2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Improved localization using visual features and maps for Autonomous Cars

Sathya Narayanan Kasturi Rangan*, Veera Ganesh Yalla** (*Member IEEE)*, Davide Bacchet and Izzy Domi

Abstract— In this paper we present our research work on self-localization of the vehicle (ego-state) aided by visual features and maps. This approach requires only last received GPS location, Odometry estimates as the vehicle moves, a database of visual features around the last received GPS estimate and Standard Definition (SD) Map. Towards this goal, we extract Oriented and Rotated Brief (ORB) Descriptors from the Images collected during a drive and use Bag of Words (BoW) approach for creating a vocabulary of visual words. We also use Inverted File Index method for fast querying of the image descriptor that is seen currently by the ego vehicle against a database of all the descriptors collected for finding the possible locations of the robot. The locations for the corresponding matches are then used to update the measurement model. This approach has helped us to localize within 3 seconds with average position and orientation error of 0.8 m and 0.38 degree respectively for all the sequences.

I. Introduction

Localization is one of the paramount requirements for autonomous vehicles. Performing path planning and control on an autonomous vehicle to go from one location to another without having the global position and orientation of the vehicle is almost impossible. Most of the vehicles today have Real-Time Kinematic Differential Global Positioning Systems (RTK-DGPS) for getting the location of the vehicle [1]. However, these sensors face signal reception problems especially when driving through urban cities with high-rise buildings, under tunnels etc. Also, usage of these sensors adds a lot of monetary costs to the vehicle. The main objective of our approach is to perform localization specifically in these circumstances.

Map-based precise localization [2] is an alternative solution to overcome the problems faced by the RTK-GPS. The maps used for precise localization are built using one or more cameras and LIDARs. In this paper, we propose another alternative solution using the road standard definition (SD) maps, which has information's like road width, curvature, number of lanes along the road networks in the world. We use traditional Monte-Carlo Localization algorithm [3] to localize the vehicle along the road networks available in the map. Our approach assumes that the vehicle travels only along the road networks. We also augment the available road networks in the standard definition maps with Oriented-Rotational BRIEF descriptors [4] which are extracted using the camera images collected during the mapping procedure.

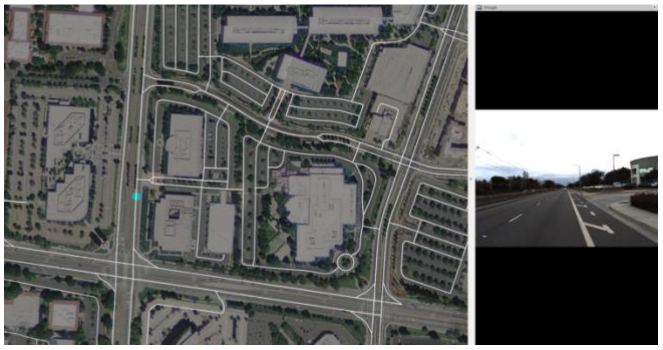
Fig.1 Our approach for finding the global position and orientation the vehicle (drawn as aqua colored dots) in a road segment map (drawn as white lines).

Finally, we use Hierarchical Bag of Words (BoW) [5] and Inverted File Indexing method [6] for finding the closest matching visual descriptor from a database of descriptors that are stored in the map and use its position and orientation information weakly, to find the global position and orientation of the vehicle more accurately and at much faster rate.

II. RELATED WORK

Localization has been studied in robotics for a long time now. Many researchers have tried to get the relative motion estimates also known as odometry using camera images [7, 8, 9], LIDAR point clouds [10, 11], as well as fusing both with IMUs [12, 13]. As these methods are incremental, they tend to often drift as the vehicle travels for a long a long distance. Simultaneous Localization Mapping [14, 15] solves the problem of drift by joint optimization of the position, orientation and map landmarks. However, the maps that are built using these methods can be used for localization only if the vehicle has the sensor setup that was used for mapping previously. Floros et. al [16] used Open Street Maps (OSM) [17] to localize the robot. The authors collect a history of odometry poses of the robot and did chamfer matching against the OSM road networks. This method assumes that history of odometry poses would resemble the shape of the road network

Sathya Narayanan Kasturi Rangan* is a MS student from Worcester Polytechnic Institute who is doing his internship at NIO (email: skasturirangan@wpi.edu)

