A Framework for Fast and Robust Visual Odometry

Meiqing Wu, Siew-Kei Lam, *Member, IEEE* and Thambipillai Srikanthan, *Senior Member, IEEE*

***Abstract*—Knowledge of the ego-vehicle's motion state is essential for assessing the collision risk in Advanced Driver Assistance Systems (ADASs) or autonomous driving. Vision-based method for estimating the ego-motion of vehicle, i.e. visual odometry, faces a number of challenges in uncontrolled realistic urban environments. Existing solutions fail to achieve a good trade-off between high accuracy and low computational complexity. In this paper, a framework for ego-motion estimation that integrates runtime-efficient strategies with robust techniques at various core stages in visual odometry is proposed. Firstly, a pruning method is employed to reduce the computational complexity of KLT feature detection without compromising on the quality of the features. Next, three strategies, i.e. smooth motion constraint, adaptive integration window technique and automatic tracking failure detection scheme, are introduced into the conventional KLT tracker to facilitate generation of feature correspondences in a robust and runtime efficient way. Finally, an early RANSAC termination condition is integrated with the Gaussian-Newton optimization scheme to enable rapid convergence of the motion estimation process while achieving robustness. Experimental results based on the KITTI odometry dataset show that the proposed technique outperforms state-of-the-art visual odometry methods by producing more accurate ego-motion estimation in notably lesser amount of time.**

***Index Terms*—Visual odometry, ego-motion, collision avoidance, feature detection, feature tracking, motion estimation, ADASs, autonomous vehicles**

I. Introduction

THE knowledge of ego-vehicle's motion state relative to the road serves as the foundation for assessing the risk of collision in Advanced Driver Assistance Systems (ADASs) and autonomous driving. Conventional means of obtaining the ego-motion of the vehicle rely on Inertial Measuring Units (IMUs) or Global Positioning System (GPS). However, IMUs cannot provide all the necessary information like the pitch angle and the roll rate [1], while GPS cannot be relied upon to obtain the vehicle's ego-motion in GPS-denied environment e.g. under bridges or urban jungles [2]. As such, vision based methods for ego-motion estimation are becoming increasingly popular as they overcome the drawbacks of IMUs and GPS. In addition, camera-based systems offer other advantages e.g. ease of maintenance and integration into other functionality modules, and reduced cost [3]. Ego-motion estimation that relies solely on vision-based sensing is referred to as visual odometry [4].

Visual odometry in the context of external traffic environment faces huge challenges. Firstly, unlike the indoor environment, the external traffic scene is totally uncontrolled. The scene can be cluttered and contains a lot of moving objects. The scene can also be subjected to inconsistent illumination. As such, visual odometry algorithms must be able to work robustly under such challenging situations. Secondly, computing systems in vehicles are embedded systems with restricted computational resources. The proposed algorithm should therefore be of low computational complexity to allow for in-vehicle deployment.

In this paper, a framework for robust and runtime-efficient visual odometry is proposed. The proposed framework integrates runtime-efficient strategies with robust techniques at each of the core stages in visual odometry. Specifically, a pruning method is employed to reduce the computational complexity of KLT feature detection without compromising on the quality of detected features. Ego-motion prior is leveraged on to determine a better initial position for KLT tracking process to increase the chance for correct convergence, which significantly increases the proposed technique's robustness in challenging environments. The robustness of the proposed technique is further enhanced by adopting an automatic tracking failure detection scheme during feature tracking. In addition, an adaptive and small integration window for each feature is set during tracking based on its distance from the ego-vehicle. This significantly reduces the computational complexity. Finally, we apply an early RANSAC termination condition in the Gaussian-Newton based motion estimation process to further increase the algorithmic robustness and reduce the algorithmic computation time. Experimental results based on the KITTI dataset show that the proposed technique outperforms state-of-the-art visual odometry methods by producing more accurate ego-motion estimation in notably shorter amount of time.

This paper is organized as follows: Section II reviews the existing works in visual odometry. Section III formulates visual odometry as a mathematical minimization problem. The proposed algorithm is presented in Section IV. A comprehensive evaluation of the proposed method with existing state-of-the-art methods using the well-known KITTI odometry dataset is presented in Section V, and Section VI concludes the paper.

II. Related Works

The core stages of visual odometry typically consist of feature correspondences setup that relies on feature detection and feature tracking, and motion estimation that solves a mathematical optimization problem on the set of correspondences [5], [6].

A. Feature Correspondence Setup

In the field of visual odometry, point features rather than edge features are preferred since they can be accurately positioned [6]. Point features consist of corner and blob. Popular

Meiqing Wu, Siew-Kei Lam and Thambipillai Srikanthan are with School of Computer Science& Engineering, Nanyang Technological University, Singapore (e-mail: wume0007@e.ntu.edu.sg).

Manuscript received March 21, 2016; revised September 12, 2016.

corner feature detectors are Harris [7], KLT [8], FAST [9], or even much simpler maxima and minima of Sobel filter response [10], etc. Popular blob feature descriptors include SIFT [11], SURF [12], FREAK [13], BRIEF [14], etc. Once features are extracted, they will be tracked or matched to find their correspondences across frames. The former like Harris and KLT aims to detect features in the first frame and track them in the second frame using local search techniques. The latter like SIFT, SURF aims to detect features independently in both of the images and matches them based on a certain similarity measure.

The extraction of reliable feature correspondences across frames in realistic environments plays a deterministic role in the success of visual odometry [15]. In addition, as illustrated in [10], [15], the computational hot spots of visual odometry lies in feature correspondences extraction.

B. Motion Estimation

Existing techniques for motion estimation can be divided into two categories [5]: monocular vision based [3], [4], [16]–[19] and stereo vision based [4], [10], [15], [20]–[23], depending on the dimension of the features. In the first case, since the features are encoded in 2D, a relative scale factor needs to be determined using methods like trifocal tensor [4]. At least 5 feature pairs are required to obtain the solution [4], which is found by determining the transformation that minimizes the re-projection error of the triangulated points in each image. For stereo vision, the 3D scene structure can be directly reconstructed through triangulation of the stereo rig. The minimal-case solution involves 3 non-collinear correspondences [4]. When feature correspondences are specified in Euclidean space, i.e. 3D-to-3D, the solution is found by determining the alignment transformation that minimizes the distances between the correspondences. Instead of minimizing the residuals in Euclidean space, a better solution is to work in the image space (i.e. 3D-2D) [23]. In this case, the solution is determined by minimizing the re-projection error.

In order to increase the robustness of motion estimation, robust estimation methods like M-estimations [24], RANSAC [25], [26] have been utilized to increase the accuracy of motion estimation in the presence of noisy or erroneous feature correspondences. In addition, visual odometry is a dead-reckoning algorithm and is prone to error accumulation over time [15]. Therefore, the estimated camera pose can easily drift from the real path. To deal with this problem, some works combine other sensor data from GPS [27] or IMUs [28]–[30] to improve the positioning accuracy. Another popular solution is the bundle adjustment algorithm [28], [31] that imposes geometrical constraints over multiple frames. However, this approach is time consuming. Recently, [15] proposes a technique to reduce the motion drift by introducing an augmented feature set that contains the accumulated information of tracked features over all frames.

C. Main Contributions of Our Work

The existing solutions fail to achieve a good balance between high accuracy and low computational complexity [32]. For example, the work in [10] is able to achieve real-time performance at the expense of sacrificing estimation accuracy. On the contrary, the work in [15] achieves high estimation accuracy, but is time consuming. The proposed work aims to bridge this gap with a solution for visual odometry that can produce accurate results in short computation time. Hence, an ego-motion estimation framework that integrates suitable runtime-efficient strategies with robust techniques at various core stages in visual odometry is proposed. The main contributions of the proposed method are as follow:

1) A fast corner detector with pruning technique that reduces the computational complexity of detecting high quality corner features.
2) A robust and compute-efficient KLT tracker that employs smooth motion constraint, adaptive window technique and automatic tracking failure detection scheme.
3) A robust and fast motion estimation method that is based on Gaussian-Newton optimization scheme with early RANSAC termination condition.
4) The above contributions are integrated into a framework for fast and robust visual odometry. Experimental results on the widely known KITTI evaluation platform [32] demonstrate that the proposed framework can produce accurate ego-motion estimation. In terms of accuracy, the proposed algorithm is ranked highly in the KITTI odometry evaluation platform. In addition, the proposed algorithm performs notably faster than most of the techniques in the KITTI odometry evaluation platform.

III. Problem Formulation

A camera installed on a moving vehicle is subjected to six degrees of freedom (DOF). That is, it can be translated in three perpendicular *x*-*y*-*z* axes, denoted as (tx, ty, tz) (in meter), and rotated about the three axes, denoted as (rx, ry, rz) (in radian). The goal of visual odometry is to obtain the value of $\boldsymbol{X} = (tx, ty, tz, rx, ry, rz)^T$ at each discrete time instance.

The motion of a camera installed on the moving vehicle from the previous frame I_{n-1} to current frame I_n can be represented by the matrix $\boldsymbol{M}_n \in \mathbb{R}^{4*4}$ as shown in Eq. 1:

$$\boldsymbol{M}_n = \begin{bmatrix} \boldsymbol{RO}_n & \boldsymbol{tr}_n \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \tag{1}$$

$$\boldsymbol{RO}_n = \begin{bmatrix} cy * cz & -cy * sz & sy \\ sx * sy * cz + cx * sz & -sx * sy * sz + cx * cz & -sx * cy \\ -cx * sy * cz + sx * sz & cx * sy * sz + sx * cz & cx * cy \end{bmatrix} \tag{2}$$

$$\boldsymbol{tr}_n = [tx \quad ty \quad tz]^T \tag{3}$$

$$\begin{aligned} sx &= sin(rx); cx = cos(rx) \\ sy &= sin(ry); cy = cos(ry) \\ sz &= sin(rz); cz = cos(rz) \end{aligned} \tag{4}$$

The camera pose $\boldsymbol{C}_n$ from the point of initialization can be obtained by concatenating all the transformations $\boldsymbol{M}_n$ as

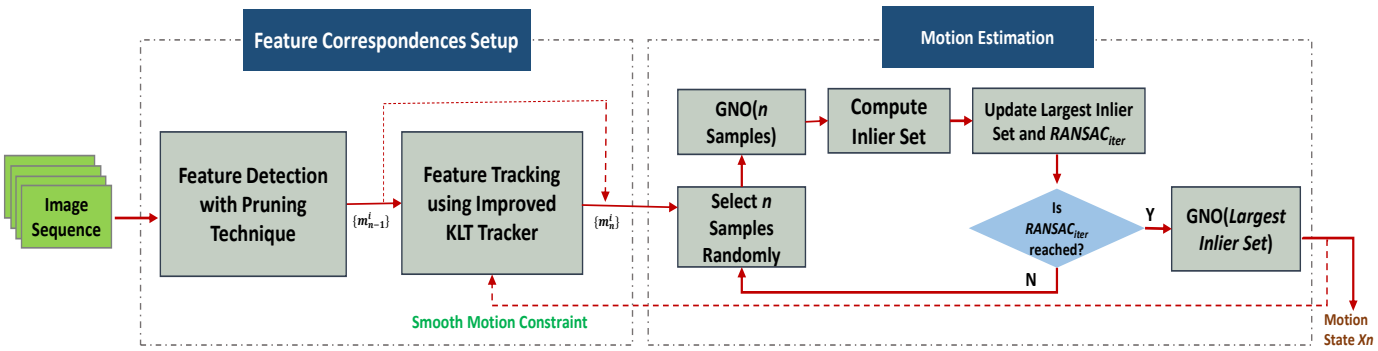


Fig. 1. Overview of the proposed visual odometry framework. The proposed framework consists of two stages. The first stage extracts feature correspondences by applying a pruning technique to the KLT corner detector for feature detection and improving the KLT feature tracker with three strategies for feature tracking. The second stage estimates motion parameters based on Gaussian-Newton optimization scheme (GNO) which is integrated with the early RANSAC termination condition.

