Dynamic Object Tracking and Classification Using a Multi-Camera System

Dexter Ong, Department of Electrical and Computer Engineering, National University of Singapore

Abstract—**Automated driving systems require robust detection and tracking of objects to navigate complex urban environments. In addition to identifying pedestrians, vehicles and other dynamic objects, the position and velocity of these objects must be tracked to predict their motion. This paper presents an approach to detecting and tracking objects using optical and scene flow with fisheye cameras. The object detection model is fine-tuned for fisheye images and vehicle types unique to the local context. Depth values are obtained using plane-sweeping stereo matching. Interest points are clustered into objects and tracked across frames, and class information from the object detector is used to estimate 3D bounding boxes.**

I. Introduction

PERCEPTION is a crucial step in the automated driving system. Sensors obtain important data about objects and conditions in the environment for localisation and navigation further down the pipeline. The types of sensors commonly used include ultrasonic, RADAR, LiDAR, cameras, inertial measurement unit (IMU) and Global Navigation Satellite System (GNSS) sensors which can provide information about the physical world immediately for real-time computation [1].

The various sensors have their advantages in different conditions and applications. Vision systems with cameras have advantages of being able to capture colour in the case of RGB-D cameras, and with image enhancement can give better results in inclement weather conditions such as rain or snow. Cameras are also much cheaper than other sensors and can be easily integrated on vehicles. However, images obtained from cameras require much more processing to obtain meaningful information and are often less accurate than LiDAR.

Perception of objects on the road in an urban setting is a challenging task due to the complexity of the environment. Detecting uncommon vehicle types, pedestrians and other dynamic objects requires a robust tracking and classification system. Aside from detecting objects, it is also important to track their positions and velocities to predict their motion. This paper aims to achieve object detection and tracking for an automated driving system using fisheye cameras.

Many object detection networks are pre-trained on large, open-source datasets. To account for the use of fisheye cameras (Figure 1), and the different vehicle types in Singapore, fine-tuning the model with a local dataset of fisheye images can improve the performance of the object detector. Object tracking involves calculating the scene flow to estimate the velocity of detected objects and track their movement across frames.

Having such information about the objects in the scene is important for predicting their motion. Identifying both dynamic and stationary objects is essential for navigation. In addition, information about the class of the object can further refine the process. For instance, a vehicle is likely to stay on its current trajectory whereas the movement of a human can be more erratic and may require a greater amount of caution.

Figure 1. **Image taken with a fisheye lens.**
Where common stereo matching methods are applicable for pinhole camera images, a different approach using optical flow is used for fisheye images. This includes fine-tuning the object detector for the image distortion.

II. Literature Review

Advances in deep learning have greatly improved 2D object detection methods in camera-based systems. Tracking-by-detection methods in camera systems utilise object detectors to provide observations of tracking. In [2], with the object position and class information from the detector over several frames, trajectories of the object are hypothesised and optimised.

Deep learning has also been used for pose estimation and can work on monocular systems. 3D bounding box estimation can be achieved with viewpoint classification by training the model to detect several vertical and horizontal viewpoints [3]. Work in [4] proposes a novel disentangling transformation to obtain depth and 3D shape.

Multi-camera systems can utilise more classical methods for object tracking. Scene flow describes the dynamic 3D scene and has multiple applications in robotics. The reconstruction of the 3D scene is traditionally achieved with a sequence of stereo image pairs. Approaches to computing scene flow often involve reconstruction of 3D geometry as a single task by comparing stereo images across frames in time to obtain reliable feature correspondences [5]. However, these approaches are mainly for pinhole cameras. Fisheye cameras are used in this project and stereo matching of these images becomes more complex. Instead, estimation of optical flow

and stereo depth can be done separately before re-combining them [6]. Recent work has taken advantage of parallel processing with GPUs to reduce computation time for feature detection and scene flow estimation [7, 8]. In addition to shape and motion, work in [9] also includes colour for tracking. By matching colour information from a video camera with LiDAR cloud points, colour changes can be modelled to track points across frames.

Calculation of the dense scene flow is computationally more expensive, hence the sparse approach is preferred to achieve real-time tracking. Lenz et al. [5] implemented an approach object tracking using sparse scene flow and will be used as the main reference for this project. This sparse method detects interest points by comparing stereo images across two frames in time to obtain reliable feature correspondences.