Veera Ganesh Yalla**, is the corresponding author currently with NIO and leads the Autonomous Driving, Perception team. 3200 North 1st Street, San Jose, CA 95134 USA (e-mail: ganesh.yalla@nio.io)

Davide Bacchet is with NIO and leads the Autonomous Driving, Simulation and Mapping team. 3200 North 1st Street, San Jose, CA 95134, USA (e-mail: davide.bacchet@nio.io)

Izzy Domi is with NIO and works on simulation and mapping technologies in the the Autonomous Driving, Simulation and Mapping team. 3200 North 1st Street, San Jose, CA 95134, USA (e-mail: izzy.domi@nio.io)

segments in OSM. However, the noisy measurements in the odometry would cause the matching to fail. Also, if there are similar road segments, there could be multiple possible locations.

In the recent past, Brubaker et al [18] used the publicly available OSM to localize the vehicle. They used graph-based mixture of gaussians approach along with probabilistic transition model and visual odometry. However, their method is very slow to converge to the actual pose of the vehicle. This is due to the fact there is no measurement model available other than the odometry information. Specifically, if the map has very similar segments, the localization may take very long time to converge and sometimes not converge at all.

Ma et al. [19] tried to overcome this problem using semantic information in the images such as direction of the sun, presence of an intersection, road type, speed limit etc. Ruchiti et. al [20] tried to localize on OSM using classified laser scans coming out of a mobile robot mounted with a LIDAR. They used particle filter-based Monte-Carlo localization approach for localizing the mobile robot. Even though they explore the semantics in the LIDAR scans i.e. classifying if a point belongs to road or non-road, their method might take a long time like [18], if there are road segments that are very similar in shape. To replace the requirements of GPS sensors, researchers have also tried instance place-recognition techniques [21, 22, 23] wherein a database of geo-tagged images or LIDAR point clouds are stored and the most relevant information to current scene that the vehicle sees is retrieved.

We approach the localization problem using particle filters along with ORB descriptors which allow us to localize the vehicle laterally as well as longitudinally in a short period of time. Instead of storing the entire image or point cloud, we augment the road segment map with ORB descriptors and we use them weakly to weight the particles in the map. We leverage the advantages of both Brubaker et al [18] method and instance place recognition method to provide a more robust and better solution to the problem of localization.

III. Localization on Road Standard definition Maps

The objective of our approach is to find the position and orientation of the vehicle using particle filters. The inputs that are available to the algorithm are a standard definition (SD) map augmented with visual feature descriptors, odometry measurements, last received GPS estimate and the current image of the world as seen by the vehicle though the cameras. In the following section, we describe about the SD maps, particle filter algorithm and how Hierarchical Bag of Words approach is used for performing fast and robust localization.

A. SD Maps and Preprocessing

SD map represents the road networks in the world. They typically contain information like road width, curvature, number of lanes in the road segment etc. Figure 2 shows a standard definition map drawn as white lines overlaid on top of a satellite map.

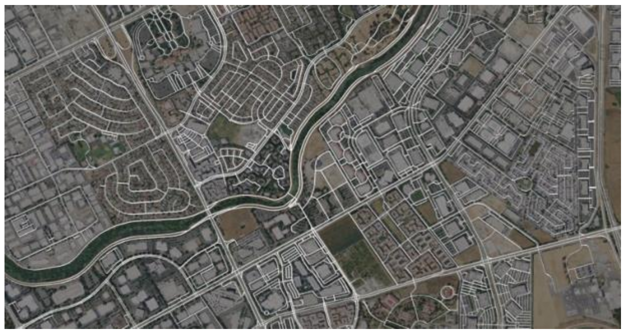

Fig. 2 Road standard definition map (drawn as white lines) overlaid on top of a satellite map. Road segment map (drawn as white lines).

The map itself is a set of linearly spaced points. For our use case, we interpolate between each set of points to create closely spaced points such that we find a segment map point in every half-a-meter. This is done for finding distance between a particle and its closest map point in an efficient way using KD-Trees [24].