shown in Eq. 5. $\boldsymbol{T}_n$ in Eq. 6 represents the scene motion, which is the inverse of the camera motion.

$$\boldsymbol{C}_n = \prod_{i=1}^{n} \boldsymbol{M}_i \tag{5}$$

$$\boldsymbol{T}_n = \boldsymbol{M}_n^{-1} = \begin{bmatrix} \boldsymbol{R}_n & \boldsymbol{t}_n \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \tag{6}$$

Assume that the set of static points (features) in Euclidean space observed in frame I_{n-1} are $\{\boldsymbol{p}_{n-1}^i = (x_{n-1}^i, y_{n-1}^i, z_{n-1}^i)^T | i = 1, 2, \ldots, k\}$ and their correspondences in frame I_n are $\{\boldsymbol{p}_n^i = (x_n^i, y_n^i, z_n^i)^T | i = 1, 2, \ldots, k\}$. Then the relationship between a pair of correspondences $\boldsymbol{p}_{n-1}^i$ and $\boldsymbol{p}_n^i$ through $\boldsymbol{T}_n$ is shown in Eq. 7:

$$\boldsymbol{p}_n^i = \boldsymbol{R}_n * \boldsymbol{p}_{n-1}^i + \boldsymbol{t}_n \tag{7}$$

Therefore the ego-motion parameters $\boldsymbol{X}_n = (tx, ty, tz, rx, ry, rz)^T$ can be found by minimizing the residual function in Eq. 8, where w_i is the weighting factor that denotes the contribution of point i to the least square solution.

$$E = \arg\min_{\{\boldsymbol{X}_n\}} \sum_{i=1}^{k} w_i \| \boldsymbol{p}_n^i - \boldsymbol{R}_n * \boldsymbol{p}_{n-1}^i - \boldsymbol{t}_n \|^2 \tag{8}$$

Eq. 8 calculates the residual in Euclidean space. However, as discussed in [23], [33], stereo triangulation error can be highly anisotropic and correlated. As such, a recommended approach is to compute the residual in image space, where the noise level is similar for all components of the measurement vector:

$$E = \arg\min_{\{\boldsymbol{X}_n\}} \sum_{i=1}^{k} w_i \| \boldsymbol{m}_n^i - h(\boldsymbol{R}_n * g(\boldsymbol{m}_{n-1}^i) + \boldsymbol{t}_n) \|^2 \tag{9}$$

Where $\boldsymbol{m}_n^i = (u_n^i, v_n^i, d_n^i)^T$ is the projection of $\boldsymbol{p}_n^i$ in the image frame I_n. g is the triangulation equation, while $h = g^{-1}$ is the projection function. b and f are the corresponding baseline and focus length.

$$\boldsymbol{p}_n^i = g(\boldsymbol{m}_n^i) = \begin{cases} x_n^i = (u_n^i - u_0) * b / d_n^i \\ y_n^i = (v_n^i - v_0) * b / d_n^i \\ z_n^i = f * b / d_n^i \end{cases} \tag{10}$$

$$\boldsymbol{m}_n^i = h(\boldsymbol{p}_n^i) = \begin{cases} u_n^i = f * \frac{x_n^i}{z_n^i} + u_0 \\ v_n^i = f * \frac{y_n^i}{z_n^i} + v_0 \\ d_n^i = f * b / z_n^i \end{cases} \tag{11}$$

The aim of feature detection is to identify a set of points $\{\boldsymbol{m}_{n-1}\}$ in frame I_{n-1}, while the aim of feature tracking is to identify $\{\boldsymbol{m}_n\}$, which are the correspondences of $\{\boldsymbol{m}_{n-1}\}$ in frame I_n. Motion estimation computes the ego-motion parameters $\boldsymbol{X}_n = (tx, ty, tz, rx, ry, rz)^T$ by solving Eq. 9.

IV. Proposed Algorithm

As highlighted in [15], the extraction of reliable feature correspondences that correspond to the static scene plays an essential role in the success of visual odometry. The KLT feature tracker [8], which consists of corner feature detection and tracking, is a widely accepted method for feature correspondence extraction [22], [34]–[37]. Although KLT has been shown to be one of the best feature tracker, direct adoption of KLT can lead to inaccurate tracking results in highly complex urban environments as will be shown in the following discussion and experimental results. In addition, KLT is time consuming [38]. On the other hand, the extracted feature correspondences may come from self-moving objects. Direct motion estimation based on feature correspondences that are contaminated with self-moving or inaccurately tracked features will lead to inaccurate results.

The proposed technique aims to overcome the limitations of existing solutions by integrating strategies to achieve robust visual odometry at low computational complexity at various core stages of the ego-motion estimation framework. Fig.1 shows the overview of the proposed framework. Taking the image sequence from the stereo rig and the corresponding disparity map as the input, the proposed visual odometry framework consists of the following two stages: 1) Feature correspondences setup, and 2) Motion estimation.

The first stage incorporates techniques for robust and low-complexity feature correspondences setup. This stage further

consists of the following two steps: *(i) Low-complexity corner detection with pruning, and (ii) Robust and low-complexity feature tracking using improved KLT tracker.* For each time step *n*, corner detection is applied on the previous left image $\boldsymbol{I}_{n-1}$ to extract a set of corner features $\{\boldsymbol{m}_{n-1}\}$ using a pruning technique. The computational complexity of corner detection is significantly reduced due to the pruning process without compromising on the quality of the extracted corner features. Next, for each feature $\boldsymbol{m}^i_{n-1}$ in $\{\boldsymbol{m}_{n-1}\}$, its correspondence $\boldsymbol{m}^i_n$ in current left image $\boldsymbol{I}_n$ is identified using an improved KLT tracker. Smooth motion constraint is utilized to determine a better starting point for KLT tracking process, which leads to fast and accurate convergence during the tracking. In addition, an adaptive window technique, which is based on the distance of the feature from the ego-vehicle and the smooth motion constraint, is employed to track each feature. This significantly reduces the runtime complexity. Finally, an automatic tracking failure detection scheme is adopted during feature tracking to further increase the robustness of the method.

The second stage incorporates techniques for robust and fast motion estimation. Given the set of feature correspondences $\{\boldsymbol{m}_{n-1}\}$ and $\{\boldsymbol{m}_n\}$, the motion parameters $\boldsymbol{X}_n = (tx, ty, tz, rx, ry, rz)^T$ are computed by solving the function formulated in Eq. 9 using Gaussian-Newton method. In order to increase the robustness and also decrease the computation time, RANSAC with an early termination condition is enabled to remove the outliers that do not exhibit coherent movement.

In the following sub-sections, detailed descriptions of each stage for the proposed framework are provided.

A. Low Complexity Corner Detection with Pruning

In order to detect corners, KLT computes a corner response λ_2 for each pixel:

$$\lambda_2 = \frac{(a+c) - \sqrt{(a-c)^2 + 4b^2}}{2} \tag{12}$$

λ_2 corresponds to the minimum eigen-value of the matrix $\boldsymbol{A}$, which approximates a local auto-correlation function:

$$\boldsymbol{A} = \begin{bmatrix} \sum_w \alpha(x) I_x^2 & \sum_w \alpha(x) I_x I_y \\ \sum_w \alpha(x) I_x I_y & \sum_w \alpha(x) I_y^2 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \tag{13}$$

Where $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$ are the horizontal and vertical gradients respectively, and $\alpha(x)$ is the weight function, which can be a simple box window or Gaussian window. The eigen-values λ_1 and λ_2 of $\boldsymbol{A}$ (where $\lambda_1 >= \lambda_2$) represent the two dominant directions of intensity change.

A threshold is applied on the corner response λ_2 to remove the obvious non-corners. The rest of the pixels are then ranked in descending order of their corner response and the pixels with the highest corner response are selected as corners after applying non-maximal suppression.

In order to identify good features, KLT computes the complex corner measure λ_2 for each pixel and chooses the ones with high λ_2 value. However, the obvious non-corners, i.e. the smooth and low curvature regions, constitute a large majority of the image in most cases. This incurs a lot of computational redundancies when the complex corner measure for the obvious non-corners are computed. As such, a pruning method to select only the most relevant features for tracking is employed. The pruning method is explained as follows.

Expanding Eq. 12, we obtain:

$$\begin{aligned} \lambda_2 &= \frac{(a+c) - \sqrt{(a-c)^2 + 4b^2}}{2} \\ &= \frac{(a+c) - \sqrt{(a+c)^2 - 4(ac - b^2)}}{2} \end{aligned} \tag{14}$$

It can be observed that λ_2 is heavily influenced by the term $(ac - b^2)$ as the two $(a+c)$ terms get cancelled out. Hence the goal of identifying pixels with large λ_2 can be simplified as one that identifies pixels with large $(ac - b^2)$. In addition, in order to maximize $(ac - b^2)$, the first term ac should be large. In other words, pixels that have small ac values are less likely to be good features.

Based on the analysis above, when applying an appropriate threshold to discard pixels with low ac values, the remaining pixels contain the final good KLT corners. Fig. 2(b) shows the corner candidates selected by applying threshold = 0.05 * max(ac).

In addition, the I_x^2 and I_y^2 terms in the a and c value can be approximated with the absolute values of I_x and I_y respectively as follows:

$$a' = \sum |I_x|; \quad c' = \sum |I_y| \tag{15}$$

This eliminates the multiplication operations involved in the squared gradients. As such, pixels that have high $a'c'$ values will also have high ac values and therefore are highly likely to be good KLT corners. Fig. 2(c) shows that the $a'c'$ map does not lose the distinctive corner regions. In addition, as shown in Fig. 3, when the threshold for $a'c'$ map is reduced, distinctive corner and edge features are released before texture and flat regions. Hence, $a'c'$ can be used as a corner indicator measure as it can effectively distinguish the corner regions from the non-corner regions.

Therefore, as illustrated in Algorithm 1, instead of computing the complex corner measure as formulated in Eq. 12 for every pixel, a much simpler corner candidate indicator $a'c'$ as formulated in Eq. 15 is utilized as a pruning measure to remove the non-corner regions quickly and generate a small set of corner candidates. The final KLT corner measure is computed only on the small set of corner candidates and corners with highest measure value are chosen. By doing this, the computational complexity for corner detection is reduced and the quality of the extracted features is not compromised. Readers are referred to [39] for a detailed discussion.

