Feature-Based Visual Odometry on Stereo Cameras

Tianze Li

May 2022

Abstract **— The project has implemented an essential subset of functions that commonly used in the established feature-based visual odometry on stereo vision. In particular, the estimation minimizes the re-projection error that formulated in perspective-n-point. It is solved by Gauss-Newton iteration on transform matrix group and the outliers are rejected by random sample consensus. The performance is evaluated on the Kitti stereo dataset. The implemented modules can also be extended to integrate with bundle adjustment and global pose-graph optimization, which are studied in this project but remain as a future work.**

I. Introduction

A. *Background*

Visual odometry (VO) is the process of estimating the egomotion of an agent using only the input of on-board cameras attached to it [1]. The significant upgrade in embedded processors and depth cameras in the past decade has exalted the potentiality of visual odometry, which constitutes a key part in the visual simultaneous localization and mapping (SLAM) that found numerous applications in autonomous driving, robot navigation and augmented reality.

Visual odometry analyses changes in the images of consecutive frames that induced by the motion. Varied pipelines have been developed for different types of cameras. A stereo camera usually consists of two monocular cameras, while a RGB-D camera combines a monocular camera with infrared sensors, therefore, by triangulation both types can directly measure the distance between the camera center to an object in the camera scope. A monocular camera, sometimes combined with heterogeneous sensors such as inertial measurement unit (IMU) and global navigation satellite system (GNSS), is capable of sufficiently accurate estimation for certain applications. It can estimate up to a scaling factor that can be determined by manual input or motion constraints [1].

Typically, the changes in the images of two consecutive frames are depicted either by the intensity of all the pixels or the location of certain features on the image plane that are ideally invariant against changes in the scale, orientation and lighting. In general, the feature-based methods are faster to compute and of higher precision [1], therefore has the priority to be studied and implemented in this project.

With a handful of matched feature points from the different frames, the bridging transform can be estimated. Bipolar constraints that often used on monocular cameras can estimate up to a scale when the the three dimensional positions of the correspondence points are unknown. If known in the first frame, perspective-n-point (P-n-P) method can be applied. When known in both frames, iterative closest point (ICP) can compute the transform independently of camera intrinsic parameters at a lower precision but suitable for sensors such as LiDAR.

We can improve the estimation by considering a sequence of frames together in a sliding window fashion, which is a process referred to as bundle adjustment. Compared to the aim of globally optimal estimation in visual SLAM based on loop-closure, visual odometry reaches the local optimal and do not keep a memory of states of the environment.

B. *Content*

This project consists of:

- Surveying and implementing feature-based visual odometry from scratch, with a special interest in the Gauss-Newton optimization on the Lie algebra of transfrom matrix group $SE(3)$ and random sample consensus (RANSAC) outlier rejection.
- Evaluating and analysing the performance on Kitti dataset.
- Studying and implementing the interface for future extensions to visual SLAM.

The implementation partly follows the structure proposed in open source project in [2]. Several modules such as RANSAC are directly implemented from the algorithms summarized in [1]. The theory of Gauss-Newton iteration on matrix Lie group $SE(3)$ is summarized in [2] and derived in details in [3]. The software dependencies are listed in Figure 1.

The source code of this project (in C++) can be found on GitHub at:

`https://github.com/TianzeLi/toySLAM`

II. Method

Considering the broad contents of visual odometry, we will only cover the relevant methods. The overall pipeline is summarized in Figure 1.

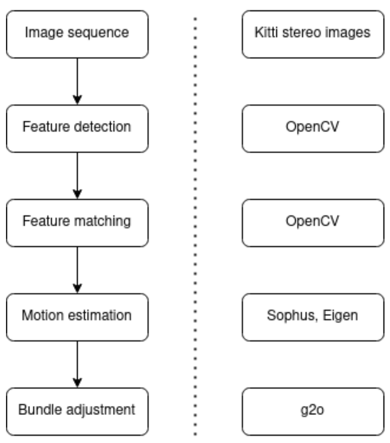

Figure 1: VO pipeline and dependencies. The motion of the camera can be recovered from changes of locations of certain object points' correspondences on the image. Thus the first step is to detect and match the correspondences and then estimate the motion based on them. Bundle adjustment can optimize the poses of several frames and the locations of related landmarks at the same time, thus enhances the precision.

