tracking Results, with static and random clutter: (a) 2D visible band camera detections vs time, (b) 3D tracks generated for bouncing ball (red) and other static clutter objects (other colors), (c) 3D track for bouncing ball (red dotted line) shown against true trajectory (blue solid line).

Fig. 8(a) shows example results for detections over a ten second window of the simulation for one of the imaging sensors (the 2D visible band camera). The detections include structured static clutter and some random detections that correspond to false alarms. The second figure, Fig. 8(b), shows the tracks generated by the moving object and the structured clutter, with the tracks being colored to show clusters of similar structured clutter. The moving ball is indicated in red and shown in more detail against the ground truth trajectory in Fig. 8(c). The moving object can be found quite simply from its motion against the majority of the obstacles, since their motion is correlated with the forward motion of the vehicle. By contrast, the moving ball appears as an outlier with respect to the static background, and its motion can be predicted to ascertain whether is crosses the path of the vehicle. The intention would be to generate an alert to ensure that the autonomous controller controlling the car slows down and takes action to avoid a collision, although in such cases additional higher-level processing may be required to consider other risks.



(a)



(b)

Fig. 9. Track Errors vs Time: (a) example RMS 3D position error for one realization (blue dashed line) and average RMS 3D position error (averaged over 2000 realizations, solid red line), (b) example RMS speed error for one realization (blue dashed line) and average RMS speed error (averaged over 2000 realizations, solid red line).

Averaging the track errors, the expected performance of the tracker can be predicted, as shown in Fig. 9. The root-mean-squared (RMS) 3D tracking errors are shown, as an average (in red) and for one example realization (in blue). The averages are calculated with slight variations in the initial conditions of the ball; including initial height, initial range, initial speed, etc. The example realization shows the periodic nature of the bouncing ball, which is particularly evident in the velocity errors in Fig. 9(b). The averaged data shows some residual evidence of the bouncing ball. This is due in part to the relatively small variations in the initial conditions. The other significant characteristics are the large initial errors, which are due to the low update rate of the LiDAR sensor. The obstacle's range contains a large error until one or two LiDAR measurements are available (approximately 0.1- 0.2 seconds). Towards the end of the simulation, the tracking error increases again, which is due to the obstacle being outside the field of view of one or more of the passive sensors and the location errors being dominated by the lower resolution of the LiDAR and the process noise.

VI. CONCLUSIONS

A multi-sensor tracking system has been developed as an initial realization of the ALEAD project to test the performance of the simulated sensors. In the system, two imaging sensors, camera and IR camera, and one LiDAR sensor have been modelled and a Kalman filter has been implemented to detect and track the moving obstacle (a bouncing ball). The simulation results show that by fusing the vision and LiDAR data, the tracking performance of the moving obstacle has been improved and the trajectory has been predicted successfully. As the simulations of the sensors are configured to match the specifications of real systems and are based on underlying physical models for the sensors, it opens the potential for the application of these models to test real autonomous driving systems.

In future work, the complexity of the simulated environment will be increased by including other forms of clutter (signs, mail boxes, and other road 'furniture'), people and other types of vehicle. Short range Radar sensors will be simulated and integrated into the autonomous car, together with GNSS-aided navigation systems. Duplicate sensors, such as cameras will also be added to increase the total field of view of the vision system. The aim will be to develop algorithms that are agnostic to whether the source of sensor data is a real physical system or a simulated sensor. Trade-studies will be carried out to evaluate the performance of collision detection with changes of the sensors. In addition, adverse weather conditions, such as rain and fog, will be added to improve the realism of the simulation and to challenge the sensor fusion and obstacle tracking methods developed for this program.

ACKNOWLEDGMENTS

This is a collaborative work between University of Liverpool and CGA Simulation as part of the Connected and Autonomous Vehicle programme. It is funded by the Innovate UK, with Grant No. 104274. The authors are also indebted to Mr. Dave Griffiths and Mr Ben Mitchell for their work on making the animations in the 3D scene and creating the temperature map.

