Stereo Visual Odometry on The KITTI Datasets

This **Course Project Report** was prepared by M.T. and one of his classmates for the AER1513 course.

Abstract—**In this study, we have implemented a classic feature based stereo visual odometry (VO) pipeline to gain experience on processing stereo images and estimating vehicle egomotion solely based on vision input. In our VO algorithm, SURF image features are extracted and matched by using the sum of squared difference similarity metric to generate both stereo matches and between-frame feature correspondences. A 3D-to-3D point cloud alignment problem is then set up to estimate the relative camera pose between two constitutive timesteps, and the full trajectory of the vehicle motion is finally produced by compounding the incremental pose changes. The performance of our VO implementation is evaluated on datasets extracted from the publicly available database: KITTI Vision Benchmark Suite.**

Supplementary Material

Results of our VO implementation on several KITTI datasets are available at: https://youtu.be/gEiOCKHEKbY

I. Introduction

Visual odometry (VO) is the process of estimating the incremental motion of an agent (e.g. vehicle, robot) by using the image information captured by one or multiple cameras mounted on the agent's body [17]. VO has been a well studied topic in computer vision and robotics since the 1980s due to its versatility and practicality. In the simplest setting, VO does not require sensors other than cameras, which are inexpensive, widely accessible and low power consumption [7]. Therefore, VO has been applied in a wide range of applications including automobile, underwater and space robotics, healthcare, wearable computing, industrial manufacturing, and gaming & virtual reality [15]. Particularly in robotics and automobiles, VO plays a crucial role in vehicle navigation, driver assistance, and control & guidance of unmanned vehicle [15].

The idea to estimate the egomotion of a robot solely based on image sequences was initially pioneered by Moravec (1980) [11] and Matthies (1989) [10], and the term Visual Odometry was first coined by Nistér et al. in 2004 [13]. Since then, many different kinds of VO algorithms have been developed and applied in studies to estimate the 6 degree-of-freedom (DOF) poses of a moving camera. In the early studies of VO, salient and repeatable image features were first detected on the images, and the 3D positions of the features were measured by triangulating them into a 3D coordinate system. The motion of the robot was then computed as a rigid body transformation to align the triangulated 3D points seen at two consecutive robot positions [17]. This kind of VO algorithms can be categorized as a feature-based method because image features are explicitly extracted and matched (or tracked) across images. Another category of VO algorithms is called appearance-based method, where the intensity information of image pixels from either the whole image or several regions of the image are utilized to estimate egomotion (e.g. [16]). Following the same classification scheme, a third category of VO algorithms, so called hybrid algorithms, is developed to take the advantages of both feature-based and appearance-based methods (e.g. [4]).

In another classification scheme, VO algorithms can be characterized according to the type of camera(s) employed by the system. In the early years of VO development, most of the studies employed stereo camera systems to capture images, and thus, the algorithms that use stereo images input are classified as stereo VO algorithms. In contrast, monocular VO algorithms employ only a single camera (either perspective or omnidirectional) to perform pose estimation (e.g. [16]). However, the motion estimated by a monocular VO algorithm can only be recovered up to a scale factor because only bearing information is available. Further scale information (e.g. measured by a range sensor) would be required in order to recover the absolute scale of the vehicle motion [17].

In this study, a feature-based stereo VO pipeline, shown in Fig. 1, is implemented by the team to gain experience on processing stereo images and performing state estimation using egomotion. The state to be estimated is the 6 DOF poses of the stereo camera mounted on a moving vehicle expressed in an inertial frame. The implementation is a frame-to-frame VO operated in an off-line fashion, where no time constraint is considered. The datasets used are extracted from the publicly available KITTI Vision Benchmark Suite [6], and only the synced and rectified grey-scale stereo images are used. The rest of this report is organized as follows: after the introduction, a literature review is presented in Section II. Section III provides the methodology and the implementation details of the VO pipeline, followed by Section IV in which experimental results are presented and discussed. Lessons learned and future work are included in Section V.

II. Literature Review

Although there exist a large number of VO algorithms proposed in literature, a general pipeline for feature-based

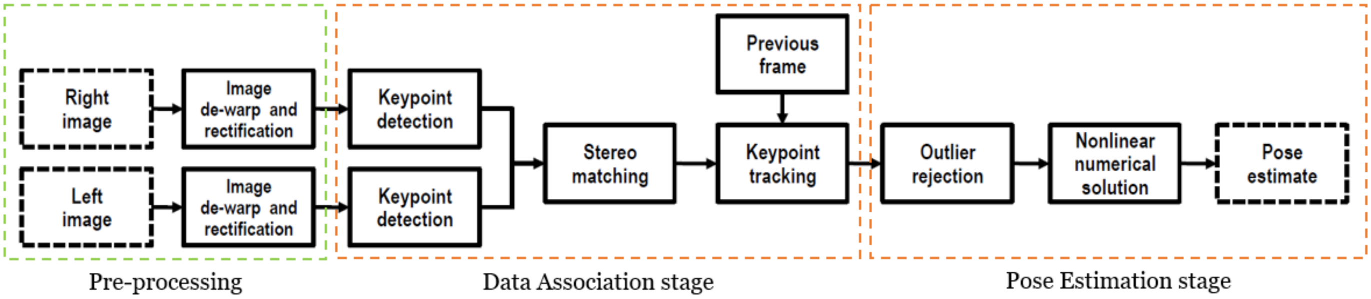


Fig. 1. The implemented stereo visual odometry pipeline. The pre-processing stage has been conducted by the KITTI Vision Benchmark Suite. This study focuses on the implementation of the Data Association and Pose Estimation stages.

stereo VO can be concluded as Fig. 1. This pipeline can be further divided into three stages: image pre-processing, data association, and pose estimation. The variations between different algorithms often appear at the data association stage and the pose estimation stage whereas the pre-processing is simply to de-warp and rectify the images so that an ideal stereo geometry is achieved for the images captured at the same time instant [2].

In the data association stage, two factors make algorithms variant. The first factor refers to the choice of feature detection-description techniques, and the second is the selection of feature matching mechanism [15].

In Moravec's first VO pipeline [11], he introduced one of the earliest feature detectors, the Moravec corner detector, to detect image features. Since then, many feature detection-description techniques were invented and applied in VO applications. For instances, Harris corner detector was used by Cheng et al. in [3], SIFT was adopted by Tardif et al. in [18], and SURF was employed by Lee et al. in [8]. The different choice of feature detection-description techniques can considerably influence the robustness and computational effort of a VO algorithm. In this project, the SURF feature detector is used because of its high robustness and relatively fast computational speed [19].