A. *Image feature detection and matching*

Points projected as characteristic components in the image are considered to be landmarks as in the traditional estimation framework and the matching of features is considered as data association. In visual odometry, an image feature is usually detected as a corner point or blob and is associated with a vector, i.e. descriptor, to store its character and compare with other features. An ideal feature appears repetitively in different camera frames and holds invariant against the changes in lighting, scale and perspectives, thus is beneficial for motion estimation in the later step.

Among the feature detectors and descriptors, scale-invariant feature transform (SIFT) detects with difference of Gaussians and stores the intensity gradient value against the surrounding pixels as the descriptor. It is relatively robust but relies on heavy computation. Oriented FAST and rotated BRIEF (ORB) only record in the binary form whether certain surrounding pixels have a larger intensity value than the center pixel at interest, thus significantly reduces the computation. In particular, FAST detects corner points based on the difference in intensity.

When an object moves farther away from the camera, its projection looks similar as in the zoomed-out original image. Therefore the original image can be replicated into smaller scales for feature detection and the feature detected on all scales are stored and compared during matching to acquire the scale invariability. Further, an angle can be computed from the moments of the adjacent pixels to be associated with the feature. When generating the descriptor, the feature is first rotated against the computed angle, thus the features can still be matched when they appeared rotated on the image. These two procedures are followed by both SIFT and ORB.

With features from multiple images, we can match them based on the similarity, i.e. the distance of their descriptors. Brutal force algorithms are the most straightforward while other algorithms exist to ease the computation. In particular for ORB, the similarity of the binary descriptors is computed in terms of the Hamming distance, that is, the amount of different binary bits.

Although multiple processes are adopted to enhance the reliability of the matched pairs, outliers may still pass through and cause significant offsets in pose estimation. Therefore in the later steps, we can deploy RANSAC to further suppresses the outliers.

B. *Triangulation and motion estimation*

The transform between two poses of a camera in three dimensions can be represented by transition $t \in \mathbb{R}^3$ together with rotation matrix $R \in SO(3)$ that form a homogeneous transform $T \in SE(3)$:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{1}$$

For a calibrated RGB perspective camera, we assume the intrinsic parameter matrix K is known in the form:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where f_x and f_y depends on the focal length along x and y axes and c_x and c_y describe the pixel center on the image plane.

Figure 2 shows the scenario that the same spatial point P is projected to two images taken from different poses that associated with rotation R and transition t. We can relate the pixel location on the image plane and the three dimensional point coordinate with the perspective projection model [4] in each camera's own frame,

$$\begin{aligned} s_1 p_1 &= K_1 P_1 \\ s_2 p_2 &= K_1 P_2 \end{aligned} \tag{3}$$

where $p = [u, v, 1]^T$ represents the pixel location in the homogeneous form. P_i stands for the location of the object point P with regard to in the respective camera frame and s equals to depth Z of the point P, which is also referred to as the scaling factor.

For consecutive frames, features are matched to estimate the transform between camera poses. For each frame of a stereo camera, points are also matched in left and right image so that the depth of the corresponding object point can be computed by triangulation.

Triangulation

In Figure 2 we can see in camera one's frame, $P_1 = RP_2 + t$. Combined with Equation 3, we can obtain the depth with regard to both images by as the least

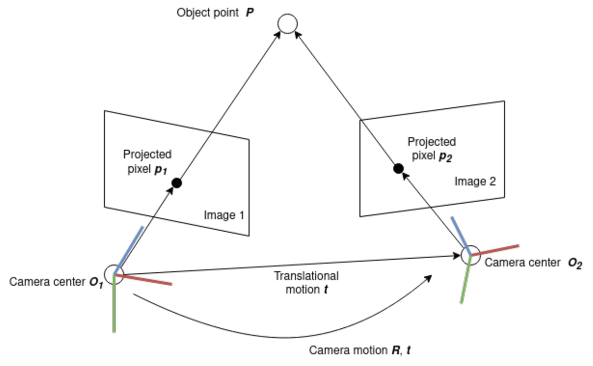

Figure 2: The corresponding pixel location of the same object changes with the pose of the camera [2].