We would like to point out that compared to the Sobel edge key points which are calculated based on the sobel response over only a single point, the pruning metric $a'c'$ is the product of the value of a' and c' that are calculated based on the sum of sobel response over their respective neighborhood window. A point that has a high sobel response may not necessarily have a large $a'c'$ value and vice versa. The pruning metric $a'c'$ has a tighter association with the conventional KLT corner measure and it ensures that corners are detected in the

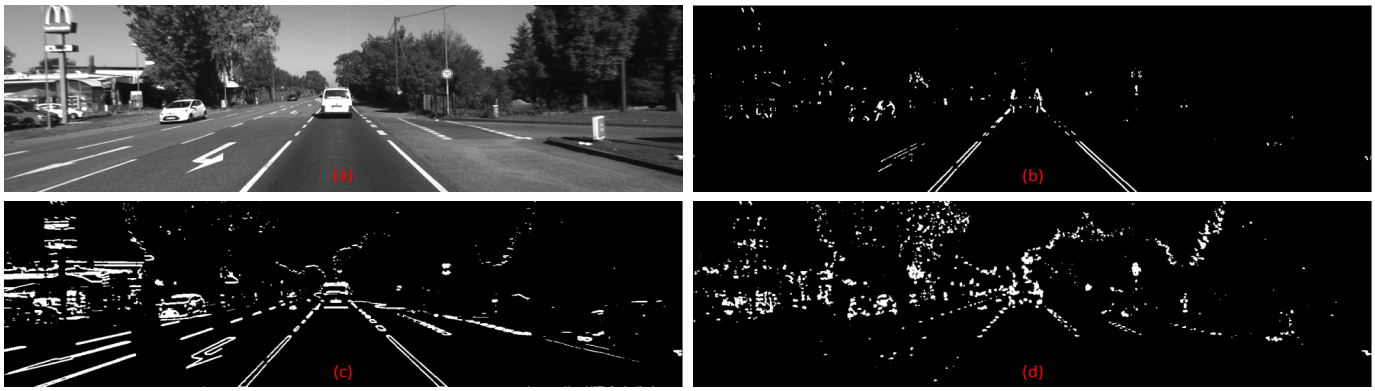


Fig. 2. Illustration of corner detection using different metrics: (a) Original image; (b) Corner candidates selected using ac at *threshold*=0.05*max(ac); (c) Corner candidates selected using $a'c'$ at *threshold*=0.05*max($a'c'$); (d) Corner regions with $\lambda_2 > 0.05$*max(λ_2).

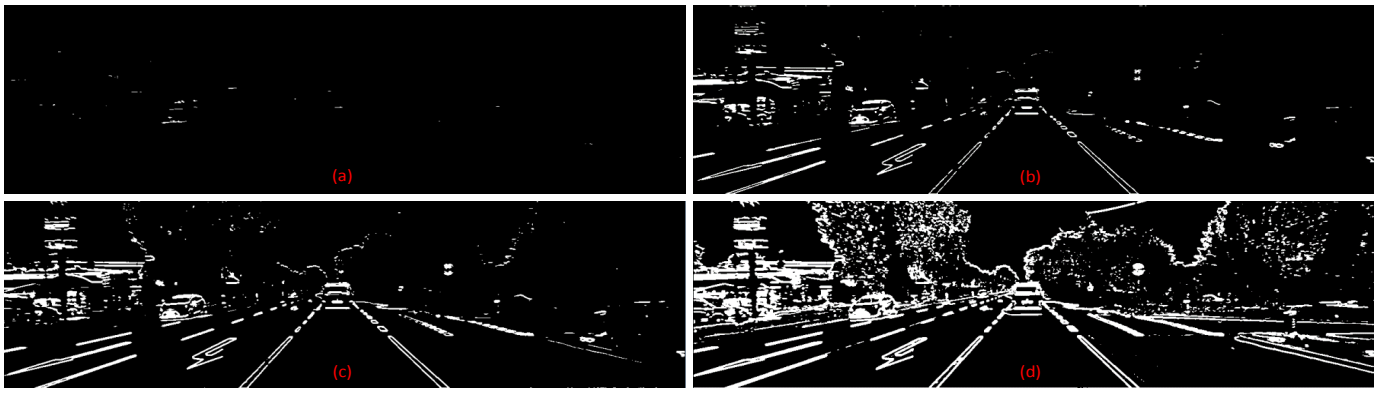


Fig. 3. $a'c'$ map at various thresholds: (a) 0.5; (b) 0.1; (c) 0.05; (d) 0.01.

same order as the conventional KLT corner detection method, that is, corners with higher KLT corner quality are detected earlier than those that are of lower corner quality. As such the proposed pruning technique enables rapid corner detection without losing distinctive KLT corners.

B. Feature Tracking using Improved KLT Tracker

Mathematically, the KLT tracking process is formulated as a least square problem to minimize a residual function over an integration window as defined in Eq. 16.

$$E = \arg \min_{\{d_x, d_y\}} \sum_{x=u_x-r}^{u_x+r} \sum_{y=u_y-r}^{u_y+r} \left(I_{n-1}(x,y) - I_n(x+d_x, y+d_y) \right)^2 \tag{16}$$

Where $I_{n-1}(u_x, u_y)$ and $I_n(u_x + d_x, u_y + d_y)$ are the correspondence located in image I_{n-1} and I_n respectively. $\boldsymbol{d} = [d_x \quad d_y]^T$ is the optical flow for feature $I_{n-1}(u_x, u_y)$. r is the radius of the integration window.

The way KLT solves Eq. 16 using an iterative Newton Raphson method consists of a sequence of search operations that try to find a image patch with size $[2r+1, 2r+1]$ in image I_n such that there is minimum intensity difference between it and the image patch in image I_{n-1} of size $[2r+1, 2r+1]$ with feature $I_{n-1}(u_x, u_y)$ in the center. As observed in [40], starting the search process in the position (u_x, u_y) in image I_n, a small integration window size is preferred to increase the accuracy by avoiding smoothing out the image details. However, the integration window must also be sufficiently large to cater to the displacement of feature that undergoes large motion to increase robustness. In order to obtain a good tradeoff between local accuracy and robustness when choosing the integration window size, the pyramidal implementation of KLT has been introduced in [40]. However, this approach is time-consuming as tracking needs to be performed at different levels of the pyramid. In addition, we will show in the following that the pyramidal implementation of KLT can still lead to inaccurate results in highly complex urban driving environment.

In order to evaluate the accuracy of the conventional KLT in complex urban driving environment, an experiment is conducted based on the KITTI's stereo/flow benchmark [32], which provides 194 training images with ground truth flow fields and disparity maps. For each consecutive pair of images, up to 500 good features are extracted and tracked using OpenCV's implementation of KLT. The computed optical flows are compared with the ground truth. Fig. 4 illustrates the distribution of estimated flow error for all features in the order of descending corner measure for one of the image pairs. It can be observed that the conventional KLT results in high tracking error even for the features with high quality. The tracking error

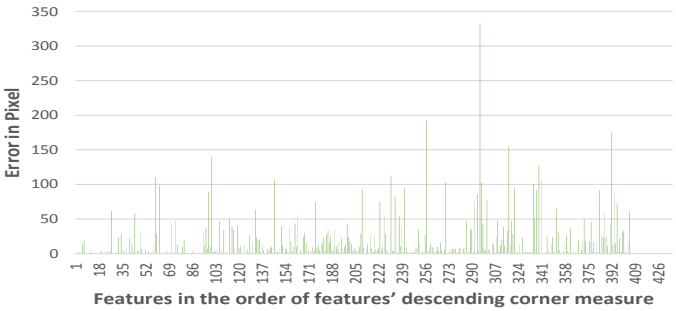


Fig. 4. Error distribution of optical flows estimated using conventional KLT algorithm.

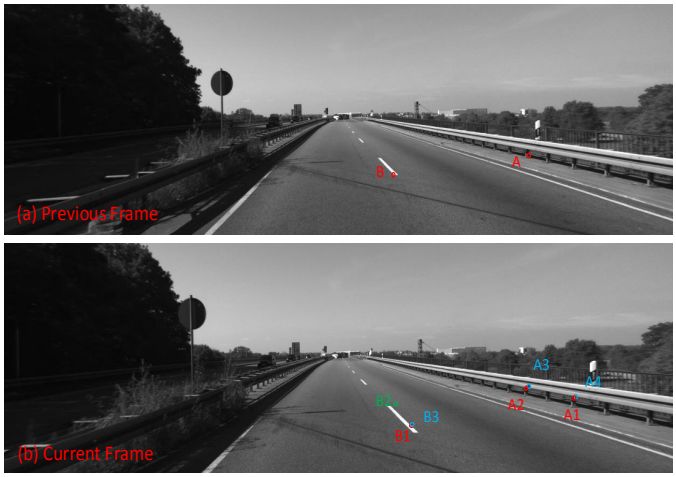


Fig. 5. An example of road scenario: Features that are detected in previous frame (a) are tracked in current frame (b).

becomes more prominent when the feature quality decreases. This indicates that the conventional KLT for feature tracking is highly susceptible to noise.

1) Improving Tracking Robustness using Smooth Motion Constraint: In the conventional KLT tracker, the optimization process as indicated in Eq. 16 starts the search of a feature in the current frame at its same position in the previous frame. This can easily lead to KLT tracking failure if the starting point is too far from the convergence region. Such cases are common in scenarios where ego-mtion is large or when the features are not distinctive enough from their surroundings. We will explain such a scenario with the help of Fig. 5. Assume that feature *A* has been detected in the previous frame. Its ground truth correspondence in the current image is *A1* but the conventional KLT tracker results in *A2*. The reason that the KLT tracker fails in detecting the correct correspondence is due to the fact that it starts the search for *A*'s correspondence at *A3* (the same position as *A* in the previous image). Since *A3* is closer to *A2* and the local patches around *A2* and *A* are largely similar, the KLT algorithm converges to *A2* and terminates the search. This demonstrates that the initial position for the correspondence search significantly affects the accuracy of KLT.

The authors in [38] also observed the importance of setting a proper starting point for KLT tracker. To ensure that the starting point falls as close as possible to the convergence point, the work in [38] relies on inertial sensor that is attached to the camera. However, this requires additional effort for sensor calibration and synchronization. Unlike [38], the proposed method determines a better starting point for KLT with the aid of the ego-motion estimated in previous step. In general, when the frame rate is high enough, a smooth motion pattern is presented between consecutive frames [22]. That is, the motion at time *n* is highly likely to be similar to the immediate previous motion at time *n-1*. Such phenomenon is referred to as Smooth Motion Constraint (SMC) [22].

Let $\mathbf{M}_{n-1}$ denotes the motion estimated from frame I_{n-2} to frame I_{n-1}. When frame I_n is available, by projecting the features detected in frame I_{n-1} to frame I_n using the previous motion $\mathbf{M}_{n-1}$, the projected location in frame I_n is highly likely to reside in the convergence region of KLT and therefore serves as a good starting point for KLT tracking process. We will describe this phenomenon again with the help of Fig. 5. By transforming *A* with the previously estimated motion and projecting it onto the current image, the new position locates in *A4*, which is much closer to the ground truth. Using *A4* as the starting point, KLT is able to adapt to the motion in the current frame and finally correctly converge to *A1*.