B. Monte-Carlo Localization

To solve the problem of localization, we use particle filters. The output of the particle filter algorithm is essentially a probability distribution over the pose x_t of the vehicle. Typically, the particle filter algorithm has a motion update step, weight update step and resampling step. The algorithm has a set of particles P. Each particle has the states $\{x_i, y_i, \theta_i, w_i, d_i\}$ where x_i, y_i represents the position of the i^{th} particle and θ_i represents the orientation of the i^{th} particle. w_i and d_i represents the weight assigned and distance from the closest map point of the i^{th} particle.

C. Motion Update for the Particles

Each of the particle by themselves represent a possible location for the vehicle. When the vehicle moves from one position to another position in the real world, we need to move each of the particles from its current position and orientation similar to the actual movement of the vehicle. The motion update step is generally represented by $p(X_t/U_t, X_{t-1}, M)$. Here X_t and X_{t-1} represent the pose of the vehicle at discrete sample time t and $t-1$, U_t represents the control input given to the vehicle and M represents the road standard definition map. Let $x_t,\ y_t,\ \theta_t$ be the pose of the vehicle at discrete sample time t and $x_{t-1}, y_{t-1}, \theta_{t-1}$ be pose of the pose of the vehicle at discrete sample time $t-1$. The following equations are used to update the pose of the particle as the vehicle moves in the world:

$$\delta_{trans} = \sqrt{(x_t - x_{t-1}) + (y_t - y_{t-1})} \quad (1)$$

$$\delta_{rot1} = atan2\left((y_t - y_{t-1}), (x_t - x_{t-1})\right) \quad (2)$$

$$\delta_{rot2} = \theta_t - \theta_{t-1} - \delta_{rot1} \quad (3)$$

$$x_t = x_{t-1} + \cos(\theta_{t-1} + \delta_{rot1}) + \mathcal{N}(0, \sigma_x^2) \quad (4)$$

$$y_t = y_{t-1} + \sin(\theta_{t-1} + \delta_{rot1}) + \mathcal{N}(0, \sigma_y^2) \quad (5)$$

$$\theta_t = \theta_{t-1} + \delta_{rot1} + \mathcal{N}(0, \sigma_\theta^2) \quad (6)$$

Here $\mathcal{N}(0,\sigma_x^2), \mathcal{N}(0,\sigma_y^2), \mathcal{N}(0,\sigma_\theta^2)$ represents the gaussian noise added to each state of the particle with mean zero and standard deviation of $\sigma_x, \sigma_y, \sigma_\theta$ correspondingly. To get the position and orientation of the vehicle at the next time step, we currently use a visual odometry system similar to [9] fused with IMU. This fusion gives us the relative transformation from discrete sample time $t-1$ to t.

D. Weight Update based on distance to the map

As the pose of the particles are propagated similar to the vehicle's motion, we need to assign weight to each of the particles. Each particle is assigned a weight $w_i = p(Z_t/X_t, M)$ where w_i is the weight of i^{th} particle and Z_t represents the observations received at discrete sample time t. For weighting the particles, we use the distance of the particle from the closest segment point. Assuming that vehicle moves only along the road network segments, we find the closest map point to the current particle and assign a weight to it based on the following equation:

$$w_i = \frac{1}{\sqrt{2\pi\sigma_d^2}} e^{-d_i^2/2\sigma_d^2} \quad (7)$$

Here, σ_d is the standard deviation distance metric and d_i is distance of the particle from closest map point. This intuitively means that, as a particle moves away from the segment map, the distance of the particle to closest map point is going to increase because of which it will be assigned a lower weight. Only the particles, which travel along the segments will be assigned with higher weights.

E. Building a vocabulary of the Image Descriptors

To accelerate the convergence of the particle filter to the true pose, we extract descriptors from images that are collected during the mapping procedure. Due to its well-known performance in [15], we extract ORB-descriptors.

Once we move from the image space to the descriptor space, we create a vocabulary of visual words that appears in all the images that are collected. For this we perform a k-means clustering algorithm. Here, k is a user specified parameter which indicates the number of visual words that are needed to be created. The main purpose of creating a set of vocabulary from images, is that it helps us to represent any image using s set of words, where $s \in k$ reduces the complexity and memory footprints. Figure 3 shows the transformation from image space to descriptor space along with clustering procedure for creating a vocabulary of k visual words.