square error solution for

$$\begin{bmatrix} K_1^{-1}p_1 & -RK_2^{-1}p_2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = t \tag{4}$$

We can evaluate the estimated depth by calculating $|K_1^{-1}p_1s_1 - RK_2^{-1}p_2s_2|/|t|$. The matched feature can be considered as an outlier when the value is too large. Also, we can set a threshold to exclude features with too large depths compared to the stereo camera baseline, since their precision will be relatively low.

C. *Perspective-n-point*

With the 3D locations of certain feature points and their 2D projections on the successive frames, we can estimate the transform by minimizing the re-projection error.

We adopt the projection model that also takes into consideration the transform between the absolute frame and the camera frame. With the camera intrinsic parameter matrix K, the transform R, t that transform the location of a point $p = [x, y, z]^T$ from absolute frame to the camera frame, we have:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KR \begin{bmatrix} x \\ y \\ z \end{bmatrix} + t \tag{5}$$

Rewriting in the homogeneous forms, such that $P = [x, y, z, 1]^T$ and $U = [u, v, 1]^T$, thus $sU = [KTP]_{1:3}$. Note that on the right hand side, only the first three rows of the vector are kept.

We can then write the sum of the re-projection error based on n pairs of correspondence and aim to minimize it,

$$\arg\min_{\mathbf{T}} \frac{1}{2} \sum_{i=1}^{n} \left\| u_i - \frac{1}{s_i} KTP_i \right\|_2^2 \tag{6}$$

Gauss-Newton iteration on P-n-P

In order to solve the above non-linear least square problem, Gauss-Newton is among the most common. Yet the T matrix inherits the constraint from its embodied rotation matrix R. To ease the constraint, we can write the cost in terms of Euler angles or quaternions, which, however, will either demonstrates singularity (lose of degree of freedom) at certain angles, or carries extra degrees of freedom (more than three) in representing rotations, thus may cast the Gauss-Newton algorithm into ill behaviours.

Then as another option, Lie algebra associated with the matrix Lie group $SE(3)$ can be introduced. Given the large content of its background, we will only briefly mention the essential conclusions that implemented in this project. The step-by-step derivation and motivation are available in [3].

The Lie algebra associated with the matrix Lie group of pose transform is

$$\begin{gathered} se(3) = \{\xi^\wedge \in \mathbb{R}^{4\times4} \,|\, \xi \in \mathbb{R}^6\} \\ \text{where} \quad \xi^\wedge = \begin{bmatrix} \rho \\ \phi \end{bmatrix}^\wedge = \begin{bmatrix} \phi^\wedge & \rho \\ 0^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \\ \phi^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times3} \end{gathered} \tag{7}$$

The elements of $SE(3)$ and $se(3)$ are related by the exponential mapping $\exp(\xi^\wedge)$, which has multiple equivalent closed-form representations. Although it is a surjective mapping, we can limit the magnitude of ϕ that in its essence refers to the angle of the rotation with in $[-\pi, \pi)$, that is, $|\phi| \leq \pi$. Then the bijection between $SE(3)$ and $se(3)$ is set up.