REFERENCES

[1] T. Litman, "Autonomous vehicle implementation predictions", Victoria, Canada: Victoria Transport Policy Institute; 2017 Feb 27.

[2] A. Hars, "Autonomous cars: The next revolution looms", Inventivio Innovation Briefs, vol. 1, no. 4, 2010.

[3] Opgal. (2019). Could thermal cameras prevent autonomous car accidents?. [online] Available at: https://www.opgal.com/blog/thermal-cameras/could-thermal-cameras-help-prevent-the-next-fatal-autonomous-vehicle-crash/ [Accessed 25 Feb. 2019].

[4] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets", in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2174-2182. 2017.

[5] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "Moving obstacle detection in highly dynamic scenes", in 2009 IEEE International Conference on Robotics and Automation, pp. 56-63. IEEE, 2009.

[6] M. Kehrer, J. Pitz, T. Rothermel, and H.C. Reuss, "Framework for interactive testing and development of highly automated driving functions", in 18. Internationales Stuttgarter Symposium, pp. 659-669. Springer Vieweg, Wiesbaden, 2018.

[7] M. Tideman, and M. Van Noort, , 2013, June. A simulation tool suite for developing connected vehicle systems", in 2013 IEEE Intelligent Vehicles Symposium (IV), pp. 713-718. IEEE, 2013.

[8] C. Häne, L. Heng, G.H. Lee, F. Fraundorfer, P. Furgale, T. Sattler, and M. Pollefeys, "3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection", Image and Vision Computing, vol. 68, pp. 14-27, 2017.

[9] I. Shimchik, A. Sagitov, I. Afanasyev, F. Matsuno, and E. Magid, "Golf cart prototype development and navigation simulation using ROS and Gazebo", in MATEC Web of Conferences, vol. 75, pp. 09005. EDP Sciences, 2016.

[10] D. Miklić, T. Petrović, M. Čorić, Z. Pišković, and S. Bogdan, 2012, May. A modular control system for warehouse automation-algorithms and simulations in USARSim", in 2012 IEEE International Conference on Robotics and Automation, pp. 3449-3454. IEEE, 2012.

[11] A. Koubâa, Robot Operating System (ROS), Verlag: Springer, 2017.

[12] Apollo. [Online]. Available at: http://apollo.auto/. [Accessed: 23 Jan 2019].

[13] Unity3d. [Online]. Available at: https://unity3d.com/. [Accessed: 14 Mar 2019]

[14] E. J. Griffith, C. Mishra, J. F. Ralph, and S. Maskell, "A system for the generation of synthetic Wide Area Aerial surveillance imagery", Simulation Modelling Practice and Theory vol.84, pp.286–308, 2018.

[15] A.S. Ahire, "Night vision system in BMW", International Review of Applied Engineering Research, vol.4, no. 1, pp.1-10, 2014.

[16] Velodyne Lidar Inc., HDL-64E User's Manual. Velodyne Lidar Inc. 345 Digital Drive, Morgan Hill, CA95037, 2008.

[17] R. Bergelt, O. Khan, and W. Hardt, "Improving the intrinsic calibration of a velodyne lidar sensor", in 2017 IEEE SENSORS, pp. 1-3. IEEE, 2017.

[18] N. Singla, "Motion detection based on frame difference method," International Journal of Information & Computation Technology, vol. 4, no. 15, pp.1559-1565, 2014.

[19] R. Huang, H. Liang, and J. Chen, "Lidar based dynamic obstacle detection, tracking and recognition method for driverless cars", Robot vol. 38, no. 4, pp. 437-443, 2016.

[20] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan, The Extended Kalman Filter. In: Estimation with Applications to Tracking and Navigation. John Wiley & Sons, pp.381-394, 2004.

[21] N. J. Gordon, D. J. Salmond, A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation" in IEE Proceedings F (Radar and Signal Processing), vol. 140, no. 2, pp. 107-113, 1993.

[22] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking", in IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 174-188, Feb 2002.