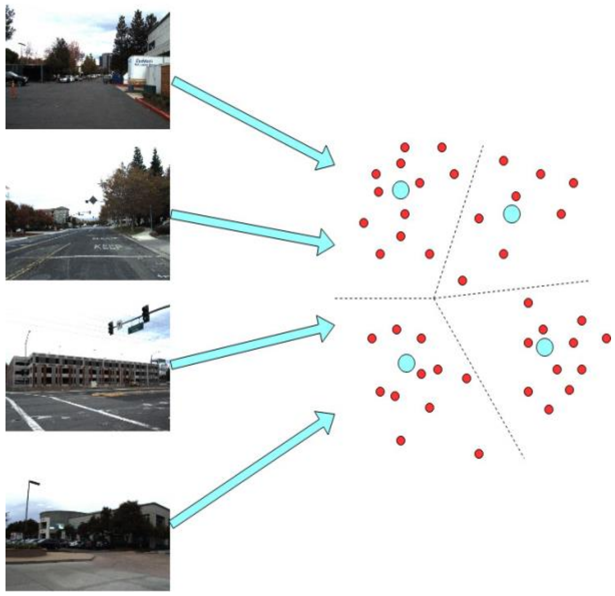

Fig. 3 Building vocabulary by going from image space to descriptor space and the performing a k-means clustering to find the k-visual words.

F. Inverted File Indexing for the Database of Visual Words

After creating the vocabulary, we create a database having visual words extracted from images collected during the mapping procedure and their corresponding position and orientation i.e., we augment the road standard definition map. Once we are on road, to localize the vehicle, we use the Inverted File Indexing [6] method for fast retrieval of pose corresponding to the Image that the vehicle sees. We generate a set of visual words from the image seen by the car and retrieve a set of o images using the $L2$-norm distance as a comparison metric.

This method accelerates the retrieval of closest matching position and orientation of the vehicle. If not, we must match the current view with each information in the database. The inverted file indexing method can be thought of as an index page in the back of a book. Instead of a word pointing to a page number, we point a set of visual words to corresponding images.

G. Weight Update Based on Closest Matching Image Descriptor

With the closest matching position and orientation in hand, we find all the particles which are within certain region from that position. For all these particles, we add the following weight to it:

$$w_i = \begin{cases} w_i + \frac{1}{\sqrt{2\pi\sigma_l^2}} e^{-l_i^2/2\sigma_l^2}, if\ l_i < dthresh \\ 0,\ otherwise \end{cases} \quad (8)$$

Here, l_i indicates the distance between the i^{th} particle and the retrieved image pose from the database, σ_l is standard deviation, *dthresh* is the distance threshold.

Sequence No.	Localization without Visual Features			Localization with Visual Features		
	Time Taken (in seconds)	Position Error (m)	Orientation Error (in degree)	Time Taken (in seconds)	Position Error (in meters)	Orientation Error (in degree)
1.	61	7.37	1.2542	3	0.84073	0.54295
2.	Failed to Localize	Not Available	Not Available	2.7	0.59064	0.021786
3.	43	2.2	1.56	2.5	0.566	0.472
4.	36	5.788563	1.96912	2.7	1.1653	0.696
5.	Failed to Localize	Not Available	Not Available	3	0.86522	0.186895

Table 1 – It shows the time taken for the method in [18] and our algorithm to localize the vehicle. Also, we recorded the position and orientation error for both the methods. Our method has an average position error of 0.8m and orientation error of 0.38 degree.

In this way, we are increasing the weights for all the particles that are close to the matched position and orientation. Thus, we will be able to converge to a true pose of the vehicle in a very short period of time.

H. Stochastic Universal Resampling

The final step in a particle filter is the resampling step. By resampling, we are essentially eliminating all the bad particles. Only the good particles can propagate through the environment. We use stochastic universal resampling method to resample all the particles and it has a complexity of $O(\log\ n)$.

IV. RESULTS

In this section, we will look at the experiments carried out and the corresponding results obtained. To evaluate our method which uses visual features along with particle filter, we compare our results against the method where we don't use visual features. The second method just uses the distance to the segment map point for updating the weights of the particles and is comparable to Brubaker et al [18].