In the Gauss-Newton iteration, we can take the Jacobian of a transform with respect to its Lie algebra vector according to $\frac{\partial Tp}{\partial \xi}$, however it can be challenging to implement (involving the adjoint matrix of T). Alternatively we can carry out the optimization by perturbing with a small transform on the left side according to $T = \exp(\epsilon^\wedge)T_0$, $\epsilon^\wedge \in se(3)$, then the Jacobian with respect to this perturbation (i.e. the left Lie derivative) is simply [3]

$$\frac{\partial(TP)}{\partial \epsilon} = \begin{bmatrix} I & -(TP)^\wedge_{1:3} \\ 0^T & 0^T \end{bmatrix}_{4\times6} := (TP)^\odot \tag{8}$$

Thus, we can formulate the derivative of the error sum (Equation 6) with respect to ϵ. For a clearer form, first we write the error sum as

$$\begin{aligned} G(T) &= \frac{1}{2} \sum_{i=1}^{n} u_m(TP_m)^T u_m(TP_m) \\ &= \frac{1}{2} \sum_{m=1}^{n} \left\| u_i - \frac{1}{s} KTP_m \right\|_2^2 \end{aligned} \tag{9}$$

Then the perturbation ϵ is reflected in each u_m as,

$$\begin{aligned} u_m(TP_m) &= u_m(\exp(\epsilon^\wedge)T_0P_m) \approx u_{ij}(I + (\epsilon^\wedge)T_0P_m) \\ &\approx \underbrace{u_m(T_0P_m)}_{\beta_m \in \mathbb{R}^{2\times1}} + \underbrace{\left.\frac{\partial u_m}{\partial x}\right|_{x=T_0P_m} (T_0P_m)^\odot}_{\delta_m^T \in \mathbb{R}^{2\times6}} \epsilon \end{aligned} \tag{10}$$

Inserting back to $G(T)$ we have,

$$G(T) \approx \sum_{m=1}^{n} (\delta_m^T \epsilon + \beta_m)^T (\delta_m^T \epsilon + \beta_m) \tag{11}$$

It is quadratic in ϵ. The corrsponding derivate is,

$$\frac{\partial G}{\partial \epsilon^T} = \sum_{m=1}^{n} \delta_m(\delta_m^T \epsilon + \beta_m) \tag{12}$$

Thus the Lie algebra vector linked to the ideal perturbation step ϵ^* that minimizes the error sum is,

$$(\sum_{m=1}^{n} \delta_m \delta_m^T)\epsilon^* = -\sum_{m=1}^{n} \delta_m \beta_m \tag{13}$$

With regard to the error sum in P-n-P (Equation 6), let $\begin{bmatrix} X & Y & Z \end{bmatrix}^T = TP_m$ and $\begin{bmatrix} u_{m1} & u_{m2} \end{bmatrix}^T = u_m$ so that we have

$$\delta_m^T = \begin{bmatrix} -\frac{f_x}{Z} & 0 & \frac{f_x X}{Z^2} & f_x \frac{XY}{Z^2} & -f_x - f_x \frac{X^2}{Z^2} & f_x \frac{Y}{Z} \\ 0 & -\frac{f_y}{Z} & \frac{f_y Y}{Z^2} & f_y + f_y \frac{Y^2}{Z^2} & -f_y \frac{XY}{Z^2} & -f_y \frac{X}{Z} \end{bmatrix}_{2\times 6}$$

$$\beta_m = \begin{bmatrix} u_{m1} - f_x \frac{X}{Z} - c_x \\ u_{m2} - f_y \frac{Y}{Z} - c_y \end{bmatrix}_{2\times 1} \tag{14}$$

Now we can present the algorithm of Gauss-Newton optimization with backtracking algorithms to prevent the update step being too long.

Algorithm 1: Gaussian-Newton Iteration

```
Gaussian-Newton( $\{u_m, P_m\}$, $T_0$, $\epsilon_0$ ):
    repeat
        Calculate $\delta_m$ and $\beta_m$ (Equation 14).
        Solve the least square error linear
          equations of $\epsilon$ (Equation 13).
        // Backtracking
        while $G(\exp(\epsilon^\wedge)T_0) > G(T_0) + \alpha \frac{\partial G}{\partial \epsilon^T}\epsilon$ do
            $\epsilon \leftarrow \beta\epsilon$
        $T_0 \leftarrow \exp((\epsilon^*)^\wedge)T_0$
    until $|\epsilon^*| < \epsilon_0$
    return $T$
```

RANSAC

To remove outliers due to wrong data association, RANSAC aims to compute the motion T from randomly sampled sets of data points and verify the hypotheses on the other data points. The hypothesis with highest consensus with others are adopted as the solution [1].

