Abstract—This paper presents a pose estimation method based on 1-Point RANSAC EKF (Extended Kalman Filter) framework. The method fuses the depth data from a LIDAR and the visual data from a monocular camera to estimate the pose of an Unmanned Ground Vehicle (UGV) in a GPS denied environment. Its estimation framework continually updates the vehicle’s 6D pose state and temporary estimates of the extracted visual features’ 3D positions. In contrast to the conventional EKF-SLAM (Simultaneous Localization And Mapping) frameworks, the proposed method discards feature estimates from the extended state vector once they are no longer observed for several steps. As a result, the extended state vector always maintains a reasonable size that is suitable for online calculation. The fusion of laser and visual data is performed both in the feature initialization part of the EKF-SLAM process and in the motion prediction stage. A RANSAC pose calculation procedure is devised to produce pose estimate for the motion model. The proposed method has been successfully tested on the Ford campus’s LIDAR-Vision dataset. The results are compared with the ground truth and the estimation error is ~1.9% of the path length.

Index Terms—SLAM, laser and vision fusion, Extended Kalman Filter, pose estimation.

1. INTRODUCTION

RECENTLY self-driving car technology has attracted much research effort from the robotics and computer vision community. One of the technical challenges that have to be overcome in developing a self-driving car is vehicle pose estimation. For navigation in a 3D space, a vehicle pose consists of the vehicle’s 3D coordinate and orientation, which is of 6 Degree-Of-Freedom (DOF). Most of the existing solutions to the pose estimation problem employ multiple sensors because navigation in urban environments may involve complicated scenarios that cannot be handled by using just one sensor modality. These scenarios include reliable obstacle detection and avoidance in the presence of dynamic objects (other cars and pedestrians) with relatively high vehicle speed (up to 7 times of the speed of a regular mobile robot platform). Due to the multiple-sensor navigational approach, previous work in pose estimation uses data from various sensors such as GPS, 3D LIDAR, camera and IMU [1]–[4]. Among these sensors, GPS may fail to provide reliable data in metropolitan areas where skyscrapers may obstruct the satellite signals. Consumer grade IMUs are not accurate for long-duration pose estimation while military grade IMUs are prohibitively expensive. As camera and 3D LIDAR are common perception sensors of an autonomous car, it is natural to dual use these sensors for pose estimation.

In the literature, 3D LIDAR or camera has been used for pose estimation of mobile robots. The advantage of using a 3D LIDAR is that it provides accurate and long range depth data which is not susceptible to sunlight and shadow condition when compared to camera. However, registering 3D laser scan (the same process of finding pose change) incurs high computational cost and is not possible when the scans are geometrically featureless. Iterative Closest Point (ICP) and its variants [5] are representative scan registration methods. They are subject to local minimum problem and require initial alignment of scans (i.e., two scans need to be close enough for the methods to succeed). In contrast, a camera may provide high resolution data with distinctive visual features that can be used to obtain depth information (e.g., stereo vision, structure from motion). However, the depth data are only accurate for short range. The depth accuracy drops quadratically with true distance. The representative visual feature based pose estimation method is the visual odometry algorithm [6]–[8] which usually uses a stereo vision system for perception. Due to the characteristics of depth measurement, the visual odometry algorithm only works well for short range scene. Also, it has been demonstrated in [9] that moving shadow cast by the robot body may cause failure to the visual odometry. To overcome the disadvantages of using a single sensor modality, visual and laser data are fused for pose estimation in the existing research.

