Using Object Classification to Improve the Accuracy of SLAM

Jason Formosa
Institute of Information & Communication Technology
University College
MCAST, Paola PLA 9032
{jason.formosa.a100452}@mcast.edu.mt

***Abstract*—SLAM (Simultaneous Localization and Mapping) provides a solution for an autonomous driving vehicle to start from an unknown position and gradually build a map of the environment and track its path through it. Autonomous vehicles are required to make decisions that are safe as possible however this is largely dependant on the accuracy of the whole system. The aim of this project is to increase the accuracy of the tracking system by ignoring non stationary objects in the scene. The proposed solution will use neural networks to detect other cars in the scene and ignore them in the tracking stage. The general outcome was overall positive; the system obtained an in increase in accuracy when other vehicles were obstructing the majority of the camera's vision.**

***Index Terms*—Autonomous Vehicles, SLAM, Neural Networks**

I. Introduction

Autonomous driving cars have gained a lot of popularity in the past few years. Major companies like Google, Uber and Zoox are all competing to be first company to release a fully autonomous car to the general public since that would guarantee them a lead in the market. The autonomous driving process can be summarized down to 4 steps; Perception, how the vehicle sees the environment; Localization, the process by which which the vehicle locates itself in the environment; Path Planning, the planning logic taking into consideration the final destination; Control, where the computers interact with the vehicle steering, accelerator and brakes in order to make it follow the determined path. The accuracy in the localization step is crucial since the following steps all depend on it to make valid and safe decisions. The mentioned companies all have the funds necessary to buy expensive hardware in order to make the system as accurate as possible. The companies all use LIDAR sensors to perceive the environment, which are unparalleled when it comes to accuracy [1], however cost thousands of dollars to purchase. This discourages independent researchers from contributing their work and therefore accelerating the development of autonomous vehicles. It also prohibits hobbyists from trying the tech out and helping to test emerging methodologies. The aim of this project was to find out a cost effective alternative to expensive sensors by using common RGB cameras and come up with an algorithm to improve their accuracy. A scenario which often produces errors in the tracking is when another vehicle is driving in front of the camera or when vehicles were driving across the camera's peripheral vision. Given this analysis, a pre-trained CNN (Convolutional Neural Network) was used in order to detect vehicles and mask them out from the tracking algorithm. The accuracy of the CNN had to be analysed as well to make sure the detected objects were valid. Considering a second system was added, the computation required and the speed of the whole system had to be analysed as well.

II. Literature Review

A. History

Recently there has been a lot of research done on mapping and localization due to the rise in popularity of self driving cars, however these algorithms date back to the 1990s [2]. SLAM is defined as a "long-term globally referenced position estimation without a priori information" [2]. It is a complex problem to solve because in order for a robot to precisely map an environment, it must know precisely where it is located in the environment, but in order to accomplish this, it must have an accurate map of the environment. This poses a question of "which came first, the chicken or the egg?". As depicted by John J. & Hugh F., this was solved by learning features of the environment from sensor data at the initial position and subsequently tracking these features at every timestep. This is currently adopted as the basis of solving the mapping and localization problem, with different variations of the algorithm adapted to use different sensors to map the environment.

There's a great variety of sensors that can be used in the tracking part of the SLAM algorithm. [3] These generally fall in these categories:

1) **Dead-reckoning**
   - **Angular Rate** (e.g. gyros)
   - **Linear Acceleration** (e.g. accelerometers)
   - **Relative Displacement** (e.g. wheel encoder, optical flow)
2) **Map-point-referenced**
   - **Range** (e.g. ultrasonic, Lidar, radar, raw GPS)
   - **1-D Bearing** (laser scan, linear CCD, phased array)
   - **2-D Bearing** (imaging sensor, quadcell, pan/tilt servo)
   - **Range rate** (Doppler radar, phase-coherent acoustic)
   - **Range difference** (e.g. TDOA acoustic or radar)

- **Dipole field component** (e.g. active source magnetic tracker, electric field tracker)

3) **Earth-referenced**
   - **Cartesian position** (e.g. cooked GPS, altimeter)
   - **Heading** (e.g. magnetic compass, gyrocompass)
   - **Speed** (optical flow, airspeed, waterspeed sensors)
   - **Direction** (e.g. optical flow)
   - **Homogeneous field** (magnetometers, gravitometers)

Each of these sensors has a different strong point between accuracy, weight and price. In the Neato Vacuum Cleaner [4], a 1-D LIDAR scanner was used in order for the vacuum to be able to better map the room and plan its path more efficiently. This type of sensor was used because of its low cost even though it has poor accuracy, however in the case of a vacuum the accuracy is not crucial. In a research by MD Robotics [5], a sensor fusion system comprised of a stereo vision camera system, LIDAR sensor and wheel sensors was used. The vision system was used to build accurate 3D models while the LIDAR was used to infrequently improve the accuracy of the visual and wheel odometry.