Regarding the feature matching mechanism, there exist two main approaches to find feature correspondences between timesteps. The first approach is to identify a set of image features in the current frame and track them in the following images using local search techniques (e.g. [3]). The second method is to independently detect salient and repeatable image features in all images and match them based on a similarity metric such as normalized cross correlation (NCC) or sum of squared difference (SSD) [17]. In this study, the latter method is employed, and the implementation details are presented in Section III.

Once the data association is finished, there are three broad categories of methods that can be applied to estimate the camera motion, namely, 2D-to-2D, 3D-to-2D and 3D-to-3D methods [15]. The 2D-to-2D methods involve the process of computing the essential matrix based on at least five 2D-to-2D feature correspondences. The camera rotation and translation between the previous and current timesteps can be recovered from the essential matrix [17]. Nister in [12] proposed a five-point algorithm which can be used to estimate the relative motion up to an unknown scale factor. For 3D-to-2D methods, the term *3D* refers to the 3D positions of the image features from the previous frame, while the term *2D* refers to the 2D image plane coordinates of the image features in the current frame [17]. The features from the previous frame (3D) are reprojected to the image plane of current frame, and an estimator is designed to find a transformation that minimizes the image reprojection error (e.g. [13]). In contrast, the 3D-to-3D methods project both previous and current features into the 3D space. A point cloud alignment problem is then set up to compute the transformation which minimizes the sum of 3D position errors of the feature points (e.g. [3]).

In this study, the SURF feature detector and descriptor are used to determine intra-frame and inter-frame feature matches by employing the SSD similarity metric. A 3D-to-3D point-cloud alignment method is then used to compute the relative transformation between the previous frame and the current frame. The full trajectory of the moving agent (i.e. a car in this case) can then be estimated by compounding the frame-to-frame transformations. It is assumed that the cameras are rigidly mounted on the moving vehicle, and the whole system performs as a rigid body. For every image sequence (i.e. every independent dataset), the first frame is set to be the stationary world frame, and the trajectory of the movement is expressed with respect to this inertial frame. The following section (Section III) explains the methodology and implementation details of this project.

III. Methodology and Implementation Details

In this study, we have implemented the VO pipeline shown in Fig. 1. The pre-processing stage was carried out by the KITTI Vision Benchmark Suite (KITTI) [6], and the implementation of rest of the pipeline is separated into two stages. In this section, we first describe the implementation details of the data association stage, followed by elaborating the methods used to perform pose estimation.

All of the datasets are extracted from the KITTI's database with an emphasis on the raw data under the City data category. The calibration parameters and groundtruth values are directly extracted from KITTI to facilitate and evaluate our

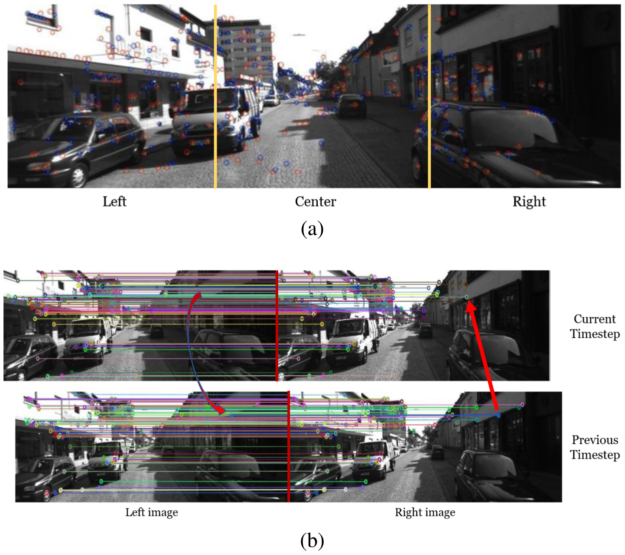
VO implementation. All of our codes are written and run in MATLAB on a PC installed with a Windows 10 system.

A. Data Association

There are three steps involved in the data association stage: feature detection, feature-based stereo matching, and between-frame feature matching. For feature detection, we detect salient image features independently on every image by using the open-source package, OpenSURF. During the feature detection process, every image is divided into three non-overlapping portions (i.e. left, center and right portions as shown in Fig. 2) to detect nonrotational-irrelevant image features. A dynamic threshold is used to ensure that every portion of the image (375×414 pixels per portion) contains at least 500 SURF features. The reasons to horizontally divide the images into three portions are mainly due to three considerations. First, by detecting features independently within each portion, we can ensure that image features are evenly distributed over every image. Second, due to the fact that the dynamic range of the camera is narrower than the radiance range of the outdoor environment, we can often observe either over-exposure in the sky, or under-exposure on the ground. By horizontally segmenting an image, we can ensure that every portion of the image contains both the sky and ground regions, so that salient features can be successfully detected. Third, since we are using a RANSAC based outlier rejection scheme in the pose estimation stage, three feature points would be required to compute a set of valid rotation and translation between two consecutive frames. By dividing an image into three portions, we force the first one third of the RANSAC trails to draw one point from each of the portion. In this way, we can guarantee that the selected inlier set after the first one third of the RANSAC trails is generated based on a transformation which has taken all image regions into account.

After detecting features on both the left and right images captured at the same timestep, we perform stereo matching by employing the SSD similarity metric on the 64-dimensional feature descriptors [19]. Because the stereo cameras are well calibrated, the stereo matching is done strictly along the epipolar line with only four pixels of offset buffer above and below it. This not only increases the matching accuracy, but also significantly improves the computational efficiency as compared to a brute-force matching [3].

In addition to the standard stereo matching process, we have also applied the following criteria to further filter the detected features. First, we remove the feature pairs that have a disparity value less than 7 pixels. This is approximately equal to a distance of 55 m away from the camera frame. The reason to remove these features is because objects that are far away from the camera are usually accompanied with large uncertainties [1]. Second, we remove the feature correspondences that have a disparity value more than 77 pixels (i.e. within 5 m of the camera). This is because we have observed that feature pairs with large disparity values are often mismatches. In this way, we can repeat the same procedure for every pair of stereo images, and we will obtain one set of within-frame feature correspondences for every timestep.



(a)

(b)

Fig. 2. (a) Images are divided into three portions, left, center and right, to detect SURF features. (b) Between-frame feature matching is performed in a circular fashion following a similar method proposed by Geiger et al. in [5]

To perform between-frame feature matching, we adopt a matching scheme that is similar to the one proposed by Geiger et al. in [5]. The overall matching mechanism is done in a circular fashion between the left and right images over the current and previous timesteps (Fig. 2b). For every two consecutive frames (i.e. current and previous frames), we start from all feature candidates in the *current left* image to find the best match in the *previous left* image by using the SSD metric. Since we have already found the within-frame feature correspondences in the stereo matching step, we can quickly find the corresponding feature in the *previous right* image. We then find the best matches between the *previous right* image and the *current right* image. A 'circle match' gets accepted, if the between-frame correspondence for the *previous right* and *current right* images coincides with the within-frame correspondence for the current timestep (i.e. current left and right). After identifying all the 'circle matches', a threshold is applied to only retain the matches that possess a small SSD.