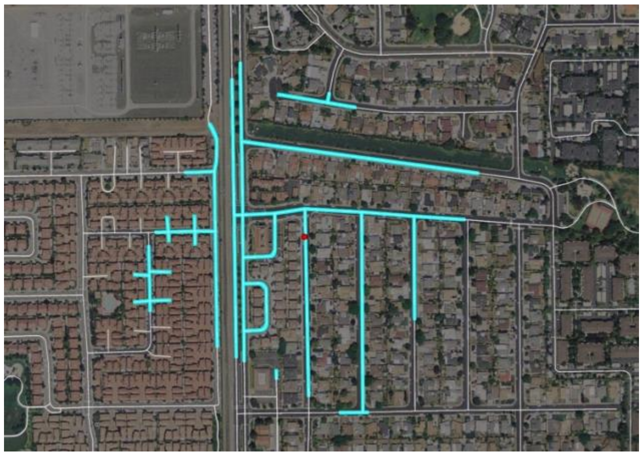

Fig. 4 Initialization of particles (aqua color) along the road segment map with last received GPS estimate (red dot in the center) as the pivot point

A. Building Vocabulary of Visual Words and Augmentation of Segment Map

For building the vocabulary of visual words and augmenting the segment map, we first need to collect image data and their corresponding poses. For collecting images, we use Allied Vision Technologies Mako GigE cameras mounted on top of a car. For the image poses, we use Novatel GPS fused with IMU. We use DBoW3 [6] library for building a vocabulary of visual words. The branching factor $k = 6$ and depth level $L = 5$ was used for creating the vocabulary. Finally, we convert all the ORB descriptors extracted from the collected images to visual words using the built vocabulary. Thus, we have a database of visual-words that will occur at different positions in the segment map.

B. Initialization of Particle Filter

As part of the algorithm, we take the last received GPS estimate just to restrict the search space. Otherwise, we do not depend on the GPS during the runtime of the algorithm. With this GPS location as a pivot point, we initialize particles around a $200\text{m} \times 200\text{m}$ region which represents the possible locations where the vehicle could be. As the value for the initialization region grows, the number of particles needed to localize the vehicle increases. Hence, we chose this value based on experimentation. We can restrict the initialization region value to a much less value than 200m, keeping into account the precision of the GPS installed in the vehicle. This would help running the algorithm at a much faster rate. Fig. 4 shows the particles initialized along the road segment map.

C. Motion Update and Weight Update for Particle Filters

After initialization, the next step is to give a motion update to the particles. We get the relative transformation between two successive time steps using camera images and IMU as mentioned earlier. We use $\sigma_x = 0.03$, $\sigma_y = 0.03$ and $\sigma_\theta = 0.001$ as the standard deviation for motion update step. These values are proportional to the rate of spread of the particles and are obtained by experimentation. Then based on the position of each of the particle and its closest road segment map point we assign a weight to the particle. The $\sigma_d = 10$ was used for weight update step based on experimentation.

Fig. 5 shows results obtained for the method where the weight update is based on the closest segment map point alone. We test the algorithm against a set of sequences which are collected by us. The red trajectory in the image is the actual trajectory taken by the vehicle and it is given by the RTK-DGPS. The algorithm converges for most cases. However, it fails at few cases. On analyzing the failed cases,