There exist works that utilize SMC for ego-motion computation in a different manner from the proposed method. For example, the work in [22] utilizes SMC to remove outliers that exhibit incoherent movement. The work in [15] utilizes SMC to generate an additional set of augmented features that try to complement the original features. The work in [41] utilizes camera motion to define a specific search region in the image for normalized cross-correlation based feature matching. In order to avoid the danger that the predicted search region misses the target, a computationally complex two-step projection operation together with uncertainty calculation is performed in [41]. Moreover, an additional step is required for re-localization in the presence of high uncertainty. Unlike existing works, the proposed method utilizes SMC to increase the chance of correct convergence for KLT tracking process

Algorithm 1 Pruning based Corner Detector

Input: Image I, feature quality threshold t
Output: A set of corners $\{\boldsymbol{m}_i\}$ in image I

/* Pruning */
Compute horizontal and vertical gradient image I_x, I_y;
Compute $|I_x|$, $|I_y|$ for each pixel in I;
Compute $a'c' = \sum |I_x| * |I_y|$ for each pixel in I;
Threshold $a'c'$ map with *threshold* = t * max($a'c'$) to obtain corner candidate set $\boldsymbol{K}$;
/* Corner Response Function */
for each pixel in $\boldsymbol{K}$ **do**
    Compute $a = \sum I_x^2$, $c = \sum I_y^2$, $b = \sum I_x I_y$;
    Compute corner response $C = \lambda_2$;
end for
Threshold $\boldsymbol{K}$ with *threshold* = t * max(C) to obtain candidate set $\boldsymbol{Q}$;
Sort $\boldsymbol{Q}$ in descending order of C;
Apply non-maximal suppression to obtain $\{\boldsymbol{m}_i\}$.

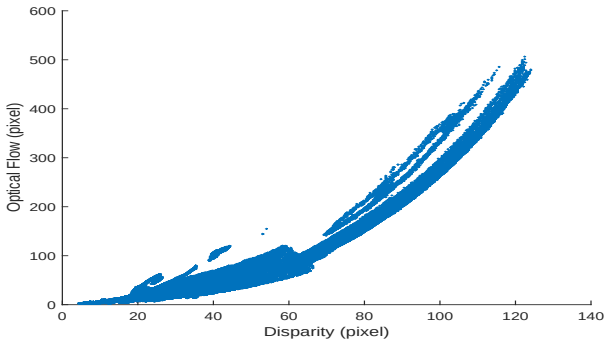


Fig. 6. Relationship between disparity and optical flow: pixels at a near distance from the camera are prone to large motion and pixels at a far distance are prone to small motion.

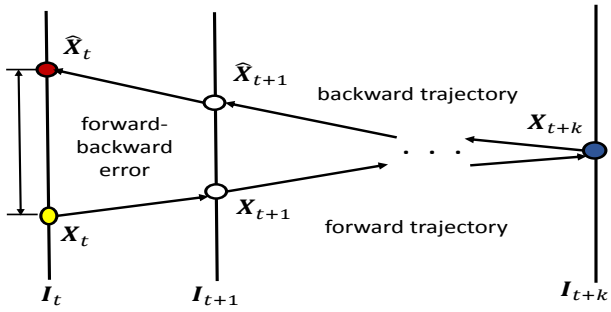


Fig. 7. Automatic tracking failure detection scheme. Figure from [42].

by determining a better starting point.

2) Improving Tracking Accuracy and Efficiency Using Adaptive Integration Window Technique: The size of the integration window for KLT tracker will affect not only the tracking accuracy but also the computational complexity. The conventional KLT tracker employs uniform window size and pyramid levels for all features. This easily violates the fact that a small window size is preferred to avoid smoothing out the details contained in the images while a large integration window is required to handle large motions.

In order to determine a suitable window size for KLT feature tracking, the relationship between the optical flow and the corresponding disparity field has been analyzed using the KITTI's flow/stereo benchmark. It can be observed from Fig. 6 that pixels at a near distance from the camera are prone to large motion and pixels at a far distance are prone to small motion. Inspired by this idea, an adaptive window size for the KLT can be employed based on the disparity information. For features in the near region, a larger window size or more pyramid levels are used. For features in the far region, a small window size or lesser pyramid levels are employed. This scheme helps to avoid the deployment of unnecessary large window size or pyramid levels for the features that undergo small motion, which will therefore improve the accuracy of the KLT tracker and also the compute efficiency.

In addition, large KLT window size or more pyramid levels which are typically required for features undergoing large motion can be further avoided by utilizing SMC. As shown in Fig. 5, given the feature *B* detected in the previous frame, its ground truth correspondence is *B1* in the current frame. The

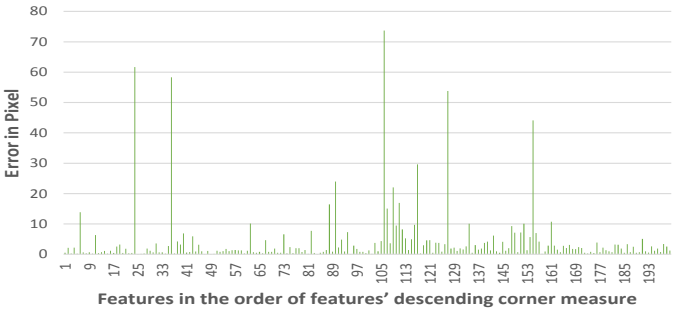


Fig. 8. Error distribution of optical flows estimated using KLT with automatic tracking failure detection.

initial search position employed by the conventional KLT is at *B2*, which is far from *B1*. As such, a large window size is needed to track feature *B* correctly. However, following the strategy proposed in previous sub-section to identify the initial point with the aid of SMC, the initial position to track feature *B* by the proposed method is set at *B3*. It can be observed that *B3* is close to *B1* and the required window size and pyramid levels can therefore be set to a smaller value.

3) Improving Tracking Robustness using Automatic Tracking Failure Detection Scheme: As pointed out in [22], tracking error is unavoidable. In order to identify such tracking failures, an automatic tracking failure detection scheme that is presented in [42] is adopted. As illustrated in Fig. 7, the basic idea is to check the forward-backward error during tracking. That is, forward and backward tracking is performed and the discrepancy between the starting point of the forward trajectory and the end-point of the backward trajectory is computed. If the forward-backward error is larger than some threshold, the corresponding feature pair is regarded as wrong setup and is therefore rejected. Fig. 8 shows the new distribution of estimated flow error after applying the automatic tracking failure detection scheme. It can be observed that the majority of features with high tracking error in Fig. 4 has been removed.

Based on the discussion above, we propose to improve the conventional KLT tracker as follows: 1) improve the tracking robustness by determining a better starting point for KLT tracking process with the aid of SMC; 2) improve tracking accuracy and efficiency by setting the integration window adaptively; 3) further improve the tracking robustness by enabling the automatic tracking failure detection scheme. The improved KLT feature tracking method is outlined in Algorithm 2.

C. Robust Gaussian-Newton based Motion Estimation with Early RANSAC Termination Condition

Given the set of feature correspondences, motion estimation computes the six motion parameters by solving the nonlinear least square problem as defined in Eq. 9. The Gaussian-Newton optimization algorithm is chosen to solve this residual function as it avoids computing the second derivatives.

Starting with an initial estimate $\boldsymbol{X}^0$, the Gaussian-Newton algorithm iteratively converges to a local minimum through Eq. 17, where $\boldsymbol{f}$ is the set of residual functions and $\boldsymbol{J_f} \in \mathbb{R}^{2k*6}$ represents the Jacobian matrix.

Algorithm 2 Improved KLT Tracker

Input: Consecutive images I_{n-1} and I_n;
Detected feature set $\{\boldsymbol{m}_{n-1}^i\}$ in frame I_{n-1};
Previous ego-motion $\boldsymbol{M}_{n-1}$;
Disparity maps D_{n-1} and D_n;
Calibrated camera parameters.

Output: Tracked feature correspondences $\{\boldsymbol{m}_n^i\}$ in I_n.

/* Forward Tracking */
for each feature $\boldsymbol{m}_{n-1}^i$ in frame i_{n-1} **do**
- Get its disparity value d_{n-1}^i;
- Set $\boldsymbol{q}_n^i$ as its initial estimate in frame I_n:
  $\boldsymbol{p}_{n-1}^i = g(\boldsymbol{m}_{n-1}^i)$;
  $\boldsymbol{p}_n^i = (\boldsymbol{M}_{n-1})^{-1} * \boldsymbol{p}_{n-1}^i$;
  $\boldsymbol{q}_n^i = h(\boldsymbol{p}_n^i)$;
- Perform KLT pyramid setup:
  Level = 1 (2 levels only);
  if $d_{n-1}^i < 10$ **then**
    integration window size r = 3;
  else if $d_{n-1}^i < 20$ **then**
    integration window size r = 5;
  else
    integration window size r = 7;
  end if
- Generate the new position of $\boldsymbol{m}_n^i$ by applying the KLT tracker.

end for
/* Backward Tracking */
for each feature $\boldsymbol{m}_n^i$ in frame I_n **do**
- Get its disparity value d_n^i;
- Set $\boldsymbol{q}_{n-1}^i$ as its initial estimate in frame I_{n-1}:
  $\boldsymbol{p}_n^i = g(\boldsymbol{m}_n^i)$;
  $\boldsymbol{p}_{n-1}^i = \boldsymbol{M}_{n-1} * \boldsymbol{p}_n^i$;
  $\boldsymbol{q}_{n-1}^i = h(\boldsymbol{p}_{n-1}^i)$;
- Perform KLT pyramid setup:
  Level = 1 (2 levels only);
  if $d_n^i < 10$ **then**
    integration window size r = 3;
  else if $d_n^i < 20$ **then**
    integration window size r = 5;
  else
    integration window size r = 7;
  end if
- Generate the new position of o_{n-1}^i by applying the KLT tracker.

end for
Reject feature correspondences where $dist(\boldsymbol{m}_{n-1}^i, \boldsymbol{o}_{n-1}^i) >$ 1 pixel.

$$\boldsymbol{X}^{s+1} = \boldsymbol{X}^s - (\boldsymbol{J}_{\mathbf{f}}^T * \mathbf{J}_{\mathbf{f}})^{-1} * \boldsymbol{J}_{\mathbf{f}}^T * \mathbf{f}(\boldsymbol{X}^s) \tag{17}$$

$$\boldsymbol{f}_i = \boldsymbol{w}_i(\boldsymbol{m}_n^i - h(\boldsymbol{R}_n * g(\boldsymbol{m}_{n-1}^i) + \boldsymbol{t}_n)), i = 1, 2, \ldots, k \tag{18}$$

$$(\boldsymbol{J}_f)_{ij} = \frac{\partial f_i(\boldsymbol{X}^s)}{\partial \boldsymbol{X}_j}, i = 1, 2, \ldots, 2k; j = 1, 2, \ldots, 6 \tag{19}$$

Eq. 17 is repeatedly computed until the residual ε in Eq. 20 is smaller than some predefined threshold.

$$\varepsilon = |\boldsymbol{X}^{s+1} - \boldsymbol{X}^s| = |(\boldsymbol{J}_{\mathbf{f}}^T * \boldsymbol{J}_{\mathbf{f}})^{-1} * \boldsymbol{J}_{\mathbf{f}}^T * \mathbf{f}(\boldsymbol{X}^s)| \tag{20}$$

The feature correspondences are usually contaminated with outliers. This is typically exhibited in feature correspondences extracted from moving objects such as pedestrians or vehicles. In addition, some feature points will be wrongly tracked. All of these noisy correspondences contribute to outliers and should be eliminated from the motion computation in order to increase the robustness of the estimated motion. In order to ensure robust estimation, the RANSAC algorithm [25] is adopted to identify outliers. The basic idea of RANSAC is to compute a fitting model from a set of samples selected randomly and check the number of points that are in consensus with the current estimated fitting model, i.e. inliers. This process is iteratively repeated until the maximum number of iterations has elapsed. Finally, the final model parameters are estimated using the largest set of inliers.