In additional to the between-frame feature matching method stated above, we also apply a dynamic SSD threshold to ensure that each of the left, center and right portions of the images contains at least 40 between-frame feature matches. We have seen noticeable improvement in terms of trajectory accuracy by performing this to avoid nonexistence of feature in an image portion.

By performing the above procedures, we are able to obtain one set of between-frame feature correspondences for every two consecutive timesteps. We then use these between-frame feature correspondences to compute the egomotion of the vehicle in the pose estimation stage.

B. Pose Estimation

At this point, we have obtained the between-frame feature correspondences for every two consecutive timesteps. Before providing details about the pose estimation, we first define our midpoint stereo camera model as (1), and inverse camera model as (2), where $\mathbf{K}$ is the stereo camera intrinsic matrix. We assume the stereo camera as a pair of perfect pinhole cameras with focal lengths f_u, f_v and principal points (c_u, c_v), separated by a fixed and known baseline, l. We define the homogeneous coordinate of a 3D landmark P_j, expressed in a vehicle frame, $\underrightarrow{\mathcal{F}}_v$, to be $\tilde{\mathbf{p}}_v^{p_j v} = [(\mathbf{p}_v^{p_j v})^T \ 1] = [x_v \ y_v \ z_v \ 1]^T$.

$$\mathbf{y} = \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix} = \mathbf{f}(P) = \mathbf{K}\,\frac{1}{z}\tilde{\mathbf{p}} = \begin{bmatrix} f_u & 0 & c_u & f_u\frac{l}{2} \\ 0 & f_v & c_v & 0 \\ f_v & 0 & c_u & -f_u\frac{l}{2} \\ 0 & f_v & c_v & 0 \end{bmatrix} \frac{1}{z}\tilde{\mathbf{p}} \quad (1)$$

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{g}(\mathbf{y}) = \frac{l}{u_l - u_r} \begin{bmatrix} \frac{1}{2}(u_l + u_r) - c_u \\ \frac{f_u}{f_v}(\frac{1}{2}(v_l + v_r) - c_v) \\ f_u \end{bmatrix} \quad (2)$$

Problem Setup. Having defined the camera models, we then move on to the pose estimation stage to compute the poses of the moving vehicle. In this study, it is assumed that the camera is rigidly mounted on the vehicle, and thus, the trajectory of vehicle motion is equivalent to the trajectory of camera motion.

The egomotion of the vehicle can be estimated by setting up a weighted-least-squares point cloud alignment problem. The same problem setup is used for every two consecutive timesteps, hence, a sequence of incremental motions can be obtained. By compounding all of the frame-to-frame pose estimates, a trajectory of the vehicle movement can finally be produced. In this section, we call the k^{th} and $(k\text{-}1)^{th}$ timesteps as current and previous timesteps, respectively.

The point-cloud alignment problem setup is illustrated in Fig. 3, and we define the following quantities:

$\mathbf{T}_{V_k V_{k-1}}$: 4×4 transformation from $\underrightarrow{\mathcal{F}}_{V_{k-1}}$ to $\underrightarrow{\mathcal{F}}_{V_k}$;
$\mathbf{C}_{V_k V_{k-1}}$: 3×3 rotation matrix from $\underrightarrow{\mathcal{F}}_{V_{k-1}}$ to $\underrightarrow{\mathcal{F}}_{V_k}$;
$\mathbf{r}_{V_{k-1}}^{V_k V_{k-1}}$: 3×1 translation of V_{k-1} to V_k expressed in $\underrightarrow{\mathcal{F}}_{V_{k-1}}$;
$\mathbf{p}_{V_{k-1}}^{P_j V_{k-1}}$: 3×1 projected coordinate based on $\mathbf{y}_{V_{k-1}}^j$ in $\underrightarrow{\mathcal{F}}_{V_{k-1}}$;
$\mathbf{p}_{V_k}^{P_j V_k}$: 3×1 projected coordinate based on $\mathbf{y}_{V_k}^j$ in $\underrightarrow{\mathcal{F}}_{V_k}$;
$\mathbf{y}_{V_k}^j$: 4×1 stereo measurement of P_j at V_k;
$\mathbf{y}_{V_{k-1}}^j$: 4×1 stereo measurement of P_j at V_{k-1}.

In this setup, we want to estimate the rotation matrix, $\mathbf{C}_{V_k V_{k-1}}$, and translation vector, $\mathbf{r}_{V_{k-1}}^{V_k V_{k-1}}$ by using the measurements of between-frame feature correspondences that we have identified in the data association stage. For the measurements, we have M pairs of between-frame feature correspondences, $(\mathbf{y}_{V_k}^j, \mathbf{y}_{V_{k-1}}^j)$, where $j = 1, \ldots, M$. Each pair is a correlated set of measurements with respect to the same point, P_j at either the current or previous timestep. We assume all the measurements are corrupted by some zero-mean Gaussian noise. In order to construct a 3D-to-3D point cloud alignment problem, we project the image plane measurement, $(\mathbf{y}_{V_k}^j, \mathbf{y}_{V_{k-1}}^j)$, into the 3D Euclidean space through the inverse camera model shown in (2). The projected coordinates based on $(\mathbf{y}_{V_k}^j, \mathbf{y}_{V_{k-1}}^j)$ are, respectively, $(\mathbf{p}_{V_k}^{P_j V_k}, \mathbf{p}_{V_{k-1}}^{P_j V_{k-1}})$. In this way, we can project all M correspondences into the 3D space, and as a result, we obtain two point clouds associated with the same set of 3D landmarks.

$$\mathbf{T}_{V_k V_{k-1}} = \begin{bmatrix} \mathbf{C}_{V_k V_{k-1}} & -\mathbf{C}_{V_k V_{k-1}} \mathbf{r}_{V_{k-1}}^{V_k V_{k-1}} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

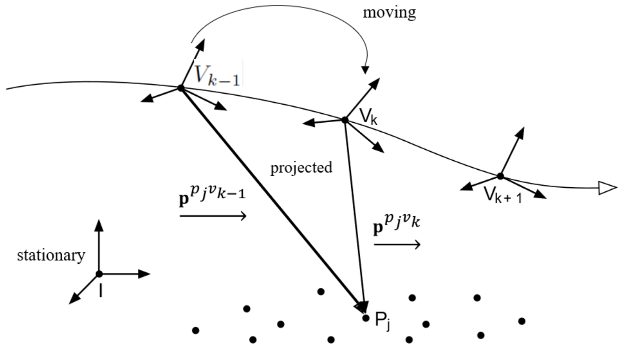


Fig. 3. Frame diagram of the point cloud alignment problem

After obtaining the two point clouds for both the current and previous frames, there are two steps remaining to acquire the final pose estimate. The first step refers to an *Outlier Rejection* step, and the second is the *Nonlinear Numerical Solution* step. In *Outlier Rejection*, we accomplish two tasks: 1. Generate an initial guess of the pose estimate based on a scalar-weighted solution; and, 2. Generate an inlier set which is used in the nonlinear numerical (i.e. matrix-weighted) solution. Due to space limitations in this document, more emphasis is put on the matrix-weighted solution, and only the key steps that are required to implement the scalar-weighted solution are included. An in-depth explanation on the scalar-weighted solution, including problem setup, cost definition and equation derivation can be found in [2]. In order to keep the notation concise, we will use the subscript b and a to represent the current and previous timesteps, respectively.