The reprojection error is evaluated with the directional error that measures the angle between the ray of the image feature and the epipolar plane.

$$\frac{\pi}{2} - \arccos\left(\frac{(K^{-1}u)^T t^\wedge (TP)_{1:3}^T}{\|(K^{-1}u)\|_2 \left\|t^\wedge (TP)_{1:3}^T\right\|_2}\right) \tag{15}$$

The number of iterations N is determined as in [1], and recommanded to take tenfold the amount.

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{16}$$

Algorithm 2: RANSAC outlier rejection

```
RANSAC( $s$, $k$, $\epsilon_0$ ):
    repeat
        Randomly select $s$ pairs of correspondence.
        Estimate the transform $T$ upon sampled
          pairs.(Algorithm 1)
        for other pairs do
            Calculate the re-projection angle error
              (Equation 15).
        Store and count inliers that projected to
          an error smaller than threshold $\epsilon_0$.
    until iterated $k$ times
    Estimate $T^*$ with the inlier set that has most
      votes.
    return $T^*$
```

where p is the goal success rate, s is the number of data points to estimate each model and ϵ is the percentage of the outliers.

D. *Pose-graph optimization*

The VO problem can be represented as a graph, where a state to be estimated is a vertex and the constraints among the states are edges that connected to the vertex. In particular, assume we would like to optimize both the robot pose and the triangulated points' positions, all these states are taken as vertexes and the rigid-body transformations between two poses and the location of a landmark w.r.t. the camera are considered as edges.

When the transformations are assumed fixed, the poses, or states, can be optimized to minimize the cost function:

$$\sum_{C} \|C_i - T_{ij}C_j\|_2^2 \tag{17}$$

E. *Windowed bundle adjustment*

When we consider consecutive M frames, a corresponding local graph can be formulated. The total re-projection error among these frames are

$$\begin{aligned} &\underset{\{\mathbf{C_m}\},\{\mathbf{P_k}\}}{\arg\min} \sum_{m=1}^{M}\sum_{k} \|u_{km} - G(C_m, P_k)\|_2^2 \\ &= \underset{\{\mathbf{C_m}\},\{\mathbf{P_k}\}}{\arg\min} \sum_{m=1}^{M}\sum_{k} \left\|u_{km} - \frac{1}{s_k}KC_mP_k\right\|_2^2 \end{aligned} \tag{18}$$

Here u_{km} is the projection of the kth landmark P_k in the mth image. We can also choose to only optimize the camera pose, thus the locations of landmarks are fixed. The initial poses are set as the VO solution obtained from previous sections.

III. Implementation and evaluation setup

The implementation follows the pipeline in Figure 1 and the customized classes and their relations in this project are outlined in Figure 3.

We first identify the basic data type is frame, each associated with a pair of stereo images and a pose of its left camera to be estimated. Feature is considered as another basic type and as in the VO pipeline, a certain amount of features are extracted and triangulated for each frame and matched between consecutive frames. The other defined data type initialize and process on the basis of frames and features.

Within the pipeline, extra interest goes to the Gauss-Newton optimization in order to reach higher clarity. Thus it is implemented upon more basic data types, i.e. $SE(3)$ implemented in `Sophus`, instead of calling `OpenCV` functions .