Instead of setting the maximum number of iterations manually, it has been pointed out in [25] that the number of iterations for RANSAC needed to achieve a desired accuracy requirement can be theoretically derived as shown below:

$$RANSAC_{iter} = \frac{\log(1-p)}{\log(1-w^n)} \tag{21}$$

Where n is the number of minimum points needed for estimating a model, w is the percentage of inliers in the data points, p is the requested probability of success. Due to the formulation equation adopted in Eq. 9, at least three points are needed for estimating a model, therefore $n = 3$. It can be observed from Eq. 21 that $RANSAC_{iter}$ is dynamically determined based on the number of inliers found in current iteration. This means that once a set of inliers that are large enough are identified, there is no need to continue repeating the RANSAC sampling operation anymore. We refer to this phenomenon as *Early RANSAC Termination Condition (ERTC)*. The proposed method has employed strategies to increase the accuracy of the extracted feature correspondences in previous sections. With the accurate feature correspondences provided from Stage 1, the Gaussian-Newton optimization with *ERTC* enabled is able to converge faster.

Based on the above discussion, given the set of correspondences $\{\boldsymbol{m}_{n-1}^i\}$ and $\{\boldsymbol{m}_n^i\}$, the method proposed for motion estimation is given in Algorithm 3 and Algorithm 4.

V. Evaluation

In this section, the proposed method will be thoroughly evaluated using a large scale benchmark. We will first describe the benchmark, evaluation criteria and baseline algorithms that are adopted in this experiment. We will then evaluate the proposed algorithm in terms of accuracy and computation time by comparing the proposed algorithm with the state-of-art baseline algorithms.

Algorithm 3 Motion Estimation

Input: Feature correspondences $\{\boldsymbol{m}_{n-1}^i\}$ and $\{\boldsymbol{m}_n^i\}$;
Successful probability p;
Output: $\boldsymbol{X} = (t_x, t_y, t_z, r_x, r_y, r_z)$
/* *Initialization* */
Transform features from 2D to 3D via triangulation:
$\boldsymbol{p}_{n-1}^i = g(\boldsymbol{m}_{n-1}^i)$ for each $i = 1, 2, \ldots, k$;
largest inlier set $\boldsymbol{\alpha} = \emptyset$;
$RANSAC_{iter} = 50$;
$trial_{count} = 0$;
/* *RANSAC Iterative Refinement* */
while $trial_{count} < RANSAC_{iter}$ **do**
  $\{\boldsymbol{C}_i\} \leftarrow$ 3 correspondences selected randomly;
  $\boldsymbol{X} = GNO(\{\boldsymbol{C}_i\})$;
  Calculate current inlier set $\boldsymbol{in_{curr}}$ based on $\boldsymbol{X}$;
  if $\boldsymbol{in_{curr}}.size > \boldsymbol{\alpha}.size$ **then**
    $\boldsymbol{\alpha} = \boldsymbol{in_{curr}}$;
  end if
  Update $RANSAC_{iter}$ based on Eq. 21;
  $(trial_{count})$++;
end while
$\boldsymbol{X}$ = GNO($\boldsymbol{\alpha}$)

Algorithm 4 Gaussian-Newton Optimization Method (GNO)

Input: Feature correspondences $\{\boldsymbol{C}_i\}$;
Successful probability p;
Maximum Gaussian Newton iteration GN_{max};
Residual threshold $\boldsymbol{t_{res}}$;
Output: $X = (t_x, t_y, t_z, r_x, r_y, r_z)$.
/* *Initialization* */
$\boldsymbol{X}^0 = 0$;
$s = 0$;
/* *Start Gaussian Newton Minimization Circle* */
while not converged and $s < GN_{max}$ **do**
  s = s + 1;
  Calculate $\boldsymbol{J_f}$ at $\boldsymbol{X}^{s-1}$;
  $\boldsymbol{X}^s = \boldsymbol{X}^{s-1} - (\boldsymbol{J}_\mathbf{f}^T * \boldsymbol{J}_\mathbf{f})^{-1} * \boldsymbol{J}_\mathbf{f}^T * \mathbf{f}(\boldsymbol{X}^{s-1})$;
  $\varepsilon = \boldsymbol{X}^s - \boldsymbol{X}^{s-1}$;
  if $|\varepsilon| < \boldsymbol{t_{res}}$ **then**
    Successfully converged;
  end if
end while
$\boldsymbol{X} = \boldsymbol{X}^s$.

A. Experimental Setup

Experiments are conducted based on the widely known KITTI odometry evaluation platform [32]. The KITTI's odometry benchmark consists of 22 stereo *1344*391* sequences, where the first 11 sequences *(00-10)* are provided with ground truth trajectories for training and the remaining 11 sequences do not have ground truth. These 22 sequences were collected from stereo cameras installed in a vehicle that was driven around Karlsruhe, Germany. This benchmark covers a variety of road scenarios and provides a very challenging test-bed. Some samples of the benchmark are illustrated in Fig. 9.

The evaluation criteria suggested by KITTI [32] is adopted, that is, **translational** and **rotational errors** for all possible subsequences of length $(100, ..., 800)$ meters. Translational errors are measured in percentage while rotational errors are measured in degrees per meter. The average of these errors are used to compare the performance of various approaches in the KITTI evaluation platform.

1) Baseline Algorithms: Many works have been submitted to the KITTI platform for evaluation. It can be observed that the existing solutions in KITTI evaluation platform fail to achieve a good balance of high accuracy and low computational complexity [32]. For example, [15] outperforms all other visual odometry methods in terms of translational and rotational accuracy till 2015, but it is time consuming. On the contrary, the work in [10] is able to achieve a real-time performance, but suffers from low accuracy. In the following, we will denote the work from [15] as *MFI* and the work from [10] as *VISO2-S*.

MFI: In order to reduce the motion drift caused by accumulation of feature tracking errors from frame to frame, *MFI* uses the whole history of the tracked feature points to compute the ego-motion. In their technique, the key idea is to integrate the features measured and tracked over all past frames into a single and improved estimate. An augmented feature set, obtained by the sample mean of all previous measured features which are transformed into the current frame, is added to the optimization formula. The importance of each feature is weighted in terms of their life age.

VISO2-S: In order to reduce the computational complexity, *VISO2-S* adopts a much simpler feature detector and descriptor. Feature locations are found by extracting the maximum or minimum Sobel filter response. In addition, instead of using the compute-intensive rotation and scale invariant feature descriptors like SURF or SIFT, features are described by concatenating the response over a sparse set of 16 locations within the 11*11 block and matched based on the sum of absolute differences (SAD) dissimilarity metric. Finally, the inputs to the visual odometry algorithm are features matched between four images, namely the left and right images of two consecutive frames. Given these 'circular' feature correspondences, the camera motion is computed by minimizing the sum of re-projection errors using the Gaussian-Newton optimization.

Fig. 9. Some samples of the benchmark.

TABLE I
OVERVIEW OF THE BASELINE AND PROPOSED ALGORITHMS

	Feature Correspondence Setup		Motion Estimation	
	Feature Detection	Feature Tracking	Optimization Scheme	Robust Estimation
MFI	Harris+FREAK	Feature matching by brute-force combinatorial search	Newton method based minimization of re-projection residual in left image space	Iteratively reject outliers whose re-projection residual is larger than some threshold
VISO2-S	Minima and maxima of blob and corner filter responses + concatenation of sobel filter responses based descriptor	SAD dissimilarity metric based two-passes circle feature matching	Gaussian-Newton method based minimization of re-projection residual in both of left and right image space + Kalman Filter Refinement	RANSAC
ORG-KLT	Conventional KLT corner detector	conventional KLT tracker with automatic tracking failure detection ability	Gaussian-Newton method based minimization of re-projection residual in left image space	RANSAC with early termination condition
Proposed	KLT corner detector with pruning	Improved KLT tracker	Gaussian-Newton method based minimization of re-projection residual in left image space	RANSAC with early termination condition

ORG-KLT: The proposed work is capable of rapidly extracting a set of accurate feature correspondences by improving the KLT feature tracker. In order to illustrate this improvement, the proposed algorithm will also be compared to the visual odometry algorithm that uses conventional KLT with automatic tracking failure detection ability and the motion estimation method proposed in Algorithm 3. This baseline algorithm is denoted as *ORG-KLT*.

The differences between the baseline algorithms and the proposed algorithm have been highlighted in Table I.

2) Implementation Details: For the proposed and *ORG-KLT* algorithm, up to 500 features are extracted for each frame and tracked in the consecutive frame. The required disparity information for features are provided by the OpenCV implementation of the Semi-Global Matching algorithm [43]. We use the default parameter settings in OpenCV and do not enable the multi-thresholding programming functionality inside OpenCV. Both of the proposed algorithm and *ORG-KLT* are implemented in C++ on a CPU @ 3.5 GHz machine. It is noteworthy that unlike the implementation of *MFI*, we currently do not employ any code optimization technique like multi-threshold programming or GPU programming. For *MFI* and *VISO2-S*, we directly use the experimental figures reported in their papers.

B. Accuracy Evaluation

First, an extensive quantitative evaluation between *ORG-KLT* and the proposed algorithm is conducted based on the 11 training sequences. Fig. 10 and Fig. 11 show the ground-truth and estimated vehicle's trajectories from *ORG-KLT* and the proposed algorithm for these 11 training sequences. This provides an intuitive way to visualize the evaluation results. It can be observed that the estimated trajectories from the proposed algorithm are closer to the ground truth than the ones from *ORG-KLT*.

In particular, interesting phenomenons can be observed from Fig. 10(b) corresponding to *Sequence 01*, where there exists a lot of challenging scenarios in the mid trajectory segments as discussed in Section IV.B. Firstly, it can be observed that the reconstructed paths from both of the *ORG-KLT* and the proposed algorithm deviate from the ground truth at Position *A* and persist for certain amount of time. At Position *B*, the reconstructed path from *ORG-KLT* deviates again. This means that the proposed method is more robust than *ORG-KLT* in dealing with challenging environment. Secondly, although the proposed method fails at position *A*, the reconstructed path from the proposed algorithm shows the same shape as the ground-truth at the end. This means that the proposed algorithm is able to automatically recover from wrong previous motion estimation when the scene is not challenging. The reason that the proposed algorithm is capable of recovering from wrong motion estimation in scenarios where the scene is not challenging is due to the fact that the utilization of SMC in the proposed method aims to increase the chance that the starting search point for KLT falls within the convergence region, thereby enabling the KLT tracker to more likely converge to the true global minimum. However, as pointed out by [38], KLT tracking process is tolerant to an initial parameter error as long as the initial point falls within the convergence region. Therefore, if the initial position guided by wrong motion still falls in the convergence region, KLT is still able to converge correctly and the proposed method is therefore able to recover from wrong previous motion estimation.