ICP, visual odometry and their variants only provide pose shift measurement in two consecutive data frames. The pose change information needs to be integrated to obtain vehicle pose over time during entire navigational course. Dead-reckoning approach leads to unbounded cumulative pose error and is thus not suitable for long range navigation. In the literature, state estimation and filtering techniques have been employed to correct the cumulative error. Davison et al. [10] proposed a Monocular camera Simultaneous Localization and Mapping (a.k.a MonoSLAM) method. MonoSLAM uses Extended Kalman Filter (EKF) as the filtering framework that simultaneously estimates the robot pose and the 3D coordinates of some static points around the robot using data from a monocular camera. MonoSLAM and its variants suffer from the scale ambiguity problem i.e., the scale of the map and the robot trajectory cannot be determined. Moosmann et al. [11] developed a SLAM algorithm based on Velodyne 3D LIDAR. In their method the point cloud from the LIDAR at each step is converted into an implicit graph where each node contains a 3D data point and its local normal vector and each...
edge is the distance between two nodes. The whole graph is then uniformly sub-sampled in order to get a smaller set of local surfaces and their respective normals. The data of the resulted graph is then processed by an ICP based method to obtain the best overlap between the global map and current point cloud. They also devised some procedures to update the map. Although the results are accurate, the process is time consuming most of the time and the result may be affected by the local minima problem.

Researchers in [12] and [2] have attempted to fuse 3D LIDAR and visual data for pose estimation. In the SLAM method proposed by Newman et al. [12] the state vector consists of past vehicle poses. State updates are made by a scan matching procedure and visual data is simultaneously used for loop detection and closure based on a similarity measure. The scan matching process is computationally costly. Pandey et al. [2] employed a RANSAC [13] procedure to provide the initial guess of pose change for performing Generalized-ICP (GICP) [5] to align the laser scan data. The GICP result is then used to construct a pose graph whose nodes and edges represent robot poses and pose changes, respectively. The pose graph formulates the pose estimation problem as an optimization problem that is then solved by the incremental Smoothing and Mapping(iLSAM) method [18]. Again, the GICP scan matching is time consuming.

To alleviate the computational cost of pose estimation, we propose an EKF-SLAM algorithm in this work. The proposed method uses visual features and their corresponding 3D laser data to estimate vehicle pose change and track the extended state vector of the EKF. This visual and laser data fusion approach removes the need of processing an entire laser scan and is thus computationally efficient. The proposed method extends the visual SLAM algorithm in [14] to the case where part of the image contains depth information (from 3D laser data). The original method introduced in [14] is a visual SLAM process [10] that maintains a temporary map of the most recent observed features and performs a modified EKF-SLAM based on this map. We refer to the method as 1-Point RANSAC EKF-SLAM or 1PRE method from now on. In each step of 1PRE, a hypothesis pool is generated for motion prediction, from which, the most plausible hypothesis (one with the highest number of inliers) is chosen by RANSAC to partially update the state estimate. Then the inlier set is recalculated based on the updated state and consequently the full state update is performed using the recalculated inlier set. The 1PRE method is suitable for urban navigation applications because it can partially cope with moving objects and dynamic environments provided that they are not the dominant part of the scene [14].

Our method differs from the work in [14] in that the fusion of 3D laser and visual data is performed both in feature initialization and motion prediction stages. This data fusion process solves the above-mentioned scale problem and increases the convergence rate and accuracy of the estimates. As a result, our method extends the applicability of the 1PRE method to urban navigation scenarios where a vehicle usually has relatively higher speed and uses distant features for pose estimation. Our data fusion scheme is also different from [12] which performs full laser scan matching and incremental 3D point-cloud map-building. Our method only keeps a temporary feature map rather than a full 3D point-cloud map and thus removes the need of laser scan registration. It is noted that the loop closing method in [12] can be integrated into our method to further improve pose estimation accuracy without sacrificing too much computational efficiency. Finally, our method has been validated with a real urban scenario [15] where the vehicle speed is several times higher than both of the aforementioned works.