Outlier Rejection and Scalar-weighted Solution. The RANSAC based *Outlier Rejection* step is performed following the algorithm depicted in Table I. The process given in Table I is performed repeatedly until a user-defined number of trials have been conducted. During each trail, we first randomly pick three matched correspondences (i.e. three from one point cloud matched to three from the other) and determine their corresponding centroids. The $\mathbf{W}_{ba}$ matrix is then computed by using (3):

$$\mathbf{W}_{ba} = \frac{1}{\sum_{j=1}^{3} w^j} \sum_{j=1}^{3} w^j \, (\mathbf{p}_b^j - \mathbf{p}_b)(\mathbf{p}_a^j - \mathbf{p}_a)^T, \quad (3)$$

where, w^j is the scalar weight associated with the j^{th} correspondence (set to 1 in our case), with $j \in [1, 3]$; $\mathbf{p}_a$ and $\mathbf{p}_b$ are the centroids of the three selected correspondences from point cloud a and b, respectively; and, $\mathbf{p}_a^j$ and $\mathbf{p}_b^j$ are the j^{th} selected correspondences from point cloud a and b, respectively. After that, we first carry out a singular-value decomposition on the $\mathbf{W}_{ba}$ matrix:

$$\mathbf{V}\mathbf{S}\mathbf{U}^T = \mathbf{W}_{ba}, \quad (4)$$

TABLE I
PSEUDOCODE OF THE RANSAC BASED OUTLIER REJECTION STEP

```
WHILE INDEX < MAXITER
    index ← index + 1
    $Set_a \leftarrow randomly\ pick\ 3\ points\ in\ frame\ a$
    $Set_b$ ← pick the 3 correspondences of $Set_a$ in frame $b$
    $Centroid_a \leftarrow (Set_a^1 + Set_a^2 + Set_a^3)\ ./\ 3$
    $Centroid_b \leftarrow (Set_b^1 + Set_b^2 + Set_b^3)\ ./\ 3$
    $W_{ba} \leftarrow \frac{1}{3}\sum_{j=1}^{3}(Set_b^j - Centroid_b)(Set_a^j - Centroid_a)^T$
    $[\mathbf{U}, \mathbf{S}, \mathbf{V}] \leftarrow \mathrm{svd}(W_{ba})$
    $\mathbf{C}_{ba}^{temp}$ ← Eq. (5)
    $\mathbf{r}_{ba}^{temp}$ ← Eq. (6)
    $inliers_a^{temp}$ ← points that fulfill Eq. (7) in frame $a$
    $inliers_b^{temp}$ ← points that fulfill Eq. (7) in frame b
    if count($inliers_a^{temp}$) > $maxinliers$ do
        $maxinliers$ ← count($inliers_a^{temp}$)
        $inliers_a \leftarrow inliers_a^{temp}$
        $inliers_b \leftarrow inliers_b^{temp}$
    end
END
```

and then compute the unknown rotation between these three correspondences as:

$$\mathbf{C}_{ba} = \mathbf{V}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & \det\mathbf{U}\,\det\mathbf{V}\end{bmatrix}\mathbf{U}^T \tag{5}$$

and translation as:

$$\mathbf{r}_a^{ba} = -\mathbf{C}_{ba}^T\mathbf{p}_b + \mathbf{p}_a. \tag{6}$$

After determining the rotation and translation associated with the three selected correspondences, we then apply this set of $\mathbf{C}_{ba}$ and $\mathbf{r}_a^{ba}$ to all the remaining points, and classify them as either inlier or outlier based on their residual error. That is:

$$J^j = \frac{1}{2}w^j(\mathbf{p}_b^j - \mathbf{C}_{ba}(\mathbf{p}_a^j - \mathbf{r}_a^{ba}))^T(\mathbf{p}_b^j - \mathbf{C}_{ba}(\mathbf{p}_a^j - \mathbf{r}_a^{ba})) \leqslant J_{thresh}, \tag{7}$$

where, J_{thresh} is the user-defined threshold used to classify whether a point is inlier or outlier; and w^j is the scalar weight associated with the j^{th} correspondence (set to 1 in our case), with $j \in [1, M]$.

After all the trails are done, we obtain a set of inliers, $\{inliers_a, inliers_b\}$, and its corresponding $\{\mathbf{C}_{ba}, \mathbf{r}_a^{ba}\}$. We use this set of $\{\mathbf{C}_{ba}, \mathbf{r}_a^{ba}\}$ as our initial guess for the pose estimate and pass the inlier set into the *Nonlinear Numerical solution* step to compute our final estimate.

It is worthwhile to reiterate that we force the first one third of the RANSAC trails to draw one point from each of the left, center and right portions of the image. In this way, we can guarantee that the selected inlier set after the first one third of the trails is generated based on a transformation which has taken all regions of the image into account.