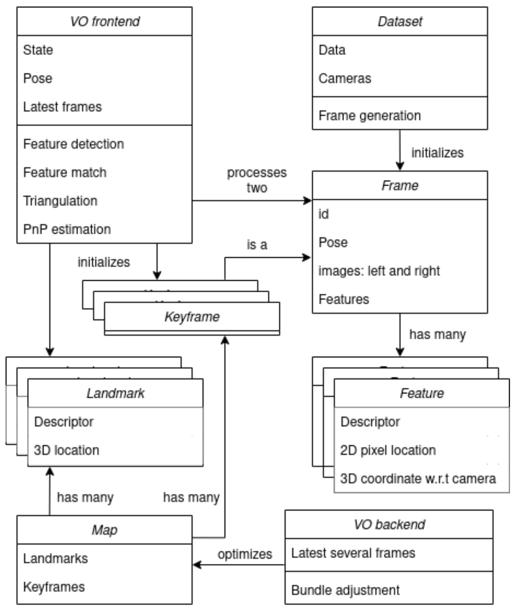

Figure 3: Classes and relations in this project. The dataset prepares the frame in order for the frontend to process, during which new keyframes might be initialized and associated with the existing landmark. The backend or the bundle adjustment process upon the map, and optimizes the locations of its landmarks and the poses of the keyframes.

A. Feature detection and matching

In this project we adopt ORB feature detection and description due to its computational efficiency. It is implemented in `OpenCV`. As mentioned, for each frame we detect and match features from the left and right image. For feature pairing, brutal force matching based on Hamming distance that implemented in `OpenCV` is selected.

For consecutive frames, only features with the triangulated depth are matched with those in the new frame. Since not all the previous features are matched. The pairing procedure is the same as in processing the left and right image, therefore it is not considered as feature tracking, but still feature matching.

B. State estimation

As proposed in Equation 4, triangulation is implemented to solve the least square error linear equations. Features with estimated depth that out of the bound, that is, negative or too large are considered as outliers therefore not passed to the following steps.

In P-n-P, the re-projection error is based on the transform that convert the feature points in the previous frame to the image plane of the current frame, not the opposite. Then the Gauss-Newton optimization integrated with RANSAC are directly implemented from the algorithms in [3] and [1] on the top of `Sophus`. The initial guess is set as the identity transform.

C. Bundle adjustment

In order to reduce the computation only one frame in certain period is chosen and it should be associated with sufficient amount of unknown features missed in the previous keyframes. The triangulated points in the key frames are also referred to as landmarks and they are supposes be observed in multiple frames. `g2o` is known for its performance and flexibility, thus proposed to be integrated in this project. However, it will be left as a future work, since several structural changes in the project have to be made to accommodate the bundle adjustment, such as implementing the keyframe initialization and the landmark object.

D. Evaluation and visualization

The absolute trajectory error depicts the difference between two sets of poses, which actually computes the root-mean-squared error (RMSE) of the Lie algebra of the poses. Assume N pairs of poses are available, the error is computed as

$$e = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\log\left(T_1^{-1}T_2\right)^{\vee}\right\|_2^2} \tag{19}$$

However, as the elements supposed to belong to rotation matrices may not fulfil the orthogonal requirement, due to rounded floating point etc, the translational trajectory error is adopted instead,

$$e = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|t_1 - t_2\|_2^2} \tag{20}$$

To visualize the result, the estimated trajectory is plotted against the ground truth in `gnuplot` in two dimensions to display the horizontal motion.

In the Kitti dataset setup, two gray scale cameras (Cam 0 and Cam 1 in Figure 4) are mounted on the top of the vehicle, aligned to the same orientation and kept 0.54 meters apart from each other. The ground truth is measured with RTK GPS that acquires the centimetre level precision.

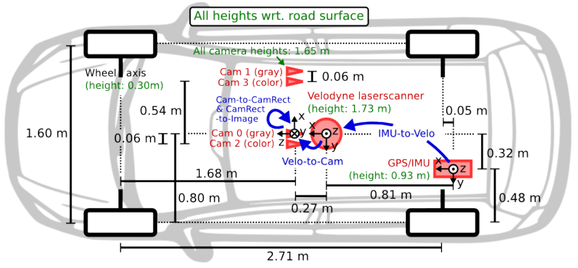

Figure 4: Sensor setup in Kitti stereo dataset [5].

IV. Results and analysis

The performance is evaluated on Kitti dataset 00 and 04. Kitti 00 consists of more than four thousands frames and the orientation of the vehicle changes significantly during the course, while Kitti 04 is much shorter and simpler that only contains more than two hundred frames that captured along an almost straight course.