2. Methodology

The proposed method is iterative and each iteration consists of five stages as shown in Algorithm (1). In the first stage, map management and pose change calculation are performed. The method maintains a map that contains a set of visual features. As vehicle moves, new features are added to and obsolete features are discarded from the map. In this stage the pose shift between steps $k - 1$ and $k$ is also calculated by a RANSAC procedure. In the second stage, the state vector and the observations are predicted and a correlation based method is used to match features. In the third stage, a RANSAC process draws individual samples from the matched features and forms a hypothesis for each sample by updating the state vector based on that sample. Support of the hypothesis is then calculated as the number of the features whose innovation values are smaller than a threshold $\tau$ under that hypothesis. Most supported hypothesis is found at the end. In the forth stage, inliers of the most supported hypothesis are used to update the state. In the fifth stage, outliers of the most supported hypothesis are examined to find possible additional inliers, based on which the state is further updated. The input to the algorithm is state estimate $\hat{x}_{k-1|k-1}$, its covariance $P_{k-1|k-1}$, threshold $\tau$ for inlier determination and the associated laser and image data for steps $k$ and $k - 1$. In this work, laser data points are projected onto the camera’s image plane for laser and visual data association (see Fig. 1). Each projected laser data point is associated with the closest image pixel. This process produces depth data $d = (d_x, d_y, d_z)$ in the camera’s coordinate for each associated image pixel. Features are extracted only from the Relevant Image Portion (RIP) that has depth data. This treatment saves a great deal of computational time in visual feature extraction. In the remainder of this section each stage of the algorithm is detailed.

2.1. Probabilistic Framework

At step $k$, the state vector $x_k$ to be estimated is the concatenation of the robot state, $x_k^r$, and the stacked vector of features in the temporary map, $\text{Map}_k = [f_1^k, f_2^k, ..., f_m^k]$, where $f_j^k$ is the $j^{th}$ feature in the map. The covariance matrix, $P_k$, for the vector $x_k$ is also updated in each step and contains the correlation information between entries of the state vector. The robot state, $x_k^r$, is represented by a 7 dimensional vector.

\footnote{C++ implementation of GICP algorithm, running on a Core i7 processor, takes around 22 seconds to calculate the pose change between two scans of a Velodyne laser scanner.}
consisting of the robot position, $r_k = [x, y, z]$, and its orientation quaternion, $q_k = [q_1, q_2, q_3, q_4]$, in the world coordinate. Features are assumed to be stationary and the robot moves in 3D space with 6 DOF. The robot motion is described by the following equation:

$$
\begin{bmatrix}
  r_k^x \\
  q_k
\end{bmatrix} = \begin{bmatrix}
  r_{k-1}^x + R(q_{k-1}) \Delta r_{k-1}^x \\
  q_{k-1} \otimes 1
\end{bmatrix}
$$

(1)

where $\otimes$ is quaternion product and $R(q_{k-1})$ is the rotation matrix. The input of the motion model, $\Delta q_{k-1}$ and $\Delta r_{k-1}$ comes from a RANSAC process that calculates the robot pose change. This is different from the 1PRE method that uses a constant velocity motion model. In this work, features are represented in two different ways depending on their linearity index [16]. The first representation uses inverse depth. Each feature in this representation is a 6-element vector $f_i^{ID} = [x_i^f, y_i^f, z_i^f, \theta_f, \phi_f, \rho_f]$, where $\rho_f = 1/||d_i||$ is called inverse depth. In this representation, $r_k^x = [x_k^f, y_k^f, z_k^f]$ is the global coordinate of the camera when feature $f_i$ is observed for the first time and $\theta_f$, $\phi_f$, and $d_i$ are (azimuth, elevation) angles and the depth of feature $f_i$ respectively. It has been shown that this representation is more suitable from filtering point of view when depth uncertainty is large [16].

The second representation for a feature is cartesian representation $f_i^C = [x_i^C, y_i^C, z_i^C]$, which uses the cartesian coordinates of feature $f_i$ in the world coordinates. This representation is more suitable when the depth uncertainty is small [16]. The linearity index [16] is used to determine the suitable feature representation and to switch from one representation to another. In our work, all features are initialized with inverse depth representation. After the initialization, their representations are determined by the linearity index. If the linearity index is below a threshold, cartesian representation is used and if the linearity index is above that threshold, inverse depth representation is used.