High accuracy sensors are critical in autonomous driving vehicles in order to guarantee safe decisions. This is why most companies working on autonomous driving vehicles have implemented LIDAR sensors on their testing vehicles. Tesla has decided to stick with using cameras for their cars due to their low cost for the accuracy obtained from them. Stereo camera systems are only slightly worse than LIDAR systems [6] when it comes to the accuracy of close range objects. The farther away the object is, the lower the accuracy. This is due to far away away object being smaller in terms of pixels and therefore having less information to compute their location in the stereo pairs accurately; a small error in pixel location results in a large error in depth.

B. PTAM

[7] The major difference between PTAM and SLAM is that in PTAM the mapping and tracking are separated and run in parallel. This allows further improvement by basing the mapping on keyframes. Subsequent frames often consist of redundant data; making the SLAM algorithm less efficient by processing these frames. Keyframes can be set to whenever a significant change in the scene occurs and map the scene at that keyframe only, therefore freeing up the CPU to do the tracking. The SLAM algorithm also suffers from data association errors. This is due to the incremental nature of the algorithm; if an error was made in an early iteration, it can carry over the whole sequence and therefore even corrupt a map.Due to tracking and mapping being run separate, data association between them need not even be made, freeing up resources used by inlier/outlier detection in SLAM.

C. ORB-SLAM

[8] ORB-SLAM is very similar to PTAM in the way that the tracking and mapping run in parallel along with an added thread for loop closing. Loop closing consists is the process of detection if the current location has been previously mapped and therefore fix any deviation that might have occurred in the current odometry. ORB-SLAM also uses an ORB feature detector and descriptor instead of image patches and these features are used for all of the tasks which makes it more efficient. The ORB feature detector can run in real-time on the CPU as opposed to requiring expensive GPUs for better performance. Another improvement is the ”survival of the fittest” strategy in regards to keyframe selection. In PTAM keyframes are selected cautiously in order to avoid excessive growth in computation complexity, in ORB-SLAM, keyframes are generated more freely but are kept only if they satisfy very strict conditions. The results in less outliers than in PTAM at the expense of less points overall in the map.

D. S-PTAM

[9] S-PTAM is a variation of PTAM in that it uses a stereo camera system instead of a mono system. It retains the parallel nature in order to achieve real-time performance. The use of stereo cameras allows the real life scale of the environment to be computed without any prior information and also provides additional depth information so that stereo constraint can be placed on the pose and map-refinement algorithms to improve robustness.

III. Research Methodology

In order to test the accuracy of the proposed method, a python implementation of the S-PTAM paper was used. This uses frames from 2 cameras positioned side by side and outputs an Nx12 matrix where N is the number of frames and each row is a transformation matrix that projects the current pose in the coordinate system of the initial pose. This means that each row is the current position of the cameras in relation to the first location. This system works by tracking the change in location of visual features in each frame from previous frames. Regions that are known to contain erroneous points, like moving vehicles in the case of this project, can therefore be masked out and not included in the calculations. In order to detect and localize the cars in the scene, a PyTorch implementation of the YOLOv3 paper was used. This outputs a text file with a line per frame. In each line the bounding box coordinates, class accuracy, object accuracy and class are stored, each object separated by a '|' character. These bounding boxes are then rendered on binary image and passed to the feature detector of the S-PTAM system as a mask. The data for this project was obtained from the odometry dataset from the The KITTI Vision Benchmark Suite. This dataset provides image data, both in colour and in black & white, for 21 sequences of car driving along certain roads with different conditions in each sequence, including driving along an open stretch of road, driving in residential roads without any other vehicles, and driving in busy city centres. The path taken by the vehicle was also provided as ground truth data for sequences 00..10 which are the sequences that were used in this project in order to compare the deviation of the path obtained by masking the vehicles with the ground truth data from the dataset. The ground truth data is also provided in the form of an Nx12 matrix, which will make it easier to compare the two paths. The comparisons will be done with evo, a ”Python package for the evaluation of odometry and SLAM” [10]. The ’evo_traj’ script will be used to plot the two paths in the same XZ space in order to get a visual for how they differ. The ’evo_ape’ script will be used to get the absolute deviation from the ground truth in metres along with a plot of time vs error. The ’evo_rpe’ script will be used to get the relative deviation from the ground truth path and a plot of time vs error as well. The relative error will help identify the frames in which the error was greater and therefore identify the types of scenes our algorithm will help improve the accuracy of and where the algorithm may fail. The object recognition system was not implemented directly into the tracking system; it was run independently for each sequence before the tracking was executed. The tracking system was modified to accept the object detection text files as the mask input. In order to reduce the computation time, the YOLOv3 network was loaded on an NVIDIA GTX1060 GPU. This dropped the computation time from 1.6s when computing on an i7-3700 CPU to 80ms per frame on the GPU.