Matrix-weighted Point Cloud Alignment. In stereo vision, points that are far away from the camera frame have more uncertainty than those that are close. Moreover, measurements made by stereo cameras have greater noise in the depth direction than the lateral directions (i.e. noise is not isotropic) [1]. Since we have assigned a scalar weight of 1.0 to every feature in the scalar-weighted solution, we have not taken any of these effects into account which will lead to sub-optimal pose estimates. Therefore, we need to take the point uncertainty into account during the pose estimation process in order to obtain an optimal solution. To do that, we assume the camera observation, $\mathbf{y}$, are corrupted by some zero-mean Gaussian sensor noise, $\mathbf{n}$:

$$\mathbf{y} = \mathbf{f}(\mathbf{p}) + \mathbf{n}, \qquad \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{8}$$

where, $\mathbf{f}(\cdot)$ is the forward camera model defined in (1); and $\mathbf{R}$ is the sensor noise covariance. When we project an image feature back through the inverse camera model, we will have:

$$\hat{\mathbf{p}} = \mathbf{g}(\mathbf{y}) = \mathbf{g}(\mathbf{f}(\mathbf{p}) + \mathbf{n}) \approx \mathbf{p} + \mathbf{G}\,\mathbf{n}, \quad \mathbf{G} = \left.\frac{\partial\mathbf{g}}{\partial\mathbf{y}}\right|_{\mathbf{y}}, \tag{9}$$

where, $\hat{\mathbf{p}}$ is the estimated landmark position; $\mathbf{p}$ is the true landmark position; $\mathbf{g}(\cdot)$ is the inverse camera model defined in (2), and $\mathbf{G}$ is the Jacobian of the inverse camera model with respect to the camera measurement. In this way, the estimate of the landmark behaves as:

$$\hat{\mathbf{p}} \sim \mathcal{N}(\mathbf{p}, \mathbf{G}\,\mathbf{R}\,\mathbf{G}^T), \tag{10}$$

and the Jacobian of the inverse camera model is defined as:

$$\mathbf{G} = \frac{l}{(d)^2}\begin{bmatrix}-u_r + c_u & 0 & u_l - c_u & 0\\ -(\frac{v_l+v_r}{2} - c_v) & \frac{1}{2}d & \frac{v_l+v_r}{2} - c_v & \frac{1}{2}d\\ -f_u & 0 & f_u & 0\end{bmatrix}, \tag{11}$$

where, $d = (u_l - u_r)$ is the disparity value in pixels; and we have omitted the fraction between f_u and f_v to keep the equation concise because the camera in our problem have the same focal length in both horizontal and vertical directions.

By incorporating the matrix point uncertainty, the objective function is defined as:

$$J = \frac{1}{2}\sum_{j=1}^{M}\left(\mathbf{p}_b^j - \mathbf{C}_{ba}(\mathbf{p}_a^j - \mathbf{r}_a^{ba})\right)^T\boldsymbol{\Sigma}^j((\mathbf{p}_b^j - \mathbf{C}_{ba}(\mathbf{p}_a^j - \mathbf{r}_a^{ba})), \tag{12}$$

$$\boldsymbol{\Sigma}^j = (\mathbf{G}_b^j\mathbf{R}_b^j\mathbf{G}_b^{j\,T} + \mathbf{C}_{ba}\mathbf{G}_a^j\mathbf{R}_a^j\mathbf{G}_a^{j\,T}\mathbf{C}_{ba}{}^T)^{-1}, \tag{13}$$

$$\mathbf{G}_b^j = \left.\frac{\partial\mathbf{g}}{\partial\mathbf{y}}\right|_{\mathbf{f}(\mathbf{p}_b^j)}, \quad \mathbf{G}_a^j = \left.\frac{\partial\mathbf{g}}{\partial\mathbf{y}}\right|_{\mathbf{f}(\mathbf{p}_a^j)}, \tag{14}$$

where, $\boldsymbol{\Sigma}^j$ is the 3×3 point uncertainty associated with the j^{th} landmark; $\mathbf{G}_a^j$ and $\mathbf{G}_b^j$ are the 3×4 Jacobians of inverse camera model with $\mathbf{f}(\mathbf{p}_a^j)$ and $\mathbf{f}(\mathbf{p}_b^j)$ as operating points, respectively; $\mathbf{C}_{ba}$ is the 3×3 rotation matrix from frame a to frame b; $\mathbf{r}_a^{ba}$ is the 3×1 translation vector from frame a to b, expressed in frame a; and, $\mathbf{R}_a^j$ and $\mathbf{R}_b^j$ are the 4×4 image plane uncertainties of the j^{th} feature point in frame a and b, respectively. Note that in this study, we have assumed an image plane sensor noise of 2 pixels to all the detected feature points (i.e. a variance of 4 [pixel^2]) in both the horizontal and vertical image plane directions.

As shown in (12), the objective function includes a matrix weight, $\boldsymbol{\Sigma}$, which means that we cannot obtain the optimal solution in closed form. In this case, the pose estimation problem must be solved iteratively, and we will use the Gauss-Newton method to solve the system. In order to minimize the objective function by iteratively optimizing $\mathbf{C}_{ba}$ and $\mathbf{r}_a^{ba}$ with the initial guess obtained from the scalar-weighted solution, we first introduce the following perturbation terms:

$$\mathbf{r}_a^{ba} = \bar{\mathbf{r}}_a^{ba} + \boldsymbol{\epsilon}, \; \mathbf{C}_{ba} = e^{-\boldsymbol{\phi}^\wedge}\bar{\mathbf{C}}_{ba} \approx (\mathbf{1} - \boldsymbol{\phi}^\wedge)\bar{\mathbf{C}}_{ba}, \; \boldsymbol{\xi} = \begin{bmatrix}\boldsymbol{\epsilon}\\ \boldsymbol{\phi}\end{bmatrix}, \tag{15}$$

where, the (3×1) $\boldsymbol{\epsilon}$ and the (3×1) $\boldsymbol{\phi}$ are the perturbations for translation and rotation, respectively; $\bar{\mathbf{C}}_{ba}$ and $\bar{\mathbf{r}}_a^{ba}$ are the solutions for rotation and translation from the previous iteration, respectively; $e^{(\cdot)}$ is the matrix exponential map operator; and $(\cdot)^\wedge$ is the skew-symmetric operator, which is defined as:

$$\boldsymbol{\phi}^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}. \tag{16}$$

After inserting the perturbations into the objective function and defining some intermediate terms, we can express the objective function as:

$$J \approx \frac{1}{2} \sum_{j=1}^{M} (\bar{\mathbf{e}}^j + \mathbf{E}^j \boldsymbol{\xi})^T \boldsymbol{\Sigma}^j (\bar{\mathbf{e}}^j + \mathbf{E}^j \boldsymbol{\xi}), \tag{17}$$

$$\bar{\mathbf{e}}^j = \mathbf{p}_b^j - \bar{\mathbf{C}}_{ba}(\mathbf{p}_a^j - \bar{\mathbf{r}}_a^{ba}),\ \mathbf{E}^j = \left[\bar{\mathbf{C}}_{ba} \ \ -(\bar{\mathbf{C}}_{ba}\,(\mathbf{p}_a^j - \bar{\mathbf{r}}_a^{ba}))^\wedge\right]. \tag{18}$$