In addition, the average translational and rotational errors relative to the ground truth for both of the proposed algorithm and *ORG-KLT* for the 11 training sequences are presented in Fig. 12. On average, the translation and rotation errors for *ORG-KLT* and the proposed algorithm for the 11 training sequences are (1.3974%, 0.0061[deg/m]) and (0.9768%, 0.0056[deg/m]). Therefore, the proposed algorithm is 30% better than *ORG-KLT*.

We have also submitted the results of the proposed method for the 11 test sequences to the KITTI odometry evaluation platform. The average translation and rotation errors relative to the ground truth for proposed algorithm, *MFI* and *VISO2-S* for the 11 test sequences are presented in Fig. 13. In addition, the corresponding estimated trajectories from proposed algorithm, *MFI* and *VISO2-S* over sequences 11-15 (the website only provide the computed trajectories relative to the ground-truth for sequences 11-15) are depicted in Fig. 14. On average, the translational and rotational errors for *MFI, VISO2-S* and the proposed algorithm for the 11 testing sequences are

(1.30%, 0.0030[deg/m]), (2.44%, 0.0114[deg/m]) and (1.26%, 0.0038[deg/m]) respectively. It can be observed that the proposed algorithm performs approximately 3% better than *MFI* and 48% better than *VISO2-S*. At the time of submission into the KITTI odometry evaluation platform, the proposed method was ranked in the 8^{th} place in terms of accuracy, while *MFI* and *VISO2-S* were ranked in the 10^{th} and 38^{th} places respectively. Currently, the proposed method is ranked in 14^{th} place in terms of accuracy, while *MFI* and *VISO2-S* are ranked in the 15^{th} and 37^{th} places respectively.

C. Computation Time Evaluation

Table II shows the computation time for the proposed algorithm and all the three baseline algorithms. In the current implementation, the dense disparity map is directly provided for the proposed algorithm and *ORG-KLT*. For a fair comparison, the computation time for disparity computation is not included for all the four algorithms. The proposed algorithm is 28% faster than *ORG-KLT*. It can be observed that both of the two stages in the proposed method contribute to runtime performance gain. This is due to the low computational complexity strategies adopted in the feature correspondence setup. In addition, since robust techniques during the KLT tracking process and the RANSAC with early termination rule are employed, the set of accurate feature correspondences allows the Gaussian-Newton process to converge faster.

MFI is able to reduce pose error compared to their earlier work [22], [23], however this is achieved at the expense of huge computational complexity. Up to 4,096 features are tracked between consecutive frames. The key-points are matched between consecutive frames by brute-force combinatorial search. The computation time reported by *MFI* is only possible after they enable the multi-thresholding programming technology OpenMP and intense code optimization using the Intel Performance Primitives library on 2.7 GHz CPU with 4 cores. Finally, *VISO2-S* achieves a short computation time at the price of reduced estimation accuracy.

The proposed method also exhibits lower computational complexity when compared to other methods that are recently submitted to the KITTI evaluation platform. For example, the computation time for the top three visual odometry methods in the KITTI platform (at the time this paper is submitted) are between 0.1 to 0.3 seconds/frame on 2.0Ghz or 2.5 Ghz platforms with dual cores. The computation time for the proposed algorithm is 0.03 seconds/frame on a 3.5GHz platform (one core) without utilizing any code optimization technique (e.g. multi-threshold programming or GPU programming). We are confident that the computation time of the proposed method will further reduce if such code optimization techniques are enabled.

TABLE II
COMPUTATION TIME COMPARISON

	Feature Correspondence Setup	Motion Estimation	Total	Platform
MFI	37.1 ms	8.8 ms	45.9 ms	2.7GHz (4 Cores)
VISO2-S	36.6 ms	4.3 ms	40.9 ms	2.5GHz (1 Core)
ORG-KLT	40.8 ms	1.1 ms	41.9 ms	3.5GHz (1 Core)
Proposed	29.5 ms	0.8 ms	30.3 ms	3.5GHz (1 Core)

VI. CONCLUSION

It has been shown that the proposed method for estimating the ego-motion of vehicle overcomes the limitations of existing solutions by integrating runtime-efficient strategies with robust techniques at various core stages in visual odometry. A novel pruning technique is adopted to notably reduce the computational complexity of detecting corner features without compromising on the quality of the extracted corner features. A robust and compute-efficient KLT tracker is proposed to facilitate the generation of the feature correspondences in a robust and runtime efficient way. The accuracy of extracted feature correspondences is improved by leveraging on ego-motion prior to determine a better initial point for fast and accurate feature convergence during tracking and incorporating an automatic tracking failure detection scheme to exclude the feature correspondences with large tracking error. In addition, the computational complexity of the conventional KLT has been improved by setting the integration window size adaptively. With the accurate feature correspondences provided, Gaussian-Newton optimization scheme supported by an early RANSAC termination condition is shown to converge faster in the motion estimation process. The above contributions are integrated into a framework for fast and robust visual odometry. The experimental results based on a widely used evaluation platform clearly demonstrate the advantages of the proposed framework over existing state-of-the-art solutions for robust and runtime-efficient visual odometry.

REFERENCES

[1] C. Rabe, U. Franke, and S. Gehrig, "Fast detection of moving objects in complex scenarios," in *2007 IEEE Intelligent Vehicles Symposium*. IEEE, 2007, pp. 398–403.

[2] Y. Watanabe, P. Fabiani, and G. Le Besnerais, "Simultaneous visual target tracking and navigation in a gps-denied environment," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*. IEEE, 2009, pp. 1–6.

[3] D. Scaramuzza, "1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *International journal of computer vision*, vol. 95, no. 1, pp. 74–85, 2011.

[4] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. IEEE, 2004, pp. I–652.

[5] D. Scaramuzza and F. Fraundorfer, "Visual odometry: Part i: The first 30 years and fundamentals," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[6] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part ii: Matching, robustness, optimization, and applications," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[7] C. Harris and M. Stephens, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15. Citeseer, 1988, p. 50.

[8] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[9] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[10] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 963–968.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[13] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*. Ieee, 2012, pp. 510–517.

[14] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.

[15] H. Badino, A. Yamamoto, and T. Kanade, "Visual odometry by multi-frame feature integration," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 222–229.

[16] G. P. Stein, O. Mano, and A. Shashua, "A robust method for computing vehicle ego-motion," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 362–368.

[17] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[18] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 2531–2538.

[19] M. J. Milford and G. F. Wyeth, "Single camera vision-only slam on a suburban road network," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 3684–3689.

[20] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, "Rover navigation using stereo ego-motion," *Robotics and Autonomous Systems*, vol. 43, no. 4, pp. 215–229, 2003.

[21] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme." in *Intelligent Vehicles Symposium*, 2010, pp. 486–492.

[22] H. Badino, "A robust approach for ego-motion estimation using a mobile stereo platform," in *Complex Motion*. Springer, 2007, pp. 198–208.

[23] H. Badino and T. Kanade, "A head-wearable short-baseline stereo system for the simultaneous estimation of structure and motion." in *MVA*, 2011, pp. 185–189.

[24] P. H. Torr and D. W. Murray, "The development and comparison of robust methods for estimating the fundamental matrix," *International journal of computer vision*, vol. 24, no. 3, pp. 271–300, 1997.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[26] D. Nistér, "Preemptive ransac for live structure and motion estimation," *Machine Vision and Applications*, vol. 16, no. 5, pp. 321–329, 2005.

[27] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell *et al.*, "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

[28] J.-P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz, "A new approach to vision-aided inertial navigation," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4161–4168.

[29] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," in *Robotics research*. Springer, 2010, pp. 201–212.

[30] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *The International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011.

[31] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustmenta modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[33] G. Sibley, L. Matthies, and G. Sukhatme, "Bias reduction and filter convergence for long range stereo," in *Robotics Research*. Springer, 2007, pp. 285–294.

[34] U. Franke, C. Rabe, H. Badino, and S. Gehrig, "6d-vision: Fusion of stereo and motion for robust environment perception," in *Joint Pattern Recognition Symposium*. Springer, 2005, pp. 216–223.

[35] F. Erbs, B. Schwarz, and U. Franke, "From stixels to objectsa conditional random field based approach," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 586–591.

[36] M. Muffert, D. Pfeiffer, and U. Franke, "A stereo-vision based object tracking approach at roundabouts," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 2, pp. 22–32, 2013.

[37] A. Barth and U. Franke, "Estimating the driving state of oncoming vehicles from a moving platform using stereo vision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 560–571, 2009.

[38] M. Hwangbo, J.-S. Kim, and T. Kanade, "Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and gpu implementation," *The International Journal of Robotics Research*, p. 0278364911416391, 2011.

[39] M. Wu, N. Ramakrishnan, S.-K. Lam, and T. Srikanthan, "Low-complexity pruning for accelerating corner detection," in *2012 IEEE International Symposium on Circuits and Systems*. IEEE, 2012, pp. 1684–1687.

[40] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[41] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[42] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Pattern recognition (ICPR), 2010 20th international conference on*. IEEE, 2010, pp. 2756–2759.

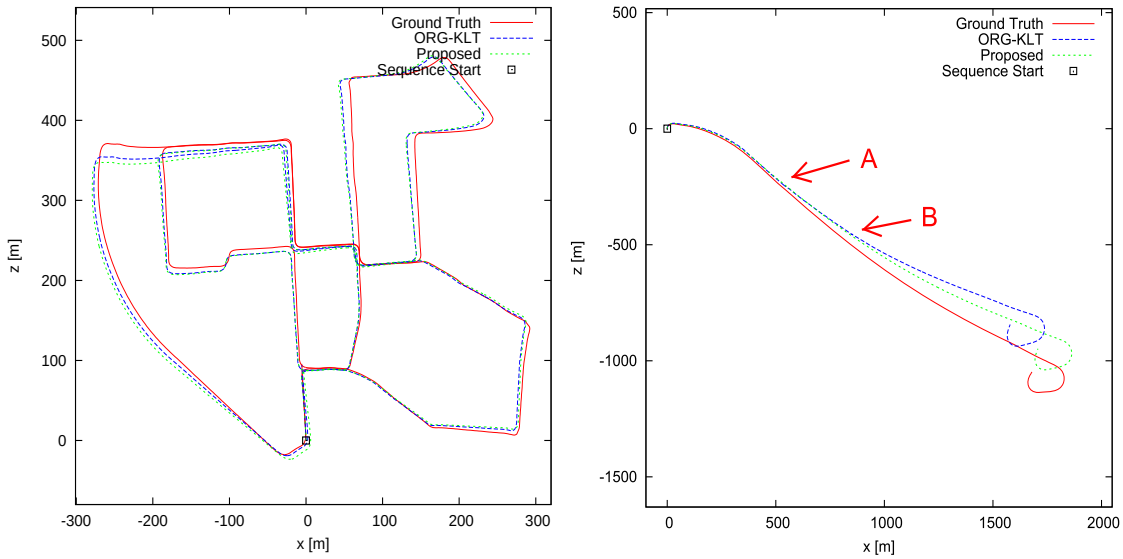
[43] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

Meiqing Wu received M.S. degree in Computer Engineering from Peking University, China in 2009. She is currently working toward the Ph.D. degree in the School of Computer Engineering in Nanyang Technological University, Singapore.