IV. Project Evaluation

Seq. No.	Min(m) - w/o mask w/ mask	Avg(m) - w/o mask w/ mask	Max(m) - w/o mask w/ mask
00	1.71 1.67	6.78 5.69	12.98 12.48
01	20.73 30.38	471.22 450.20	757.11 740.96
02	3.03 5.69	10.53 11.66	28.64 30.39
03	0.09 0.11	1.08 0.60	2.27 1.10
04	0.13 0.06	0.53 0.62	1.14 1.21
05	0.98 1.10	3.98 4.54	9.12 11.09
06	1.24 1.50	3.11 3.47	7.53 9.11
07	4.60 0.39	9.49 1.29	20.15 3.32
08	1.90 2.10	4.42 4.55	10.38 11.09
09	1.26 1.26	5.64 5.75	17.72 16.25
10	0.10 0.03	1.04 0.93	2.11 1.94

TABLE I
Table showing absolute deviation from ground truth for sequences 00..10 in the KITTI Dataset

In scenes that include a number of parked cars, the proposed implementation with masking performed slightly worse than without the masking. This is as expected, the cars aren’t getting filtered by whether they are stationary or not, and therefore useful data from parked cars is being removed from the computation.

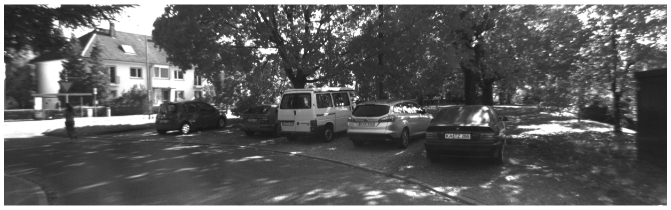
Fig. 1. Sequence 00 Frame 4368

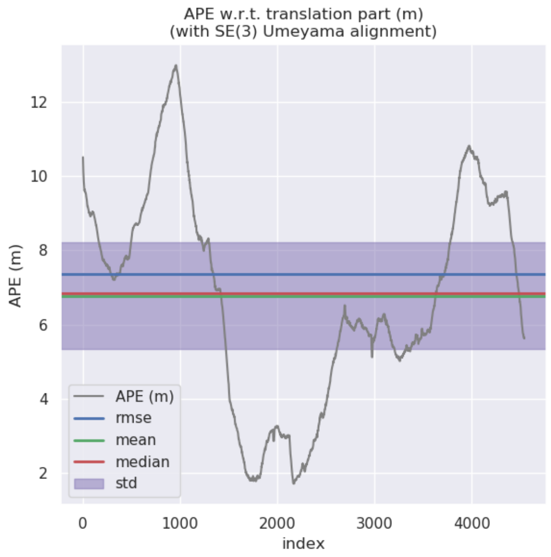

Fig. 2. Sequence 00 Absolute Error

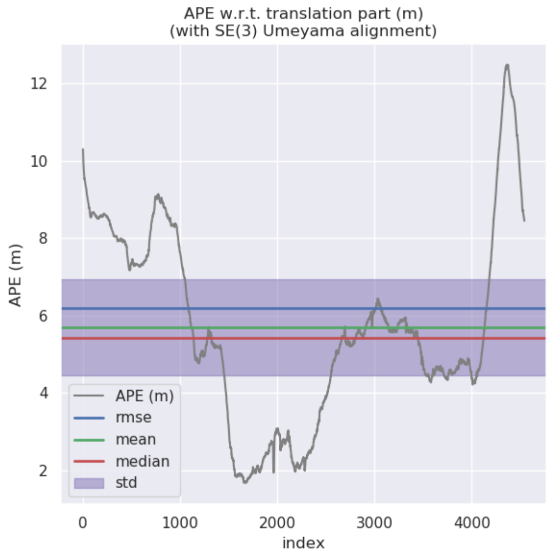

Fig. 3. Sequence 00 Masked Absolute Error