2.2. Observation Model

Despite the fact that the 3D LIDAR provides depth data for the RIP (c.f. Fig. 1), we opt to use the same observation model used in the 1PRE method, i.e., use 2D image pixels as observations and adopt the two feature representations as mentioned earlier. The reason is that not every image pixel in the RIP has associated depth data and this prevents us from using depth as an observation in general. Undistorted pixel coordinates of the projection of a 3D laser point in the camera image can be obtained from the pinhole camera model by:

$$
\begin{bmatrix}
  u \\
  v
\end{bmatrix} = \begin{bmatrix}
  u_0 - \frac{f^C}{d} h^C \\
  v_0 - \frac{f^C}{d}
\end{bmatrix}
$$

(2)

where $(u_0, v_0)$, $(d_x, d_y)$ and $\frac{f}{d}$ are image center, vertical/horizontal pixel length and camera focal length, respectively and they are known a priori. $h^C$ is the vector pointing from camera’s focal point toward the feature point and is calculated based on the choice of the feature representation. For inverse depth representation $h_i^{ID} = h_i^{C,ID}$ is given by:

$$
h_i^{ID} = R^T(q_k^C) \begin{bmatrix}
  \rho_f \\
  y_f^C - r^v \\
  z_f^C
\end{bmatrix} + m(\theta_f, \phi_f)
$$

(3)

where $m()$ is a unit vector whose azimuth and elevation angles are $\theta_f$ and $\phi_f$, respectively. Other parameters have been defined earlier. For Cartesian representation, we have $h_i^C = h_i^{C,ID}$ which is expressed by:

$$
h_i^C = R^T(q_k^C) \begin{bmatrix}
  x_w^C \\
  y_w^C \\
  z_w^C
\end{bmatrix} - r^v
$$

(4)

2.3. Preprocessing

2.3.1) Map Management: As the robot moves, obsolete features (features that are predicted but not observed for a number of times) are removed from the map and new features are added to the map. The criteria for feature deletion is the ratio of the number of times a feature is observed to the number of times it is predicted to be in the camera’s field of view. If this ratio falls below 0.5, the feature is discarded from the map. In case that the number of matched features in the current map falls below a predefined value (25 in our implementation), new features are extracted and initialized into the map.
Algorithm 1: Estimation Algorithm

input: \( \hat{x}_{k-1|k-1}, P_{k-1|k-1} \) (EKF estimate at step \( k-1 \)), Frame\(_{k_1}\), Frame\(_{k-1}\) (laser and image data at steps \( k \) and \( k-1 \)), \( \tau \) (threshold for low-innovation points)

output: \( \hat{x}_{k|k} \) (EKF estimate at step \( k \))

A) preprocessing

\[ \hat{x}_{k-1|k-1}, P_{k-1|k-1} = \text{MAP}\_\text{MANAGEMENT} (\hat{x}_{k-1|k-1}, P_{k-1|k-1}) \]

\[ u_{k-1} = [\Delta r_{k-1}^x, \Delta q_{k-1}^y] = \text{RANSAC}\_\text{CALC} (\text{Frame}_{k_1}, \text{Frame}_{k-1}) \]

B) EKF prediction and individually compatible matches

\[ \hat{x}_{k|k-1}, P_{k|k-1} = \text{Predict}(\hat{x}_{k-1|k-1}, P_{k-1|k-1}, u_{k-1}) \]

\[ \hat{h}_{k|k-1}, S_{k|k-1} = \text{Predict}\_\text{Observation}(\hat{x}_{k|k-1}, P_{k|k-1}) \]

\[ z^{IC} = \text{Search}\_\text{IC}\_\text{matches}(\hat{h}_{k|k-1}, S_{k|k-1}) \]

C) 1-Point Hypotheses generation and evaluation

\[ z_i^{hi\_inliers} = \] \( n_{hyp} = 1000 \) (RANSAC max iteration. This value will be updated in the loop)

\[ \text{for } i \leftarrow 0 \text{ to } n_{hyp} \text{ do} \]

\[ z_i = \text{Select}\_\text{Random}\_\text{Match}(z^{IC}) \]

\[ \hat{x}_i = \text{EKF}\_\text{State}\_\text{Update}(z_i, \hat{x}_{k|k-1}) \]

\[ h_i = \text{Predict}\_\text{All}\_\text{Measurements}(\hat{x}_i) \]

\[ z_i^r = \text{Find}\_\text{Match}\_\text{Below}\_\text{Threshold}(z^{IC}, h_i, \tau) \]

if size\( (z_i^r) > \text{size}(z_i^{hi\_inliers}) \) then

\[ z_i^{hi\_inliers} = z_i^r \]

\[ \epsilon = \frac{\text{size}(z_i^{hi\_inliers})}{\text{size}(z_i^{IC})} \]

\[ n_{hyp} = \frac{\log(1-\mu)}{\log(1-\epsilon)} \]