This is exactly quadratic in the perturbations (after dropping the product of two small terms). To find the optimal update, $\boldsymbol{\xi}^*$, we can take the derivative of (17) with respect to the perturbations and set it to zero:

$$\frac{\partial J}{\partial \boldsymbol{\xi}^T} = \sum_{j=1}^{M} \mathbf{E}^{j^T} \boldsymbol{\Sigma}^j (\bar{\mathbf{e}}^j + \mathbf{E}^j \boldsymbol{\xi}) \tag{19}$$

$$\left(\sum_{j=1}^{M} \mathbf{E}^{j^T} \boldsymbol{\Sigma}^j \mathbf{E}^j \right) \boldsymbol{\xi}^* = -\sum_{j=1}^{M} \mathbf{E}^{j^T} \boldsymbol{\Sigma}^j \bar{\mathbf{e}}^j, \tag{20}$$

where, $\boldsymbol{\xi}^*$ is the 6×1 optimal perturbations to update the current estimates of rotation and translation:

$$\bar{\mathbf{r}}_a^{ba} \leftarrow \bar{\mathbf{r}}_a^{ba} + \boldsymbol{\epsilon}^*, \quad \bar{\mathbf{C}}_{ba} \leftarrow e^{-\boldsymbol{\phi}^{*\wedge}} \bar{\mathbf{C}}_{ba}, \quad \boldsymbol{\xi}^* = \begin{bmatrix} \boldsymbol{\epsilon}^* \\ \boldsymbol{\phi}^* \end{bmatrix}. \tag{21}$$

We then repeat the above process until convergence (i.e. $\boldsymbol{\xi}^*$ is sufficiently small). At this point, we have finished implementing the entire VO pipeline as stated in Fig. 1, and acquired a whole sequence of pose changes between two consecutive camera frames, $\{(\mathbf{r}_{v_1}^{v_2 v_1}, \mathbf{C}_{v_2 v_1}), \ldots, (\mathbf{r}_{v_{k-1}}^{v_k v_{k-1}}, \mathbf{C}_{v_k v_{k-1}})\}$. Note that we have used the subscript 1 to denote the initial timestep which complies with the indexing scheme in MATLAB. To estimate the pose of the vehicle at the k^{th} timestep ($k \in [1, N]$, where N is the total number of frames), we recursively compound the frame-to-frame estimates to produce an estimate relative to the stationary inertial frame (i.e. the first frame of the entire sequence):

$$\mathbf{C}_{v_k v_1} = \mathbf{C}_{v_k v_{k-1}} \mathbf{C}_{v_{k-1} v_1};\ \mathbf{r}_{v_1}^{v_k v_1} = \mathbf{C}_{v_{k-1} v_1}^T \mathbf{r}_{v_{k-1}}^{v_k v_{k-1}} + \mathbf{r}_{v_1}^{v_{k-1} v_1} \tag{22}$$

By doing so, we are able to recover the entire trajectory of the vehicle motion.

C. Robust M-Estimation

To further improve our estimator, one can incorporate a robust cost function in the matrix-weighted solution to further reduce the influence of outliers. As mentioned by Kaess et al., even though the RANSAC technique can effectively filter majority of the outliers, its performance is sensitive to the user-defined threshold, and poor performance may be observed in nearly degenerate data (e.g. bad lighting condition, motion blur) [7]. As demonstrated by MacTavish and Barfoot in [9], the robust M-estimation technique can also be used to effectively increase the robustness of the estimator.

In this study, we use the Geman-McClure (G-M) robust cost function to modify the shape of the original sum-of-squared-error objective function (17). The G-M function used in this study is one member of the G-M family, which has the form:

$$\rho(u) = \frac{1}{2} \frac{u^2}{1 + u^2}. \tag{23}$$

To incorporate with our original objective function (17), we define the following:

$$u_j(\boldsymbol{\psi}^j) = \sqrt{\boldsymbol{\psi}^{j^T} \boldsymbol{\Sigma}^j \boldsymbol{\psi}^j}, \quad \boldsymbol{\psi}^j(\boldsymbol{\xi}) = \bar{\mathbf{e}}^j + \mathbf{E}^j \boldsymbol{\xi}, \tag{24}$$

where, $\boldsymbol{\Sigma}^j$, $\bar{\mathbf{e}}^j$, $\mathbf{E}^j$, and $\boldsymbol{\xi}$ follow the same definitions as above. As a result, the original cost is simply, $J = \frac{1}{2} \sum_{j=1}^{M} u_j^2$, and the new objective function, $J^{'}(\boldsymbol{\xi})$, can be obtained by substituting (24) into (23). The gradient of the new objective function is determined by using the chain rule:

$$\frac{\partial J^{'}}{\partial \boldsymbol{\xi}} = \sum_{j=1}^{M} \alpha_j \frac{\partial \rho}{\partial u_j} \frac{\partial u_j}{\partial \boldsymbol{\psi}^j} \frac{\partial \boldsymbol{\psi}^j}{\partial \boldsymbol{\xi}}, \quad \frac{\partial \rho}{\partial u_j} = \frac{u}{(1 + u^2)^2}, \tag{25a}$$

$$\frac{\partial u_j}{\partial \boldsymbol{\psi}^j} = \frac{1}{u_j} \boldsymbol{\psi}^{j^T} \boldsymbol{\Sigma}^j, \quad \frac{\partial \boldsymbol{\psi}^j}{\partial \boldsymbol{\xi}} = \mathbf{E}^j, \tag{25b}$$

where, we have set α_j to 1. By taking the Jacobian of the new objective function and setting to zero, we will have:

$$\frac{\partial J^{'}}{\partial \boldsymbol{\xi}^T} = \sum_{j=1}^{M} \mathbf{E}^{j^T} \mathbf{Y}_j (\bar{\mathbf{e}}^j + \mathbf{E}^j \boldsymbol{\xi}), \tag{26}$$

$$\mathbf{Y}_j = \frac{1}{\left(1 + u_j(\boldsymbol{\xi}_{op})^2\right)^2} \boldsymbol{\Sigma}^j, \tag{27}$$

$$\left(\sum_{j=1}^{M} \mathbf{E}^{j^T} \mathbf{Y}_j \mathbf{E}^j \right) \boldsymbol{\xi}^* = -\sum_{j=1}^{M} \mathbf{E}^{j^T} \mathbf{Y}_j \bar{\mathbf{e}}^j, \tag{28}$$

where, $\boldsymbol{\xi}_{op}$ is the optimal update from the previous Gauss-Newton iteration. In this case, we solve the original least-squares problem at each iteration, but with a modified covariance matrix that updates as $\boldsymbol{\xi}_{op}$ updates. This is referred to as iteratively re-weighted least squares in literature [9].