Her current research interests include stereo vision, motion analysis, object detection and tracking for urban traffic scene understanding.

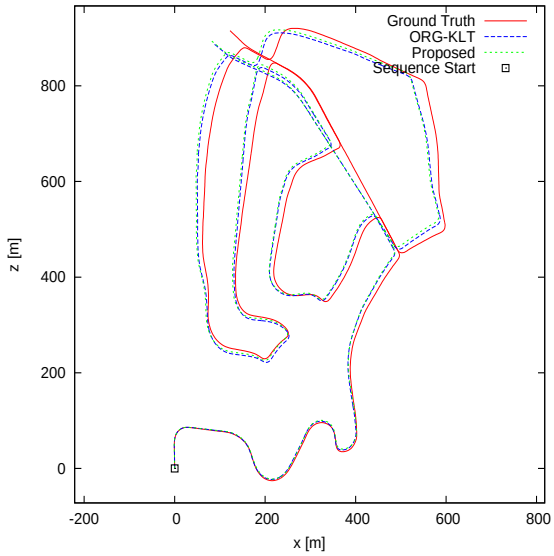
Siew-Kei Lam (M03) received his BASc, MEng and PhD from Nanyang Technological University (NTU), Singapore. He is currently an Assistant Professor in School of Computer Engineering (SCE), NTU and his research investigates methods for realizing custom computing solutions in embedded systems. He has published over 75 international refereed journals and conferences in design methodologies for heterogeneous and reconfigurable systems, embedded vision and autonomous systems, and high-speed computer arithmetic.

Thambipillai Srikanthan (SM92) joined Nanyang Technological University (NTU), Singapore in 1991. At present, he holds a full professor and joint appointments as Director of a 100 strong Centre for High Performance Embedded Systems (CHiPES) and Director of the Intelligent Devices and Systems (IDeAS) cluster. He founded CHiPES in 1998 and elevated it to a university level research centre in February 2000. He has published more than 250 technical papers. His research interests include design methodologies for complex embedded systems, architectural translations of compute intensive algorithms, computer arithmetic, and high-speed techniques for image processing and dynamic routing.

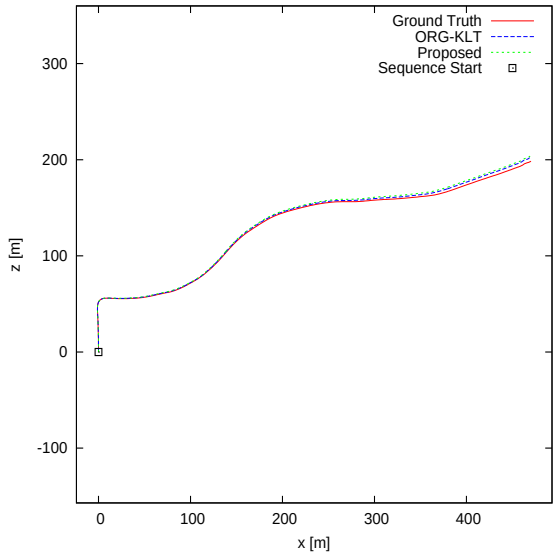


(a) Sequence 00

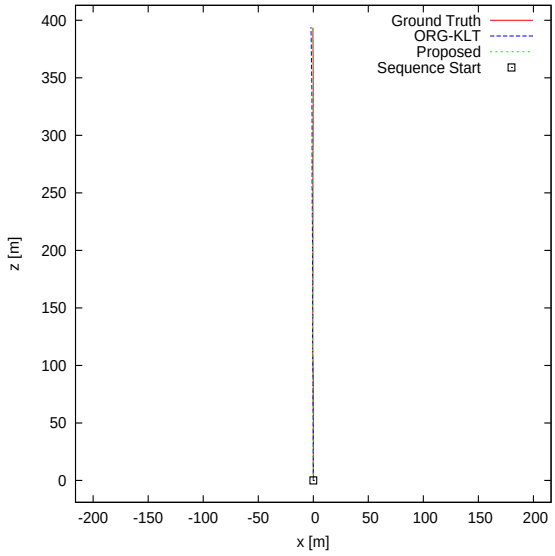
(b) Sequence 01



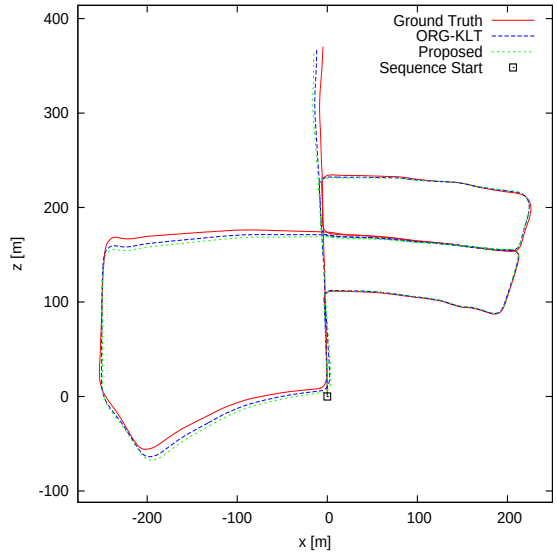
(c) Sequence 02



(d) Sequence 03

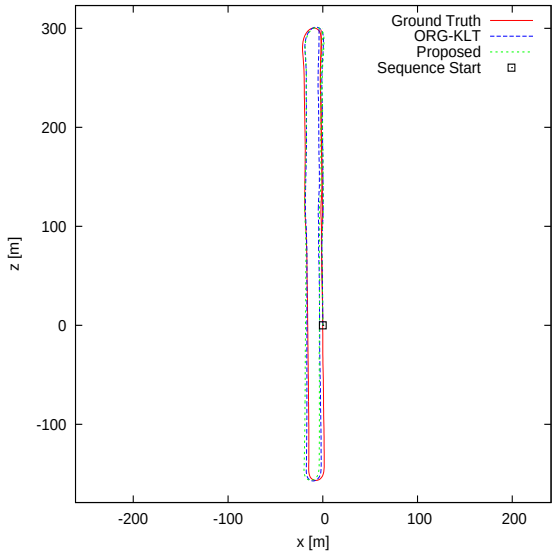


(e) Sequence 04

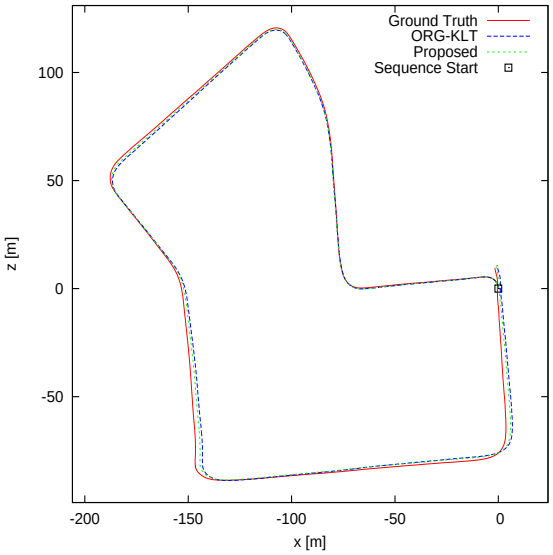


(f) Sequence 05

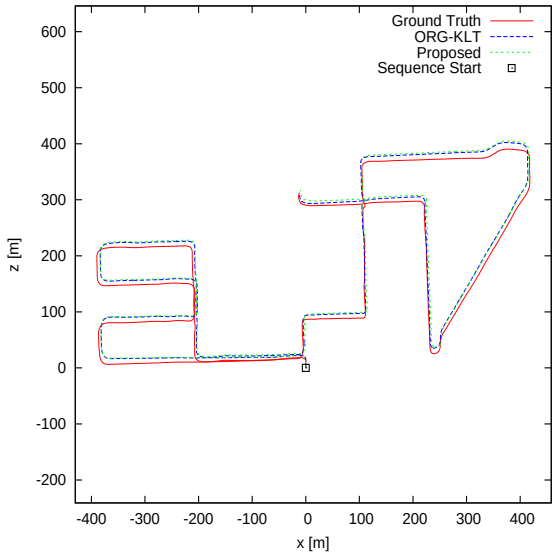
Fig. 10. Reconstruction of paths from *ORG-KLT* and *proposed* algorithm for Sequences 00-05.



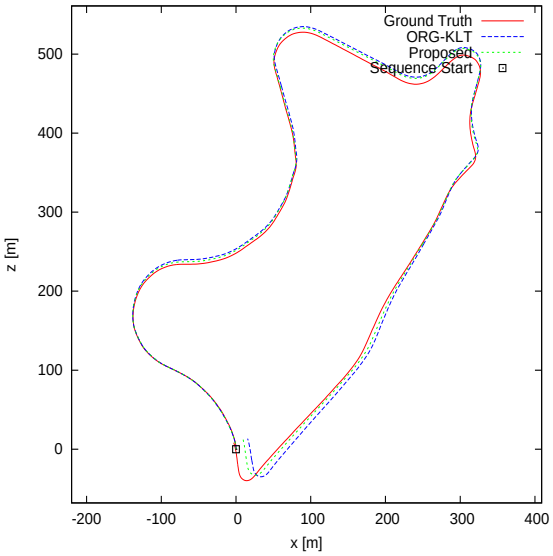
(a) Sequence 06



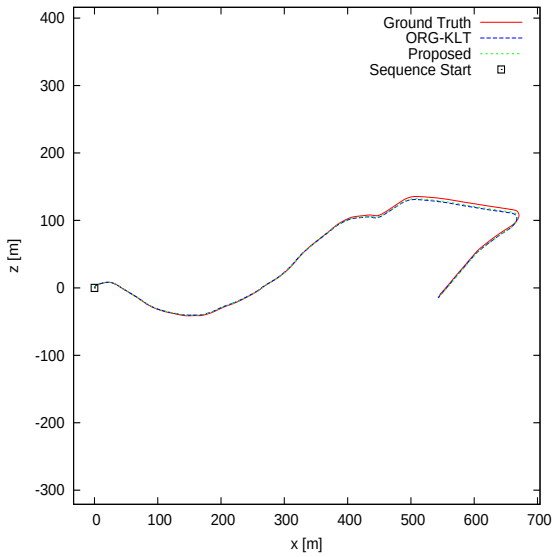
(b) Sequence 07



(c) Sequence 08



(d) Sequence 09



(e) Sequence 10

Fig. 11. Reconstruction of paths from *ORG-KLT* and *proposed* algorithm for Sequences 00-05.