D) Partial update using low-innovation inliers

\[ \hat{x}_{k|k}, P_{k|k} = \text{EKF}\_\text{Update}(\hat{x}_{k|k-1}, P_{k|k-1}, z_i^{hi\_inliers}) \]

E) Partial update using high-innovation inliers

\[ z_i^{hi\_inliers} = [\] \n
for every matched \( z_j \) with innovation above a threshold \( \tau \) do

\[ [\hat{h}_{j}, S_{j}] = \text{Point}\_\text{Prediction}(\hat{x}_{k|k-1}, P_{k|k-1}, j) \]

\[ v_j = z_j - \hat{h}_j; \]

if \( v_j^T S_j^{-1} v_j < \chi^2_{0.01} \) then

\[ z_i^{hi\_inliers} = \text{Add}\_\text{Match}\_\text{j}(z_i^{hi\_inliers}, z_j) \]

if size\( (z_i^{hi\_inliers}) > 0 \) then

\[ \hat{x}_{k|k}, P_{k|k} = \text{EKF}\_\text{Update}(\hat{x}_{k|k}, P_{k|k}, z_i^{hi\_inliers}) \]

2.3.2) Feature Initialization: This is part of the map management procedure. In our implementation, new features are extracted using FAST method [17]. The covariance matrix is also expanded as follows. The depth of feature \( z_i = (u_i, v_i) \) takes the depth value of the corresponding laser data (whose reprojection is closest to the feature). Let \( h_i \) denote the observation model for feature \( z_i \) and \( h_i^{-1} \) denote the inverse observation model. The feature initialization process is performed in the following manner:

When the initial observation of a new feature is obtained, the state vector is expanded with the new feature estimate \( f_i = h_i^{-1}(u_i, v_i, r_i) \) and the covariance matrix is expanded as follows:

\[ P_{k-1|k-1} = J \left( \begin{array}{cc} P_{k-1|k-1} & 0 \\ 0 & R_i \end{array} \right) J^T \]  

where \( J \) has the following structure:

\[ J = \left( \begin{array}{cc} I & 0 \\ -\frac{\partial h_i^{-1}}{\partial u_i} & \frac{\partial h_i^{-1}}{\partial v_i} \end{array} \right) \]

2.3.3) Pose Change Calculation: Since we are not relying on odometry or IMU data, we have to devise a way to calculate the pose change between consecutive steps for the motion model. In this work, we use a RANSAC procedure as follows:

1) Extract SIFT features in two consecutive images. Find the matched features and pair each SIFT feature with its corresponding 3D laser point

2) Randomly select a set of 4 pairs of associated points denoted by \( S_i \) and form the hypothesis [\( \Delta r, \Delta q \)] which minimizes the cost function \( f(S_i) = \sum_{i=1}^{4} ||p_i^k - (R(\Delta q)p_i^k + \Delta r)|| \).

3) Calculate the support of the hypothesis as the number of points whose association error are below a threshold, \( tr_a \). The association error for pair \( A_i = (p_i^k, p_i^{-1}) \) is \( \epsilon(A_i) = ||p_i^k - (R(\Delta q)p_i^k + \Delta r)|| \).

4) Update variable \( MaxSteps \) (the maximum number of steps for the RANSAC process). This is simply calculated from the following formula in each step \( MaxSteps = min(MaxSteps, log(1-\mu)/log(1-\epsilon)) \) where \( \mu \) is the level of confidence that the resulted correspondence is outlier-free (usually assigned to 0.99) and \( \epsilon \) is the true inlier ratio. In practice, this probability is approximated by the inlier percentage of the best hypothesis so far which is updated step by step.