D. Uncertainty Propagation

In order to measure the uncertainty associated with the k^{th} pose estimate, we need to compound the uncertainty associated with each incremental motion from the first timestep up to the k^{th} timestep. To simplify the notation, we use transformation matrix in this section instead of treating rotation and translation separately, where the estimated transformation matrix between two constitutive frames is defined as:

$$\bar{\mathbf{T}}_{v_k v_{k-1}} = \begin{bmatrix} \bar{\mathbf{C}}_{v_k v_{k-1}} & -\bar{\mathbf{C}}_{v_k v_{k-1}} \bar{\mathbf{r}}_{v_{k-1}}^{v_k v_{k-1}} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{29}$$

Since we have set the first frame as our inertial frame, we assume zero uncertainty when $k = 1$ (i.e. $\hat{\mathbf{P}}_{v_1} = \mathbf{0}$). For $k \in [2, N]$, we have the pose uncertainty defined as (30a) if (20) is used (i.e. original objective function is employed), or as (30b) if (28) is used (i.e. robust cost function is employed).

$$\hat{\mathbf{P}}_{v_k} = \left(\sum_{j=1}^{M} \mathbf{E}^{j^T} \boldsymbol{\Sigma}^j \mathbf{E}^j \right)_{v_k}^{-1} + \bar{\mathcal{T}}_{v_k v_{k-1}} \hat{\mathbf{P}}_{v_{k-1}} \bar{\mathcal{T}}_{v_k v_{k-1}}^T\ , \tag{30a}$$

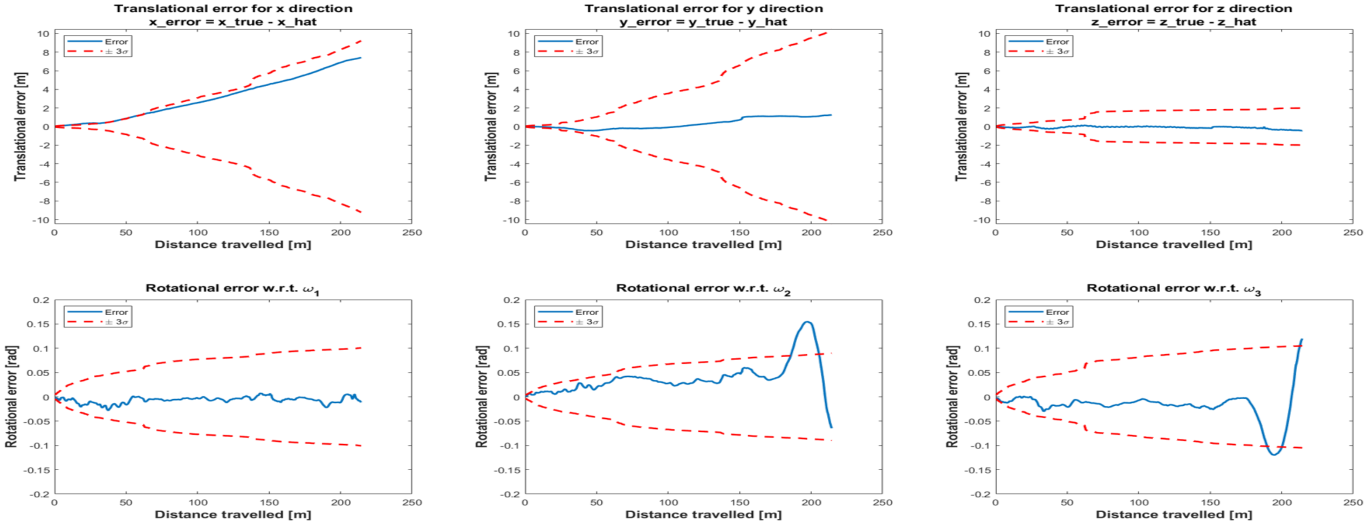


Fig. 4. Error plots of the VO implementation on the *2011_09_26_drive_0059* dataset. Errors are shown with respect to the camera frame, where the camera x-axis coincides with the vehicle pitch-axis, camera y-axis coincides with the vehicle yaw-axis, and the camera z-axis coincides with the vehicle roll-axis.

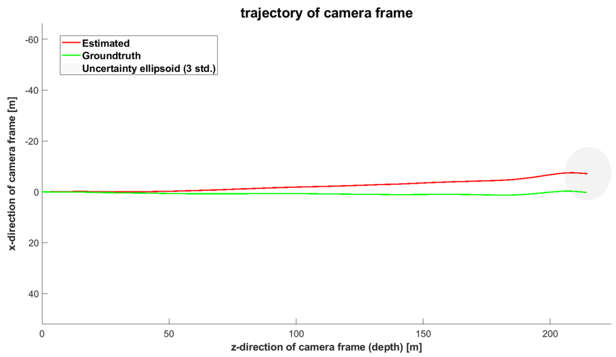


Fig. 5. Estimated vs true trajectory of the *2011_09_26_drive_0059* dataset.

TABLE II
ENDING-POINT TRANSLATIONAL AND ROTATIONAL ERRORS OVER THE TOTAL DISTANCE TRAVELLED ON SEVERAL KITTI DATASETS

Dataset ID	Distance Travelled	Translation Error	Rotational Error
2011_09_26_0059	214 m	2.8 %	0.0096 [rad/m]
2011_09_26_0009	254 m	5.6 %	0.0080 [rad/m]
2011_09_26_0011	113 m	1.5 %	0.0034 [rad/m]
2011_09_26_0013	173 m	2.6 %	0.0121 [rad/m]
2011_09_26_0095	62 m	7.2 %	0.0342 [rad/m]
2011_09_26_0096	383 m	4.3 %	0.0052 [rad/m]

$$\hat{\mathbf{P}}_{v_k} = \left(\sum_{j=1}^{M} \mathbf{E}^{j^T} \mathbf{Y}_j \mathbf{E}^j\right)^{-1}_{v_k} + \bar{\mathcal{T}}_{v_k v_{k-1}} \hat{\mathbf{P}}_{v_{k-1}} \bar{\mathcal{T}}^T_{v_k v_{k-1}} \ , \tag{30b}$$

where, $\bar{\mathcal{T}}_{v_k v_{k-1}} = \mathrm{Ad}(\bar{\mathbf{T}}_{v_k v_{k-1}})$ is the 6×6 adjoint form of the estimated transformation matrix, which is defined as:

$$\bar{\mathcal{T}}_{v_k v_{k-1}} = \begin{bmatrix} \bar{\mathbf{C}}_{v_k v_{k-1}} & (-\bar{\mathbf{C}}_{v_k v_{k-1}} \bar{\mathbf{r}}^{v_k v_{k-1}}_{v_{k-1}})^\wedge \bar{\mathbf{C}}_{v_k v_{k-1}} \\ \mathbf{0} & \bar{\mathbf{C}}_{v_k v_{k-1}} \end{bmatrix}. \tag{31}$$

Therefore, the uncertainty associated with the pose estimate at each timestep can be computed.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we present the results of our VO implementation experimented on several KITTI datasets. The overall performance reveals the fact that further improvement is required. The error plots and the estimated trajectory plot regarding the *2011_09_26_drive_0059* dataset are illustrated in Fig. 4 and Fig. 5. The ending-point estimation errors on several other datasets are summarized in Table II.