The proof of concept is done upon the first ten frames on Kitti 00 while the drifting is significant to lead the estimation off the track after several hundred meters (Figure 5) . Thus to maintain the evaluation conceivable, we switch to Kitti 04 as a easier setup for the moment.

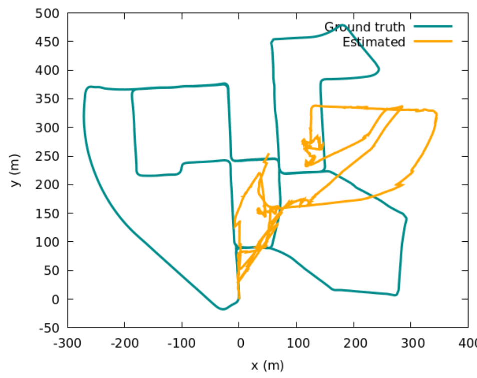

Figure 5: Estimated trajectories without RANSAC on Kitti 00. The vehicle starts at the lower corner and the drift became significant after several hundred meters.

A. *Feature detection and matching*

With in one frame, 500 features are matched between the left and right image. Around 120 features fall within the current Hamming distance threshold and are passed to the matching between two consecutive frames. Then around 60 features can be matched with those in the new frame. Figure 6 demonstrates the matched pairs on the left and right image, while matching results among the consecutive frames are similar, considering the transform between camera poses are of similar scale, given that the frame rate and vehicle velocity.

No mechanism is implemented to evaluated the matching, however at a rough glance (for example in Figure 6), the majority of the raw matches are credible. Checking one by one the better matches in a few consecutive frames, nearly all of them are correct. Since the matched features will be further examined in the RANSAC and the outliers will be suppressed, we consider the matching result sufficiently reliable.

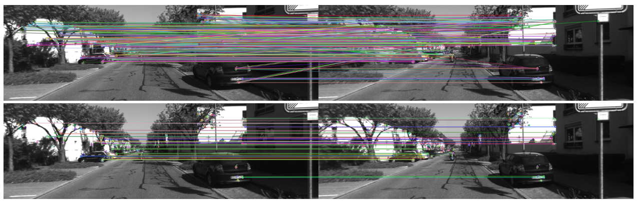

Figure 6: Detected and matched feature points from the left and right image in the same frame. All matches are examined by the similarity and only sufficiently similar matches are kept as good matches that shown in the lower image pair.

The amount of features are sufficient to draw estimation. However, if more features should be generated, we can either loosen the threshold or change feature matching to tracking, thus more triangulated points can be matched, nevertheless at the trade-off of increasing outliers. Further we can changed the triangulation method from current solving least square linear equation to horizontal epipolar line searching, as the two cameras are carefully aligned to the same orientation and of the same depth 0 in each other's fixed frame. Therefore all the matched points between two frames can be associated with a depth. In that case, the triangulation will be carried out after consecutive frames are matched.

The triangulation accuracy is not evaluated, however it should be carried as a future work, possibly making use of the LiDAR measurement of the Kitti dataset. The pose of the LiDAR is displayed in Figure 4. As mentioned, a few features with negative or too large estimated depth are rejected as mismatches, which is likely the cause.

B. *Motion estimation*

The estimation on Kitti 04 is plotted against the ground truth in Figure 7. We can see the offset in each step adds up to the overall drift from the course. Especially the drift in orientation has a large influence in the overall performance. To measure the effectiveness of RANSAC, we present separately the results when RANSAC is used or not and we can see RANSAC does enhance the performance by ruling out certain features.

	without RANSAC	with RANSAC
RMSE (m)	17.6	10.4
Final offset (m)	40.3	15.5

Table 1: RMSE and final offset for Kitti dataset 04, which is a straight course around 400 meters long.