5) After \( MaxSteps \) iterations, pick the hypothesis with biggest support and use the corresponding data points to calculate [\( \Delta r, \Delta q \)] by SVD (Singular Value Decomposition). The cost function is similar to \( f(S_i) \) in step 2 but summation is performed over all data points. The result from this step is the input to the motion model in Eq. (1).

2.4. EKF prediction and individually compatible matches

2.4.1) Prediction: In this part of the algorithm, the state vector and its covariance are predicted based on the motion model (Eq. (1)) and its Jacobians:

\[ \hat{x}_{k|k-1} = f_c(\hat{x}_{k-1|k-1}, u_{k-1}) \]

\[ P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + G_{k-1}Q_{k-1}G_{k-1}^T \]
where $Q_{k-1}$ is the covariance matrix of the input vector $u_{k-1}$, and $G_{k-1} = \partial f_v/\partial u|_{x_{k-1}, u_{k-1}}$ and $F_{k-1} = \partial f_v/\partial x|_{x_{k-1}, u_{k-1}}$ are jacobians of the motion model with respect to the input and state vector respectively.

2.4.2) Predict Observation: The result of the prediction step is used to assist feature matching between two frames. After the state vector is predicted, we can predict which features are visible in the current frame and for each of those features, the observation model $h(\hat{x}_{k|k-1})$ gives the predicted pixel coordinates where we expect to see the feature. By projecting state covariance into image plane through observation model, we also find a region within which we expect to find the actual match. The observation prediction equations are as follows:

$$
\hat{h}_{k|k-1} = h(\hat{x}_{k|k-1}) \quad (8a)
$$

$$
S_{k|k-1} = H_k P_{k|k-1} H_k^T + R_k \quad (8b)
$$

where $H_k = \partial h/\partial x|_{x_{k|k-1}}$ is the jacobian of the observation model and $R_k$ is the covariance of the observation noise (in practice it is assigned to an identity matrix multiplied by pixel noise variance $\sigma_{pixel}$). Eigenvalues of matrix $S_{k|k-1}$ in Equation (8b) determine the region within which we anticipate to find the match for each feature.

2.4.3) Finding Individually Compatible Matches: For each feature, the match is found via a correlation based procedure explained below. At the time when a feature is initialized into the state vector, we save the image patch centered at the feature. As the vehicle moves, the matrix for this patch is updated based on $u_{k-1}$. Update is performed by warping the patch based on our estimate of the vehicle pose shift between two steps. In other words, at each step the appearance of the patch is predicted and the correlation between this patch and the predicted region for the feature is then computed. This result is a 2D matrix, of which the maximum correlation value, if bigger than a threshold (0.6 in our implementation), results in a 2D matrix that is the maximum correlation value, if bigger than a threshold (0.6 in our implementation), results in a 2D matrix, of which the maximum correlation value, if bigger than a threshold (0.6 in our implementation), results in a 2D matrix, of which the maximum correlation value, if bigger than a threshold (0.6 in our implementation), results in a 2D matrix, of which the maximum correlation value, if bigger than a threshold (0.6 in our implementation), results in a 2D matrix, of which the maximum correlation value, if bigger than a threshold (0.6 in our implementation), results in a 2D matrix, of which the maximum correlation

$$
\chi^2(k|k-1) = \sum_{i=1}^{m} (z_{li} - h(\hat{x}_{k|k-1}))^T S_{k|k-1}^{-1} (z_{li} - h(\hat{x}_{k|k-1})) \quad (9b)
$$

$$
P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (9c)
$$

2.5. Partial Update Using High Innovation Inliers

After performing the partial update, some of the feature correspondences that were classified as outliers in the previous step due to their high innovation values may become inliers [16]. These features are called high innovation inliers. In this stage, we will incorporate these inliers into the filter and use them to perform another update of state. A $\chi^2$ test is performed on the outlier points resulted in the previous stage. Those points that pass the test are used for the partial update. The result is the estimated state for step $k$. In next section, we will evaluate the performance of this algorithm in a real situation.