In 1, the deviation spiked in the masked implementation as can be seen in Fig. 3 at frame 4368. This is due to the vehicle being in a turn (loss of details because of motion blur) and parked cars being in front of the camera which doesn't occur in the non-masked version (Fig.2).

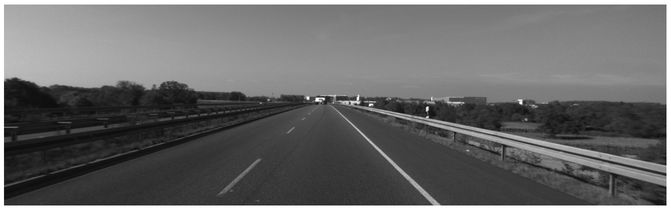
Fig. 4. Sequence 01 Long stretch of road

The large deviation in sequence 01 as seen in Table. I is due to an open stretch of road on highway with no individually distinguishable features in close proximity to the vehicle (Fig. 4); the image feature detection system failed on both versions.

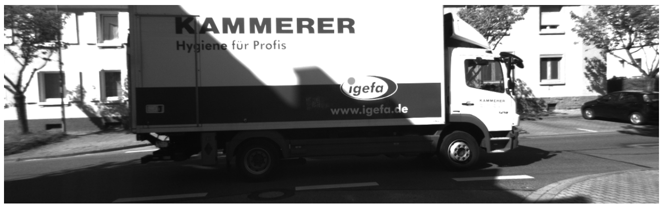

Fig. 5. Sequence 07 Cars in camera's peripheral vision

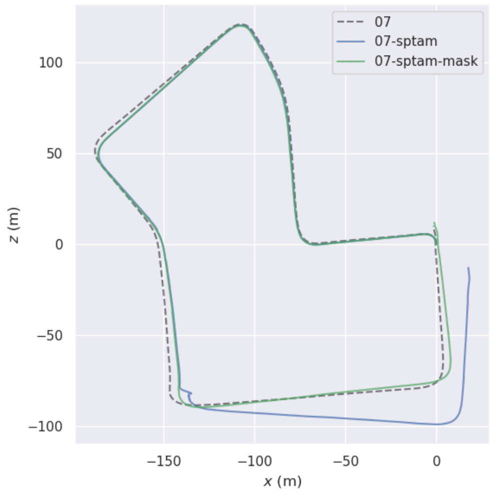

Fig. 6. Sequence 07 Trajectories

In sequence 7 the version with the mask really came out ahead. As can be seen in Fig. 5, the car was stationary with other cars crossing in front of the majority of the camera's field of view. In the non masked version, these cars were considered as valid point to use for locating the car in the scene which corrupted the path at that point in time (as can be seen in Fig. 6) whilst in the masked version these points were ignored and