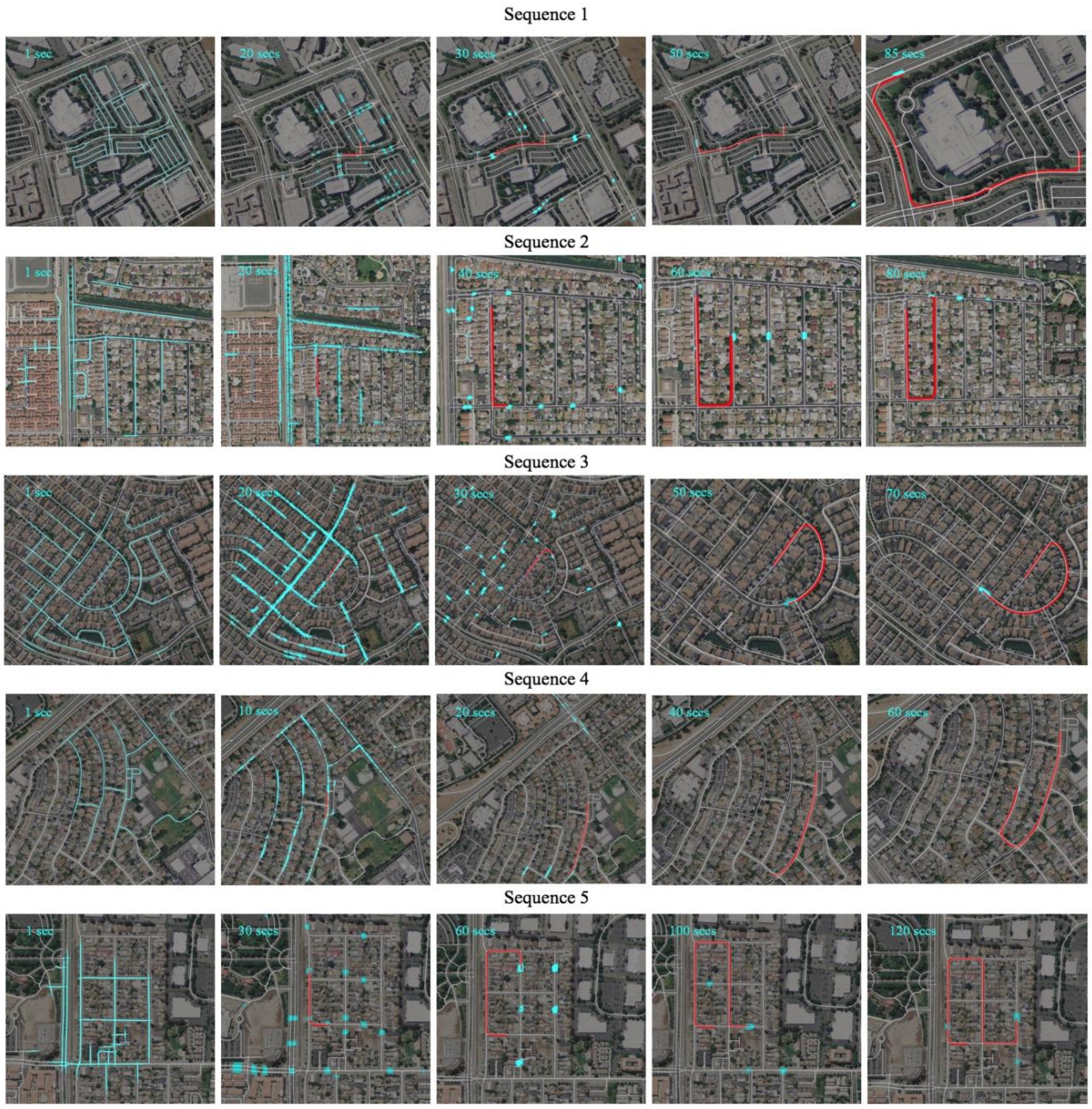

Fig. 5 Self-localization similar to [18], where the weight update for the particles are based on the distance to the segment map. The algorithm is tested against a set of sequences as shown. The number on the top right corner indicates the time at which the image was taken. The particles are represented in aqua color and the red line indicates the true trajectory taken by the vehicle given by the RTK-DGPS. The vehicle localized itself in sequence 1, 3, 4. However, due to similar road structure in sequence 2 and 5 the vehicle was not able to localize.

we see that the sequences have certain ambiguities in it. The road shape is symmetric and hence, algorithm could not converge to true position and orientation of the vehicle. Also, one other major drawback is that the algorithm takes a long time to converge to the true pose of the vehicle. In case of safety critical systems such as self-driving cars, this time taken for localization is not very optimal. To solve this problem, in addition to the above weighting we also add weights to the particles based on the visual features to achieve faster localization. We use the database and vocabulary that was built offline to perform this fast and robust localization. The $\sigma_l = 10$ was used for weight update using the visual features based on experimentation.