3. Results

We tested our method with a publicly available real-world dataset collected from a 3D LIDAR (Velodyne HDL-64E) and an omnidirectional camera system (Point Grey Ladybug3) mounted on the roof of a Ford F-250 vehicle [15]. Ground truth data is also available from a high end Inertial Measurement Unit (IMU) (Applanix POS-LV). Navigation coordinate and the arrangement of the sensors on the car are shown in Fig. 2. The dataset was collected in a challenging urban scenario where the vehicle moved at a maximum speed of 20 mph and there was incoming traffic from time to time. In this dataset, the vehicle moved through a 1.443 km trajectory in downtown Dearborn, Michigan, in 366 seconds. Sampling time for the LIDAR and camera data capture is 0.1s and 0.125s, respectively.

3.1. Visualization of Feature Correspondence and Tracking

To facilitate algorithm development, we developed a Matlab graphic tool to visualize the feature correspondence and tracking process at each step for the proposed method. The tool is used to observe, fine-tune and validate the method’s ability in tracking features, detecting and rescuing high innovation inliers, and indentifying features on moving objects (or with erroneous depth) as outliers. Fig. 3 depicts three screen-snapshots at three different time steps of the navigational
task: the first row contains the full camera views and the second row shows their zoom-in views. In this figure, a feature’s mean and uncertainty (i.e., estimated covariance) are projected onto the image plane and denoted by a "+" and an ellipsoid, respectively. Individually compatible matches, low-innovation inliers, high-innovation inliers, outliers, and unmatched features are depicted in green, thick red, thin red, pink and blue, respectively.

In Fig. 3a the vehicle is moving forward while other cars are travelling in all lanes. As it can be seen in the zoom-in view (Fig. 3d), features on moving vehicles are identified as outliers (marked in pink). There are some outliers on static objects as well, which are caused by the inaccurate depth values of the features. These features pass the test for individually compatible matches because their depth errors are not big enough to fail the test. However, the RANSAC procedure (stage C of the method) identifies them as outliers and our method excludes their use for the first partial state update. Fig. 3b depicts another scenario with moving objects where the vehicle is turning and following another car. The zoom-in view (Fig. 3e) shows that features on the moving car are categorized as outliers and are excluded from the state update process. It is noted that in a conventional EKF-SLAM method these features are used in the Kalman filter state update and may introduce error in state estimate.

Finally, Fig. 3c and Fig. 3f show a case where high innovation inliers are detected and rescued after the first partial update in our method. In this case, three features did not pass the innovation threshold test (in stage C of the method) and were initially classified as outliers. However, after the first partial state update, they are detected as inliers (i.e., high-innovation inliers) in stage E of the method and used to further update the state. This two-stage inlier detection and state update process reduces the chance of using outliers for state update and thus improves the performance of the EKF method.

3.2. Estimation Accuracy

In order to evaluate the performance of the proposed method, we compare it with two other methods that rely on the result of laser data registration. In our experiment we only used the data from the front camera and the related laser data. (Ladybug3 comprise six cameras, one of which looks forward.) The two methods considered are:

1) GICP dead-reckoning (using full laser data). This method calculates the pose change between consecutive scans in each step and the vehicle’s pose is calculated by integrating the incremental pose changes. The original implementation of GICP is used in our experiment.

2) iSAM [18] method that obtains pose estimates by solving a pose-graph. A pose-graph is a graph representation of the pose estimation problem in which nodes are vehicle poses and edges represent constraints between nodes. In our experiment, full laser data is used and pose-graph is constructed by creating GICP-derived constraints between consecutive vehicle poses and additional GICP-derived constraints between poses three steps away. Kaess’ open-source incremental Smoothing And Mapping (iSAM) package [18] is used to solve the pose-graph.

Estimated trajectories for the three methods are shown in Figs. 4 and 5 along with the ground truth trajectory. The inertial coordinate (z points forward, x points right and y points downwards) is used for navigation and state estimate. The vehicle starts moving from \((0,0,0)\) in this coordinate and returns to the same location at the end. The norms of the positional errors of the three methods are depicted in Fig. 6 and the final positional errors and error percentages are summarized in Table I.