Clustering of the interest points will allow segmentation of objects from the background. Tracked points are clustered based on scene flow differences to estimate object shape and velocity. Points can be grouped according to 3D coordinates as in [8]. However, the reconstructed scene is subject to error especially at larger distances. Another approach using the Mahalanobis distance between adjacent interest points is proposed in [5] as the threshold for scene flow difference.

A sparse scene flow approach will be adopted for this project with the addition of the object detector to obtain class information and initialise new tracks by providing bounding boxes that can be sampled for features.

III. System Overview

A. Object Detection

The Tensorflow object detection API is used for fine-tuning the model. Various models in the Tensorflow detection model zoo are evaluated. The pre-trained models are fine-tuned on the local dataset, to account for differences in vehicle types between those in the COCO dataset and those of the local context. The images in the local dataset were also taken with a fisheye lens and the fine-tuning aims to account for the image distortion as well.

The models considered from the Tensorflow detection model zoo are pre-trained on the COCO dataset. The two models of interest are the SSD and Faster R-CNN models. Since there are pre-trained weights provided, fine-tuning can be done based off those training checkpoints. Parameters for training are tweaked for a smaller learning rate and aspect ratio according to the local dataset. The fine-tuned models are then evaluated on the test dataset. The results show the Faster R-CNN having a much better detection performance at Mean AP 0.2 compared to the SSD at 0.28, though it is slower at 43ms compared to the SSD at 21ms. Since this is deemed fast enough for this application, the Faster-RCNN model is the more suitable choice.

The local dataset does not include labels for people and a portion of the COCO dataset containing labels for people are extracted for use instead. Of these images, those that containing vehicles are rejected, to prevent interference with the vehicle labels in our own dataset.

The remaining images are added to the training set. Only the labels for people are added to the tfrecords. The model is then fine-tuned with this new training set. The resultant model obtains similar scores during evaluation for the classes in the local dataset, while achieving an acceptable AP of 0.6 for detecting people on the COCO evaluation set.

B. Optical Flow

To minimise computation time, the OpenCV Computer Vision library is used to estimate the optical flow. CUDA-accelerated implementations of feature detection and optical flow calculation enable the use of GPUs the greatly speed up computation time. The CUDA-accelerated version of the OpenCV optical flow library runs much faster at 13ms per iteration compared to the non-accelerated one at 150ms.

Feature extraction is achieved with the Shi-Tomasi corner detection algorithm. From initial results, the algorithm seemed to perform well for fish-eye images and was suitable for this application. However, it was also found that for objects that are small or far away in the image, the algorithm does not detect enough feature points for reliable estimation of shape and velocity. This is partly due to the library requiring down sampled 8-bit images, resulting in loss of information and performance.

For accurate estimation of the object size and velocity, many feature points are required, evenly spread across the object. For example, an insufficient number of feature points are detected by the OpenCV library in Figure 4 and points are missing from the top and bottom edges of the car, which will affect the estimation of its size. To ensure reliable tracking for objects that have been identified by the object detector, points are sampled from within the bounding boxes and added to the detected feature points for optical flow calculation, shown in Figure 5. Points are only added if there is no existing point nearby pixel-wise, to ensure an even distribution of points. Points in the background of the object may unintentionally be included as well but will eventually be rejected in the clustering step.

Figure 2. **Feature points from algorithm (top) and manually added (bottom).** Many feature points are required, even spread across the object, for accurate shape estimation.

Figure 3. **Original image (top) next to depth map (bottom).** Depth values are invalid beyond a certain range, most noticeably for the building and road in the background of this image.

C. Scene Flow

With optical flow results, the scene is reconstructed in 3D with the PlaneSweepLib (PSL) that implements a plane sweeping stereo matching algorithm [10]. The PSL provides a depth map of fisheye images, which can then be used to obtain the scene flow from optical flow.

With the optical flow and the depth map, the scene flow can be calculated. However, the depth map is only valid up to a limited range of roughly 30 metres, beyond which the data is much more prone to error or simply invalid, as seen in Figure 3. Even within smaller distances, other factors such as reflective surfaces in windows can result in inaccurate depth estimation.

D. Clustering

Clustering involves segmenting the points obtained from the scene flow into objects. Multiple approaches were evaluated for clustering. The more conventional approach of grouping points based on their geometric 3D position was insufficient as expected based on work in [5]. While clustering using a K-Means algorithm as in [11] worked well for points close to the camera, points further away have an increasingly large error in their depth values and cannot be properly clustered.