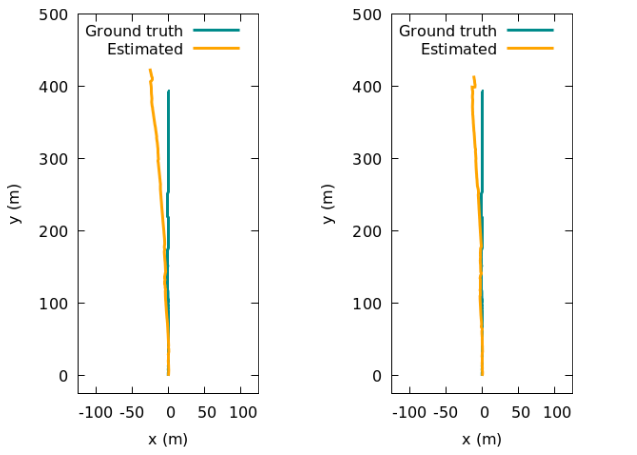

Figure 7: Estimated trajectories without RANSAC (left) and with RANSAC (right). The one processed by RANSAC has a higher precision.

In the detailed examination of the each iteration, we find the estimation converges before the set max iteration times of 60 is reached. Also, at the end of the RANSAC, the average re-projection error is around 1, that is, the projected pixel location almost coincides with its actual location, thus the optimization is considered performing well, especially given that the resolution of the image is relatively high (1241×376). Thus the offset is mainly introduced by remaining mismatches or inaccurate triangulation.

However, one source of the outlier could be the fellow vehicles, i.e. the dynamic objects along the camera. Since visual odometry in this project assumes a static environment and estimates the relative motion between them, the motion of the environmental object won't be estimated or considered in the camera pose estimation. Therefore an offset as large as the environmental object's motion is introduced during triangulation. However, if only a small portion of features come from the dynamic objects, the estimation that involves them would drift from those based on others, thus those features will be considered as outliers.

The above result is obtained with 7 pairs of correspondences for each iteration in RANSAC, which is not the minimal set mentioned in [1]. We observes in the experiment that when it is set to less pairs, the performance degrades, possibly due to the noise carried in the small set of pairs, which are averaged, thus less significant in a larger set. Nevertheless, estimation based on less pairs indeed eases the computation. On the other hand, increasing the amounts of pairs is not always beneficial, since it could also let in more noise.

C. *Miscellanea*

To get an idea of the load of computation and the desired real-time processing, we time each frame. Without RANSAC, it takes less than 0.1 second and with RANSAC, ranging from 0.1 to 0.8 second. The frame rate of the Kitti dataset is around 10 frame per second. Thus without RANSAC the performance is nearly real-time, but clearly it should be optimized.

As proposed, the initial guess to start the Gauss-Newton iteration is the identity transform. Motion estimated by other sensors, such as wheel odometry could be used to predict an initial value that closer to the solution, so that the times of iterations before convergence can be reduced.

D. *Future work*

Regrading the existing issus and conceivably desired functions, the project could benefit from

- Improving the implemented functions:
  - Evaluate the triangulation precision.
  - Investigate causes of sudden undesired changes in the estimation.
- Implementing and integrating with other established methods:
  - Bundle adjustment based on `g2o`.
  - Bag-of-word loop closure detection and pose-graph optimization based on `g2o`.
  - Keyframe maintenance in order to save the storage.

V. Conclusion

In this project, the implemented feature based visual odometry algorithm managed to estimated the motion of a stereo camera in an autonomous driving setup, with a special interest in the optimization on the $SE(3)$ group. However the precision need to be enhanced and multiple more feature based methods can be implemented and compared with appearance based methods. The Kitti dataset has provided a versatile benchmark that worths further exploitation. The main remaining components to be implemented are bundle adjustment and loop-closure detection and the development of this project will continue.

References

[1] D. Scaramuzza and F. Fraundorfer, “Visual odometry [tutorial],” *IEEE Robotics Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[2] X. Gao and T. Zhang, *14 Lectures on Visual SLAM: From Theory to Practice*. Publishing House of Electronics Industry, 2017.

[3] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[4] R. Szeliski, *Computer Vision: Algorithms and Applications*. Berlin, Heidelberg: Springer-Verlag, 1st ed., 2010.

[5] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.