It can be seen from Table I that GICP has the smallest final error among the three methods. However, as shown in Fig. 6 the proposed method always has the smallest positional error along the trajectory except for the final 10% of the trajectory where it is outperformed by GICP and the final a few steps where it is outperformed by iSAM. Therefore, it can be said that the proposed method has the best overall pose estimation performance. A closer look into the dataset and the result reveals that iSAM has a better performance than GICP until timestamp 283.7s, starting from which the GICP has a smaller positional error. This demonstrates that the use of the additional constraints in iSAM usually helps to improve pose estimation accuracy. The deterioration of accuracy after
Fig. 3. Feature tracking performance of the proposed method: (a), (b), (d) and (f) demonstrate that features on moving objects or features with erroneous depth are identified as outliers. (c) and (f) show a scenario where high innovation inliers are detected and rescued after the initial filter update.

Fig. 4. Comparison of the ground truth vs. other methods (eye bird view)

timestamp 283.7s might be caused by occasional large error introduced by the constraints. We will look into this issue and investigate an approach to alleviate the effect. Considering that we only used 1/5 of the data in the experiment, there is room to improve the proposed method’s accuracy if all data are used.

TABLE I

<table>
<thead>
<tr>
<th>method</th>
<th>GLICP</th>
<th>iSAM</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute error (m)</td>
<td>16.75</td>
<td>26.59</td>
<td>27.93</td>
</tr>
<tr>
<td>final error percentage (%)</td>
<td>1.16</td>
<td>1.84</td>
<td>1.93</td>
</tr>
</tbody>
</table>

Fig. 5. Comparison of the ground truth vs. other methods (3D view)

Fig. 6. Transformation Error Norm comparison between iSAM (blue dashed line), proposed algorithm (black dashed-dot line) and GICP (green line)

4. Conclusion

We have presented an extended 1-Point RANSAC EKF method for 6 DOF vehicle pose estimation. The method fuses 3D LIDAR and visual data in both depth initialization and motion prediction stages. RANSAC is employed to calculate vehicle pose change for the motion model and to identify inlier features for state update. The EKF state update uses a two-step refinement approach. First, inliers (features with low innovation values) identified by the RANSAC process are used to update the state. Second, features with high innovation
values (i.e. outliers from previous step) are re-examined with
the updated state to find new inliers which are then used
to further update the state for state estimation refinement. A
dynamic map management scheme is devised to discard the
obsolete features and to replace them with new ones and thus
maintain a suitable number of features for state filtering.

The data fusion scheme, reduces the number of laser data
points to be processed without sacrificing the pose estimation
accuracy and solves the scale problem of monocular based
visual SLAM approaches. The RANSAC procedures help
to prevent erroneous data (features on moving objects and
inaccurate depth measurements) from being used in state
update. This allows the proposed method to deal with dynamic
environment to certain extent.

Our experiments on a real world dataset shows that the
proposed method has overall better performance compared to
GICP and the iSAM method. The proposed method can be
used for pose estimation of unmanned ground vehicles in GPS
denied environments.

REFERENCES


Amirhossein Tamjidi

Amirhossein Tamjidi received his BS degree in Electrical Engineering from Tabriz University, Tabriz, Iran, in 2006. He received his MS degree in Control Engineering from K.N. Toosi University of Technology, Tehran, Iran, in 2009. He is currently working towards a PhD in Systems Engineering at the University of Arkansas at Little Rock, AR. He is a student member of IEEE and his current research interests include pose estimation and 3D computer vision in robotics.

Cang Ye

Cang Ye received B.E. and M.E. degree from the University of Science and Technology of China, P. R. China in 1988 and 1991, respectively. He received his Ph.D. degree from the University of Hong Kong in 1999.

He is currently an Associate Professor in Department of Systems Engineering, University of Arkansas at Little Rock, AR. He was a research investigator at University of Michigan, Ann Arbor, MI from 2003 to 2005. His research interests are in mobile robotics, robotic assistive technology, fuzzy systems and reinforcement learning.

Dr. Ye is a senior member of IEEE and a member of Technical Committee on Robotics and Intelligent Sensing, IEEE Systems, Man, and Cybernetics Society.