Rather than grouping points to a centroid for each object, building a graph with Delaunay triangulation works better, since points are only compared to their immediate neighbours. This approach from [5] provided better results by also grouping points based on the Mahalanobis distance between adjacent points in the graph. The Mahalanobis distance accounts for the standard deviation of the points and hence the increasing error with distance. Most points from different objects and points in the background of the object are properly rejected as seen in Figure 8.

For the image point [u, v] mapped to the 3D point P = [x, y, z], the Jacobian of the scene flow for point P is calculated as such

$$\mathbf{J} = \begin{bmatrix} \frac{\delta P_x}{\delta u} & \frac{\delta P_x}{\delta v} & 0 \\ \frac{\delta P_y}{\delta v} & \frac{\delta P_y}{\delta v} & 0 \\ 0 & 0 & \sigma_{X_z} \end{bmatrix} \cdot \frac{1}{\triangle t}$$

With the diagonal measurement noise matrix S and covariance $\Sigma = JS^{-1}J$, the Mahalanobis distance is then given by

$$\triangle (V_i, V_j) = \sqrt{(V_i - V_j)\Sigma_{i,j}^{-1}(V_i - V_j)}$$

Figure 4. **Objects tracked across frames with trails visualised.**

E. Object Association

With points clustered into separate objects in each frame, the next step involves associating the objects across frames to create continuous tracks. The object to track association follows the Global Nearest Neighbour (GNN) approach used in [5]. The position of each object in the current frame is predicted for the next frame. Objects in that next frame which are within a distance from the predicted position of each track are associated to that track and the track is updated with a Kalman filter.

Unassociated objects in each frame will initiate new tracks. Tracks are considered valid after at least two consecutive frames of association and are removed after two consecutive frames without association.

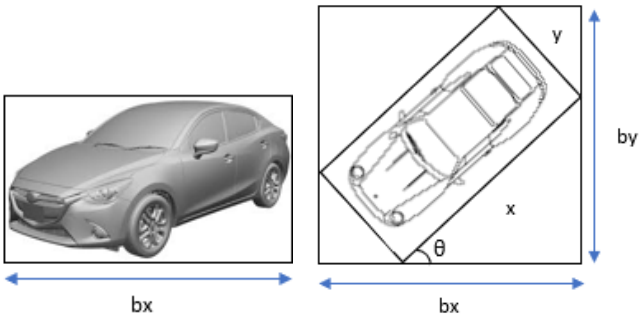


Figure 5. **Bounding box estimation by dimension ratio.**

Pre-trained Model	
Label	**AP Score**
Car	0.434453
Truck	0.202238
Bus	0.166504
Motorcycle	0.160256
Bicycle	0.060000
Mean AP	0.171575

Fine-tuned Model	
Label	**AP Score**
Car	0.609483
Truck	0.371108
Bus	0.488729
Motorcycle	0.352094
Bicycle	0.163558
Mean AP	0.283568

Table 1. **Results of pre-trained model and fine-tuned model.**

F. Bounding Box Estimation

The heading and dimensions of each object can be estimated and visualised with a 3D bounding box.

An initial approach involved the use of L-shape fitting [12]. The proposed method iterates through various rectangle sizes and angles and calculates the distance of each point to the nearest edge. The lowest cost configuration is found by a search-based algorithm and can estimate both the heading and the bounding box size of the object. However, results were not satisfactory with our data. Due to the high error in depth values at larger distances, outliers significantly affected the estimation of both the bounding box and the heading.

Since heading cannot not be reliably estimated by L-shape fitting, velocity is used instead. The bounding box of each object is then determined using an estimated ratio *x:y* of the dimensions of the object based on its class provided by the object detector. Though the object detector does not provide depth information, the bounding box from the object detection can be used to estimate the dimensions *x* and *y* of the object given its heading.

Edge *bx* can be obtained from either the 2D bounding box or the unprojected points. Edge *by* would ideally be given by the depth values. However, assuming that the depth values are not reliable, with the dimension ratio of *x:y* and the heading θ, both *x* and *y* can be solved for (Figure 5).

This method assumes that the entire object is detected. In cases of occlusions or where part of the object is out of the frame, the bounding box will not fit well. When the velocity and hence the heading of the tracked object is inaccurate, the bounding box estimation will be affected as well.

Estimating the bounding box of the vehicle using the velocity also only holds true when the vehicle has significant velocity. For objects below a velocity threshold where the heading can no longer be determined reliably, L-shape fitting is used instead.