Fig. 6 shows the results obtained by using visual features for localization. Our approach is a combination of particle filters and place recognition technique to provide a reliable

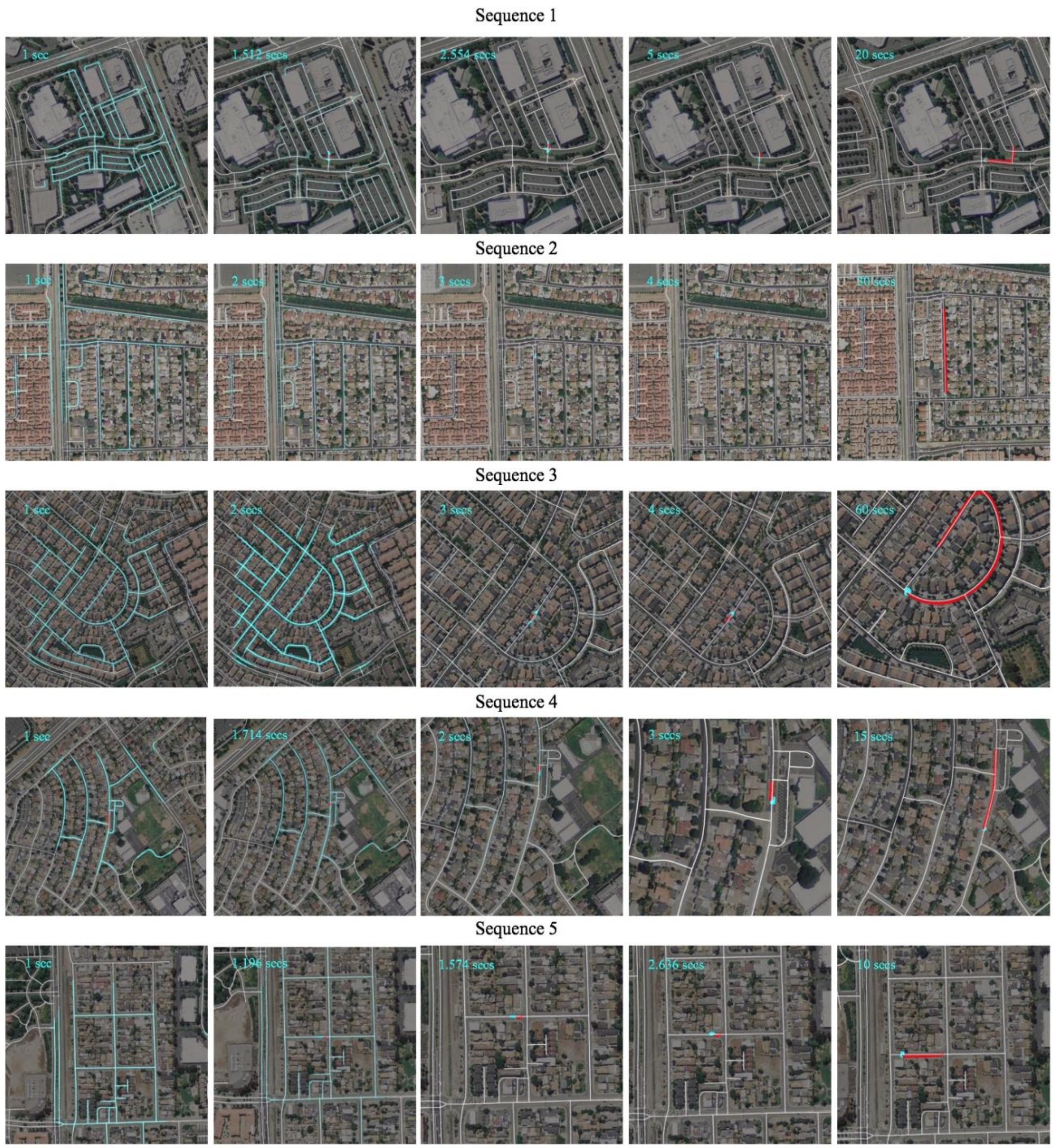

Fig. 6 Self-Localization aided by visual features for faster localization. The aqua colored dots represent the position of the particles and red trajectory indicates the true trajectory taken by the vehicle as given by RTK-DGPS. The time at which the image was taken is printed on the top left corner of each image.

location of the vehicle in a short period of time. As can be seen from the Fig. 9, we were able to localize the vehicle even in difficult sequences like sequence 2 and 3 where there are lot of similar road structures. Table 1 shows the time taken to localize for a sequence along with the position and orientation errors for both the methods. The algorithm runs in real-time providing correct localization estimates. The algorithm is tested on a system with Intel Core i7-6820HQ CPU @ 2.70 GHz x 8 using C++ and ROS. The algorithm can be accelerated further with GPU implementation of feature extraction, matching and particle filter.