In Fig. 4, the translational and rotational errors of the pose estimate at the k^{th} timestep is computed according to:

$$\delta\mathbf{r}_k = \begin{bmatrix} \delta r_{x,k} \\ \delta r_{y,k} \\ \delta r_{z,k} \end{bmatrix} = \mathbf{r}^{v_k i}_i - \bar{\mathbf{r}}^{v_k i}_i ; \ \delta\boldsymbol{\theta}^\wedge_k = \begin{bmatrix} \delta\theta_{x,k} \\ \delta\theta_{y,k} \\ \delta\theta_{z,k} \end{bmatrix}^\wedge = \mathbf{1} - \mathbf{C}_{v_k i} \bar{\mathbf{C}}^T_{v_k i}, \tag{32}$$

where, letter i denotes the stationary inertial frame (i.e. the first frame of the sequence); $\bar{\mathbf{C}}_{v_k i}$ and $\bar{\mathbf{r}}^{v_k i}_i$ are the estimated rotation and translation at the k^{th} timestep, respectively; $\mathbf{C}_{v_k i}$ and $\mathbf{r}^{v_k i}_i$ are the groundtruth rotation and translation; and $(\cdot)^\wedge$ is the skew-symmetric operator defined in (16).

As can be observed in Fig. 4, the estimation errors and uncertainty envelope grow without bounds as the vehicle moves further. This agrees with our expectation that VO suffers from unbounded accumulation of drift error over time. According to Olson et al. [14], the increasing orientation errors will lead to a super-linear growth of drift error with distance travelled. The small estimation errors associated with each timestep will compound together, and eventually result in a large drift in the trajectory estimation.

In this dataset (*2011_09_26_drive_0059*), majority of the estimation errors fall within the three standard-deviation uncertainty envelope. This means that our estimator can be considered as consistent. However, we have also observed that this consistency may not always be true in some other datasets.

As shown in Table II, the translational and rotational errors from our VO implementation are greater than 1.5% and 0.0034 [rad/m] on the experimented datasets. State-of-the-art VO algorithms often return an ending-point translational error of less than 1%, and a rotational error that is about 5 to 10 times smaller than our estimates. In addition, we can observe that large rotational error is correlated to large translational error in our case. This reveals the fact that further improvement is required to increase the performance of our VO algorithm.

In general, our VO implementation has better performance when the vehicle motion is relatively straight and smooth (e.g. smooth curve), while poor performance is produced when there exists large turns in the movement trajectory. Bad performance can also be observed when the vehicle undergoes frequent speed changes. Moreover, it has been found that the quality of the between-frame feature matches has crucial impact on the estimation accuracy. Several mismatches in the early stage of the image sequence may orientate the vehicle to a totally off direction. This kind of mismatches may affect the pose estimation stage if the vehicle is moving relatively slow while there are moving objects with strong image gradients in the scene. For instance, this kind of situation can happen when the vehicle is just about to start moving (e.g. after waiting for traffic light) while there are cyclists or pedestrians who are also slowly moving in the environment. In addition, objects like trees, grass and shadows which possess strong image gradients are often big sources of error in the data association stage.

V. Lessons Learned and Future Work

Overall, this project provides a good opportunity for the team to gain experience on processing stereo images and estimating vehicle egomotion through implementing the classic feature based VO pipeline. During this process, we have learned that there exist many factors which can affect the performance of a VO algorithm. In the data association stage, moving objects in a dynamic scene are usually big sources of errors which can lead to feature mismatches. The influence of these mismatches can accumulate with time and eventually cause inconsistency in the estimator. In addition, phenomena like motion blur and variation in lighting condition between image frames are big challenges not only for feature matching, but also feature detection and description. The detected salient features in one image frame can suddenly become unavailable in the next frame due to motion blur and illumination changes. Such effects can further result in detection and selection of many distant features. This is because the image plane motion and variation in pixel intensity for far-away features are usually small as compared to those features that are close. Due to the fact that far-away features possess more uncertainty, employing too many far-away landmarks can potentially introduce more errors into the estimation.

For the pose estimation stage, we have found that computing the vehicle transformation based on image features that are widely distributed on the image plane can lead to noticeable improvement in terms of trajectory accuracy. If rotation and translation are estimated according to features that are extracted from the same image region, it can lead to a pose estimate that is biased towards that specific local image patch. Moreover, we have noticed that there exists a certain level of randomness in our VO implementation even if the same initial condition is applied. This may be due to the inherent randomness in the RANSAC algorithm. Regarding the robust M-estimation, we have observed that the results generated by the M-estimation is smoother than the ones produced by using the original sum-of-squares objective function. This may be due to the weights assigned to each of the landmark points are more uniform in the M-estimation.

Future work for this project can be carried out by employing a 3D-to-2D pose estimation method, where the image plane reprojection error will be minimized. A sliding-window technique can also be applied to take information from multiple frames into account during each pose estimation process.

References

[1] Timothy Barfoot, James R. Forbes, and Paul T. Furgale. Pose estimation using linearized rotations and quaternion algebra. *Acta Astronautica*, 68(1):101 – 112, 2011.

[2] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, New York, NY, USA, 1st edition, 2017.

[3] Yang Cheng, Mark Maimone, and Larry Matthies. Visual odometry on the mars exploration rovers. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 903–910. IEEE, 2005.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, May 2014.

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

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[7] M. Kaess, K. Ni, and F. Dellaert. Flow separation for fast and robust stereo odometry. In *2009 IEEE International Conference on Robotics and Automation*, pages 3539–3544, May 2009.

[8] G. H. Lee, F. Faundorfer, and M. Pollefeys. Motion estimation for self-driving cars with a generalized camera. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2746–2753, June 2013.

[9] K. MacTavish and T. D. Barfoot. At all costs: A comparison of robust cost functions for camera correspondence outliers. In *2015 12th Conference on Computer and Robot Vision*, pages 62–69, June 2015.

[10] Larry Henry Matthies. *Dynamic Stereo Vision*. PhD thesis, 1989.

[11] Hans Peter Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford, CA, USA, 1980.

[12] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004.

[13] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004.

[14] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone. Stereo ego-motion improvements for robust rover navigation. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 1099–1104 vol.2, May 2001.

[15] Shashi Poddar, Rahul Kottath, and Vinod Karar. Evolution of visual odometry techniques. *ArXiv*, abs/1804.11142, 2018.

[16] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on Robotics*, 24(5):1015–1026, Oct 2008.

[17] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[18] J. 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*, pages 2531–2538, Sep. 2008.

[19] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.