IV. RESULTS

The object detector is shown to have significant improvement in detection scores after fine-tuning with a local dataset. The fine-tuning addresses issues in both local vehicle types and image distortion due to fisheye lenses.

The fine-tuned model that was trained on the local dataset is compared against the Faster R-CNN model provided by Tensorflow that was pre-trained on the COCO dataset. From the results in Table 1, it is evident that the fine-tuning provided a significant improvement in performance. There are improvements in the detection score across all classes with significantly higher scores in the classes of vehicles unique to the local context like buses.

Tracks are initialised by the object detector using the bounding box information. Objects are tracked across frames with estimated velocity and 3D bounding boxes. At the intersection in Figure 7(a), L-shape fitting is used for the stationary car in the foreground, while dimension ratio estimation is used for the moving vehicles in the background.

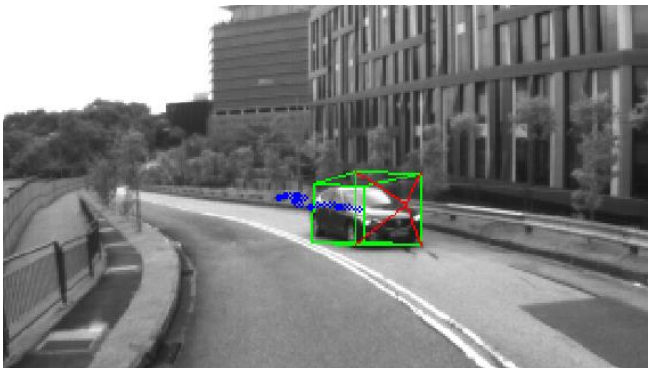


Figure 6. **Bounding box estimated using the dimension ratio of the object class removes the need for accurate depth values.**

For objects near the range limit of the depth map, tracking of object position and heading is more erratic. Other issues include insufficient points spanning the entire object or failing to reject some points in the background of the object, resulting in inaccurate shape estimation. This can be attributed to the feature extraction and optical flow calculation. Due to the lack of ground truth for object positions and bounding boxes, quantitative evaluation is difficult. However, several limitations in the approach can be identified based on the results.

Due to the insufficient feature detection, the interest points used for optical flow are not optimal and can result in inaccurate scene flow. For example, manually added points may lie on reflective surfaces or in shadows that contain inaccurate depth values. Manually adding interest points also means that the system is dependent on the object detector to initialise new object tracks. Objects that are false negatives and unique object shapes and classes which are not detected by the object detector will not be tracked.

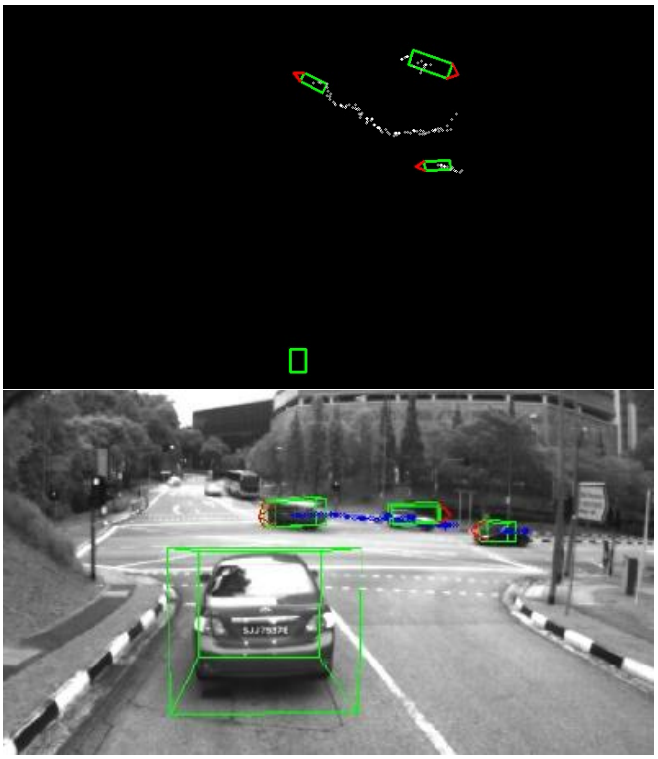
As mentioned previously, the depth map has limitations in range. While enough to provide estimation of velocity, the depth values cannot be used for bounding box estimation. There is also significant fluctuation in the velocity at larger distances.