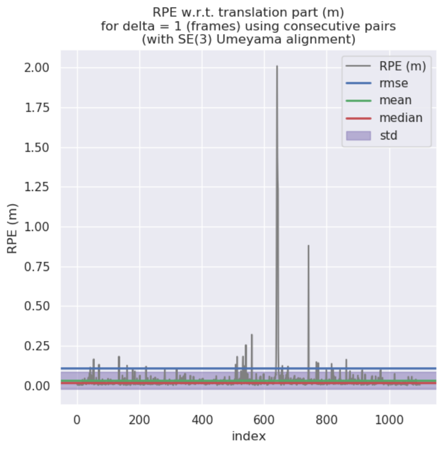

Fig. 7. Sequence 07 Relative Error

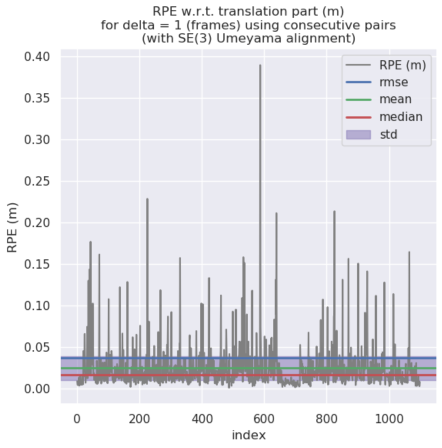

Fig. 8. Sequence 07 Masked Relative Error

the correct path was retained. This can be clearly seen in Fig. 7 & Fig. 8 where the spike at frame 638 is eliminated in the masked version.

V. Conclusion

A solution for improving accuracy in vehicle localization on public roads was presented. This method uses Neural Networks to detect other vehicles on the road and removes any features that were placed on them from the feature detector of the localization step.

In order to improve the performance of the system, the Neural Network was loaded on a GPU and a parallel implementation of SLAM called S-PTAM was used which uses a stereo camera system for the localization and mapping.

The accuracy of this method was tested on the sequences from the odometry dataset from The KITTI Vision Benchmark Suite with which a ground truth pose file was provided. The effectiveness of the system with masking was compared to the system without masking in these sequences to determine where the strengths and drawbacks of the system are. Results indicate that the proposed system didn't improve the accuracy by much when the scene doesn't include any cars and even decreased it when numerous parked cars are present, however the system showed a substantial improvement in scenes where non stationary vehicles occupy most of the camera's field of view.

A an improvement for this system would be the ability to identify stationary vs non stationary objects in the scene with an acceptable accuracy even while the vehicle is in motion. This can be done through scene flow analysis which determines where each pixel is moving in 3D space. This is also what I plan to achieve in my future work.

Appendix A
Supporting Material

Seq. No.	**Min(m) - w/o mask w/ mask**	**Avg(m) - w/o mask w/ mask**	**Max(m) - w/o mask w/ mask**
00	0.0014	0.0298	0.3476
	0.0011	0.0319	0.4067
01	0.0119	1.3587	4.8784
	0.0100	1.3260	3.200
02	0.0007	0.0345	0.3924
	0.0007	0.0353	0.4541
03	0.0016	0.0293	0.5264
	0.0022	0.0282	0.4458
04	0.0021	0.0381	0.2047
	0.0044	0.0382	0.2367
05	0.0007	0.0247	0.4383
	0.0007	0.0260	0.3842
06	0.0011	0.0328	0.2302
	0.0007	0.0344	0.2994
07	0.0006	0.0326	2.0099
	0.0006	0.0243	0.3893
08	0.0009	0.0350	0.3417
	0.0011	0.0362	0.4531
09	0.0015	0.0358	0.6641
	0.0014	0.0374	0.5125
10	0.0015	0.0280	0.2603
	0.0023	0.0274	0.2421

TABLE II
TABLE SHOWING RELATIVE DEVIATION FROM GROUND TRUTH FOR SEQUENCES 00..10 IN THE KITTI DATASET

Fig. 9. MCAST Logo

References

[1] J. Castellanos, J. Montiel, J. Neira, and J. Tardos, "The spmap: a probabilistic framework for simultaneous localization and map building," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, p. 948952, 1999.

[2] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, Nov 1991, pp. 1442–1447 vol.3.

[3] E. M. Foxlin, "Generalized architecture for simultaneous localization, auto-calibration, and map-building," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, Sep. 2002, pp. 527–533 vol.1.

[4] E. Ackerman, "Neato adds persistent, actionable maps to new d7 robot vacuum," Aug 2017. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/home-robots/neato-adds-persistent-actionable-maps-to-new-d7-robot-vacuum

[5] S. Se, H.-K. Ng, P. Jasiobedzki, and T.-J. Moyung, "Vision based modeling and localization for planetary exploration rovers," *55th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, 2004.

[6] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," 2018.

[7] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, p. 11471163, 2015.

[9] T. Pire, T. Fischer, J. Civera, P. D. Cristoforis, and J. J. Berlles, "Stereo parallel tracking and mapping for robot localization," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[10] M. Grupp, "evo: Python package for the evaluation of odometry and slam." 2017. [Online]. Available: https://github.com/MichaelGrupp/evo/