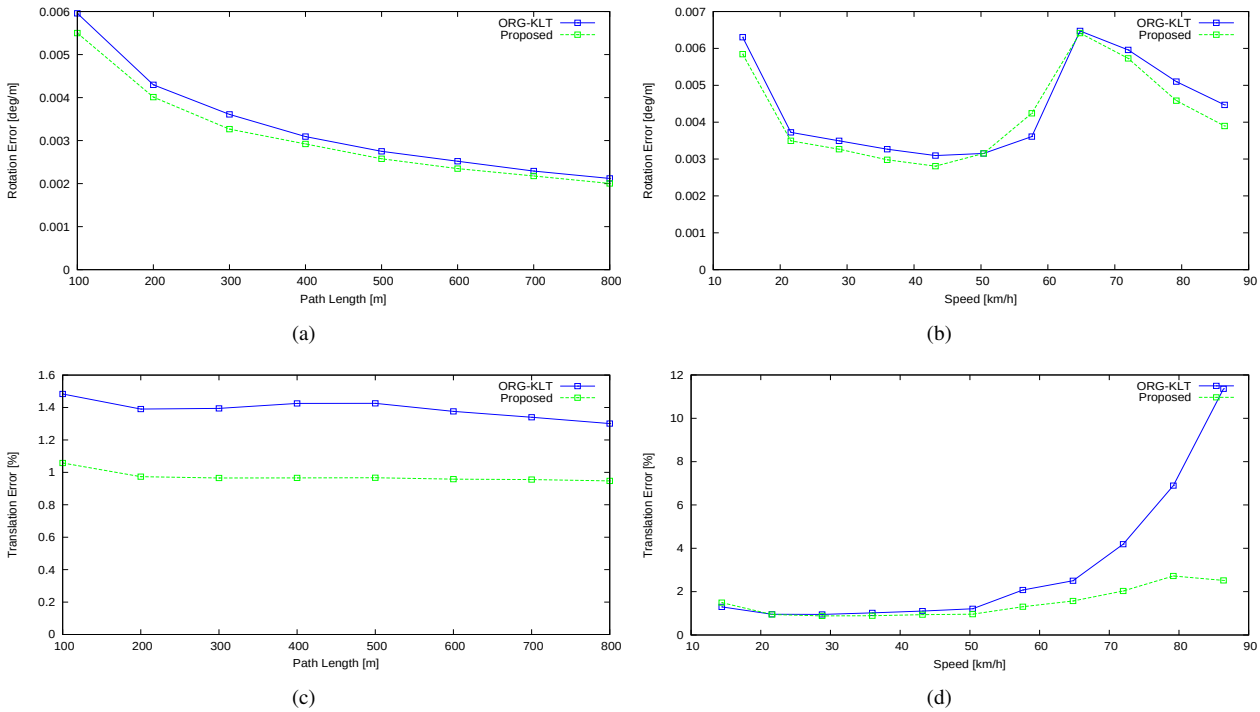


Fig. 12. Average translational and rotational error for *ORG-KLT* and proposed algorithm over sequences 00-10. The proposed algorithm is 30% better than *ORG-KLT*.

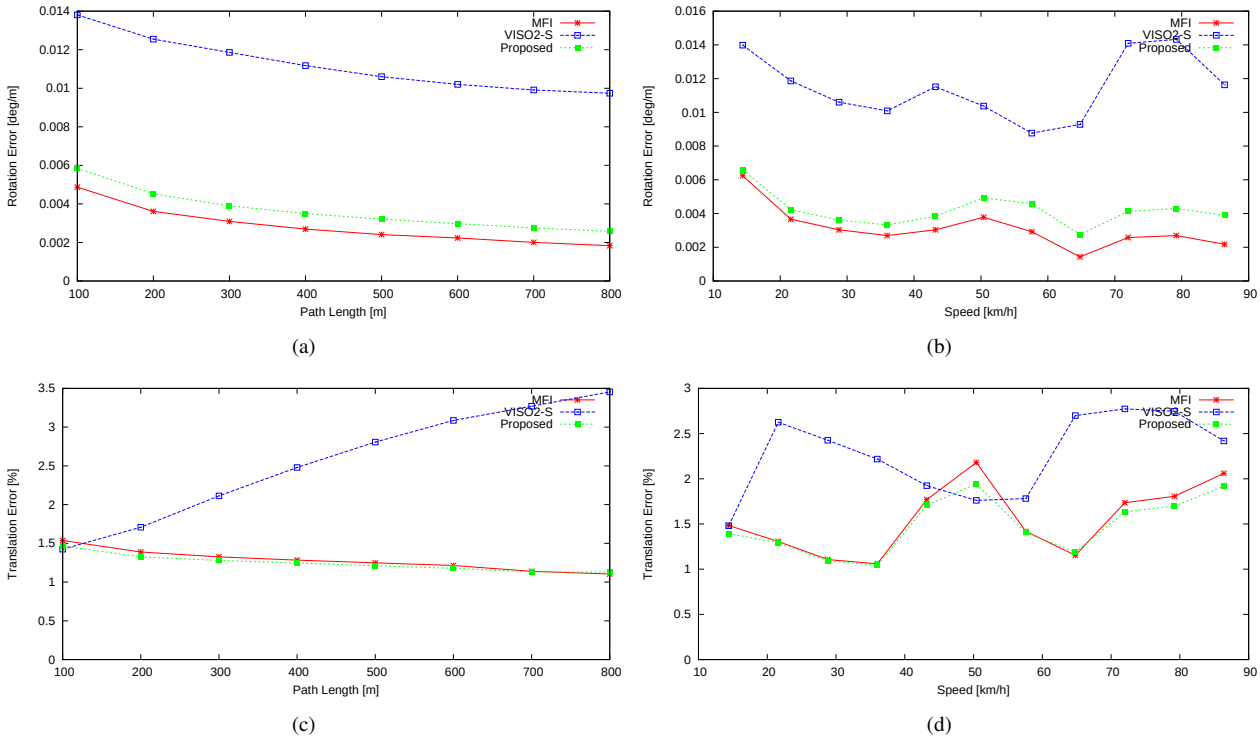
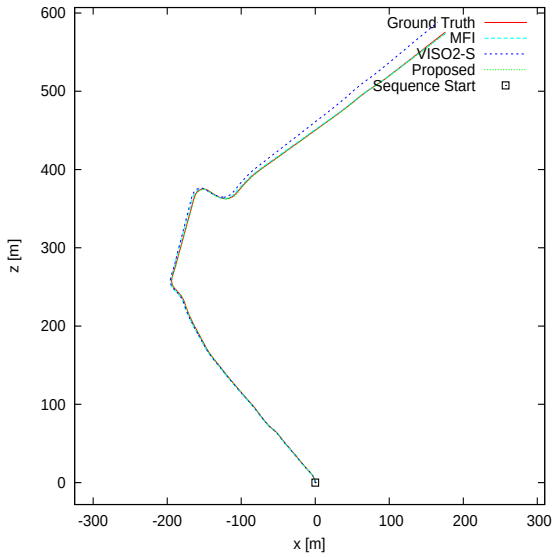
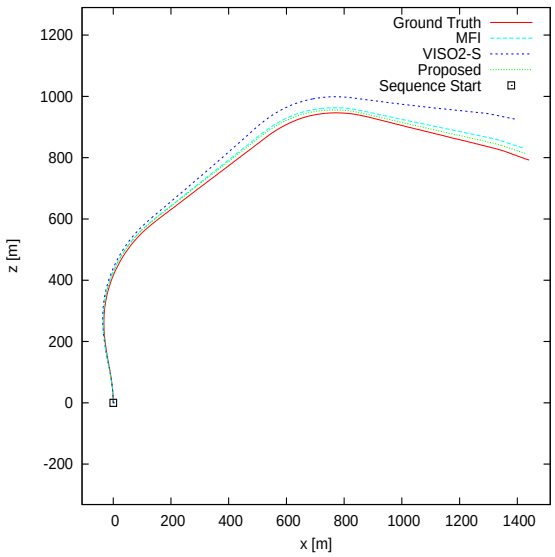


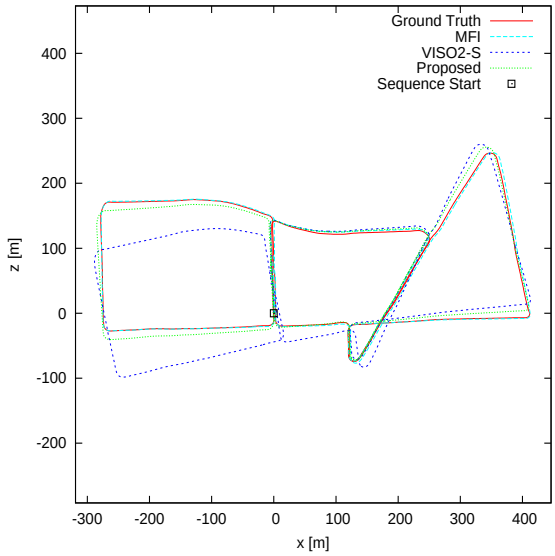
Fig. 13. Average translational and rotational error for *MFI*, *VISO2-S* and *proposed* algorithm over Sequences 11-21. The proposed algorithm performs 3% better than *MFI* and 48% better than *VISO2-S*.



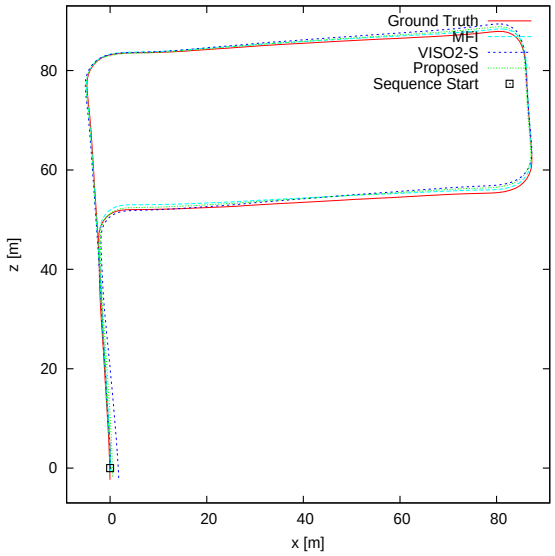
(a) Sequence 11



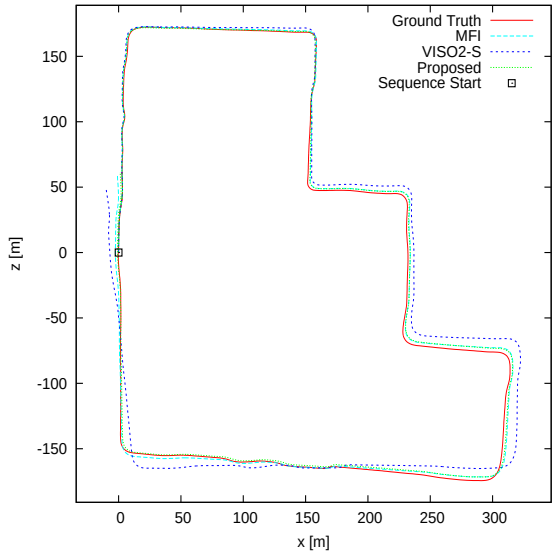
(b) Sequence 12



(c) Sequence 13



(d) Sequence 14



(e) Sequence 15

Fig. 14. Reconstruction of paths from *MFI*, *VISO2-S* and *proposed* algorithm for Sequences 11-15.