V. Further Work

Nvidia's optical flow SDK for Turing GPUs can achieve dense optical flow at a higher bit depth. Better optical flow calculation can be expected without significant computational cost. With a robust feature extraction method, objects can be tracked without prior information from the object detector. This will enable tracking of all dynamic objects in the scene.

With the limitations of the depth map, object shapes cannot be accurately estimated. While the object detector is currently used apart from object tracking for bounding box detections, texture plane measurements and semantic shape priors can also be considered to better approximate object shapes [13].

Deep learning-based approaches can also be explored for object tracking. While such methods are limited by the GPUs on the vehicle, limitations in certain areas of the system like the bounding box estimation can be overcome with 2D detection, such as viewpoint classification in [3] for pose and bounding box estimation.



(a)

(b)

(c)

Figure 7. **Results of object detection, tracking and bounding box estimation on fisheye images.** (a) Tracking at intersection visualised with top-down view. (b) Two vehicles in opposite lane. (c) Car moving past intersection.

VI. Conclusion

Object detection and tracking methods are explored for application with fisheye images. The object detector has better detection scores after fine-tuning with fisheye images from a local dataset. Where methods used in pinhole cameras for feature extraction and depth estimation are insufficient, optical flow combined with a depth map generated by a plane sweep algorithm is proven to be a feasible approach for computing the scene flow for fisheye images.

Objects are tracked successfully across frames with velocity and bounding box estimation, though limitations in the current approach are identified in the feature extraction and depth map. Further work in feature extraction can remove dependence on the object detector for initialising new tracks and improve the robustness of the object tracking to different object types.

Acknowledgements

The author thanks Dr. Lionel Heng and Professor Robby T. Tan for their invaluable mentorship and guidance. The author also expresses appreciation to the NUS Department Electrical and Computer Engineering for accommodating the schedule for this project.

References

[1] Francisca Rosique, Pedro J. Navarro, Carlos Fernández and Antonio Padilla. A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research. Perception Sensors for Road Applications (2019).

[2] Andreas Ess, Konrad Schindler, Bastian Leibe, Luc Van Gool. Object detection and tracking for autonomous navigation in dynamic environments. The International Journal of Robotics Research (2010).

[3] Peiliang Li, Tong Qin, Shaojie Shen. Stereo Vision-based Semantic 3D Object and Ego-motion Tracking for Autonomous Driving. European Conference on Computer Vision (2018).

[4] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel Lopez-Antequera, Peter Kontschieder. Disentangling Monocular 3D Object Detection (2019).

[5] P. Lenz, J. Ziegler, A. Geiger, M. Roser. Sparse scene flow segmentation for moving object detection in urban environments. Intelligent Vehicles Symposium (IV) (2011).

[6] R. Schuster, O. Wasenmuller, D. Stricker. Dense Scene Flow from Stereo Disparity and Optical Flow. DFKI – German Research Center for Artificial Intelligence (2018).

[7] S.A. Mahmoudi, M.Kierzynka, P. Manneback. Real-Time GPU-Based Motion Detection and Tracking Using Full HD Videos. 5th International ICST Conference (2013).

[8] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, D. Cremers. A Primal-Dual Framework for Real-Time Dense RGB-D Scene Flow. IEEE International Conference on Robotics and Automation (2015).

[9] D. Held, J. Levinson, S. Thrun, and S. Savarese. Robust Real-Time Tracking Combining 3D Shape, Color, and Motion. International Journal of Robotics Research (2016).

[10] Christian Häne, Lionel Heng, Gim Hee Lee, Alexey Sizov, Marc Pollefeys. Real-Time Direct Dense Matching on Fisheye Images Using Plane-Sweeping Stereo. Proc Int. Conf. on 3D Vison (3DV) (2014).

[11] M. Jaimez, M. Souiai, J. Stuckler, J. Gonzalez-Jimenez, D. Cremers. Motion Cooperation: Smooth Piece-Wise Rigid Scene Flow from RGB-D Images. 3D Vision (2015).

[12] Xiao Zhang, Wenda Xu, Chiyu Dong and John M. Dolan. Efficient L-Shape Fitting for Vehicle Detection Using Laser Scanners. IEEE Intelligent Vehicles Symposium (2017).

[13] Kyel Ok, Katherine Liu, Kris Frey, Jonathan P. How, Nicholas Roy. Robust Object-based SLAM for High-speed Autonomous Navigation. IEEE International Conference on Robotics and Automation (2019).