V. Conclusions

In this paper, we presented a localization approach using a combination of particle filters and Bag of Visual Words (BoW). The inputs for the algorithm were camera images and road standard definition maps. We tested our algorithm on five different sequence collected locally using our vehicle platform mounted with sensors. We can localize the vehicle in less than 3 seconds with average position and orientation error of 0.8m and 0.38 degree. We obtained 15x faster results with lower position and orientation error than [18]. We are currently investigating deep learning-based feature descriptor extraction methods along with other information in the maps such as number of lanes, lane markers, road width traffic signs, points of interest (POI) information to significantly reduce the position and orientation errors. We also plan accelerate the implementation of particle filters with parallel computing for increased performance.

Acknowledgments

We would like to thank NIO for sponsoring this R&D work.

References

[1] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., ... & Gittleman, M. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, *25*(8), 425-466.

[2] Levinson, Jesse, Michael Montemerlo, and Sebastian Thrun. "Map-Based Precision Vehicle Localization in Urban Environments." *Robotics: Science and Systems*. Vol. 4. 2007.

[3] Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte carlo localization for mobile robots. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* (Vol. 2, pp. 1322-1328). IEEE.

[4] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on* (pp. 2564-2571). IEEE.

[5] Grana, C., Borghesani, D., Manfredi, M., & Cucchiara, R. (2013, March). A fast approach for integrating ORB descriptors in the bag of words model. In *Proc. SPIE* (Vol. 8667, pp. 866709-866709).

[6] Gálvez-López, D., & Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, *28*(5), 1188-1197.

[7] Nistér, D., Naroditsky, O., & Bergen, J. (2004, June). Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (Vol. 1, pp. I-I). IEEE.

[8] Scaramuzza, D., & Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE robotics & automation magazine*, *18*(4), 80-92.

[9] Zhang, J., Kaess, M., & Singh, S. (2014, September). Real-time depth enhanced monocular odometry. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on* (pp. 4973-4980). IEEE.

[10] Zhang, J., & Singh, S. (2014, July). LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems*(Vol. 2).

[11] Zhang, J., & Singh, S. (2015, May). Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*(pp. 2174-2181). IEEE.

[12] Kneip, L., Chli, M., & Siegwart, R. Y. (2011). Robust real-time visual odometry with a single camera and an imu. In *Proceedings of the British Machine Vision Conference 2011*. British Machine Vision Association.

[13] Konolige, K., Agrawal, M., & Sola, J. (2010). Large-scale visual odometry for rough terrain. In *Robotics research* (pp. 201-212). Springer, Berlin, Heidelberg.

[14] Engel, J., Schöps, T., & Cremers, D. (2014, September). LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision* (pp. 834-849). Springer, Cham.

[15] Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, *31*(5), 1147-1163.

[16] Floros, G., van der Zander, B., & Leibe, B. (2013, May). Openstreetslam: Global vehicle localization using openstreetmaps. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (pp. 1054-1059). IEEE.

[17] Haklay, M., & Weber, P. (2008). Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, *7*(4), 12-18.

[18] Brubaker, Marcus A., Andreas Geiger, and Raquel Urtasun. "Lost! leveraging the crowd for probabilistic visual self-localization." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013.

[19] Ma, W. C., Wang, S., Brubaker, M. A., Fidler, S., & Urtasun, R. (2017, May). Find your way by observing the sun and other semantic cues. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on* (pp. 6292-6299). IEEE.

[20] Ruchti, Philipp, et al. "Localization on openstreetmap data using a 3d laser scanner." Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015.

[21] Badino, H., Huber, D., & Kanade, T. (2012, May). Real-time topometric localization. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on* (pp. 1635-1642). IEEE.

[22] Wu, J., & Rehg, J. M. (2008, June). Where am I: Place instance and category recognition using spatial PACT. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1-8). IEEE.

[23] Hays, J., & Efros, A. A. (2008, June). IM2GPS: estimating geographic information from a single image. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1-8). IEEE.

[24] Muja, M., & Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, *2*(331-340), 2.