Feature based Monocular Visual Odometry for Autonomous Driving and Hyperparameter Tuning to Improve Trajectory Estimation

Ziyang Zheng*

1 The Experimental High School Attached to Beijing Normal University, Beijing, China

*Corresponding author’s e-mail: ziyangzheng011025@gmail.com

Abstract. In this paper, we will present an efficient feature-based localization method and use hyperparameter tuning to improve the accuracy of the method. We will first address the advantages of vision-based localization algorithms. These algorithms are more cost-efficient without GPS and are easier to integrate with vision based perception algorithms such as object detection and semantic segmentation. With camera equipped on cars, visual simultaneous localization and mapping, also called visual SLAM, improves many applications such as autonomous driving and advanced driver-assistance systems (ADAS). Currently, there are many different Visual SLAM algorithms and we will analyse the advantages of featured-based sparse methods. In the paper, we will briefly talk about the pipeline of our proposed method including feature extraction, key points matching, 8-point RANSAC algorithm and single value decomposition (SVD). Then we will use the grid search method to tune the parameters in the algorithm and improve the accuracy of the featured based visual localization method.

1. Introduction
Visual simultaneous localization and mapping [1], also called Visual SLAM, is one of the most important topics discussed in computer vision society. It is a method to determine an object's positions which forms its trajectory, and build the ambient 3D environment, which is known as mapping, without using GPS signal. Its applications include navigation systems in self-driving and augmented reality, virtual reality, drones, robotics, etc.

Visual SLAM was first used to determine robots' positions when they move in a unknown environment. The translations and rotations of the robots can be computed using methods in multi-view geometry. Theories in multi-view geometry contribute to the foundation of visual SLAM research. With the epipolar geometry theories, we can recover the camera position changes using fundamental matrices. Solving the fundamental matrices requires the singular value decomposition (SVD) in linear algebra.

One of the advantages of Visual SLAM algorithms is lower cost because they do not require expensive LIDAR or GPS equipment. People just need to capture pictures from camera. For example, in autonomous driving, they can integrate the information recorded by multiple cameras and improve the accuracy of localization and vision algorithms.

Due to the rising demands of faster and more accurate localization methods in driver-less cars and ADAS, visual SLAM algorithms have been improved by researchers in computer vision. The
optimization techniques include loop closure that reduces the drift and key frame selection, by which improving the accuracy and speed of visual SLAM algorithms.

Scientists found that through detecting feature points and extracting descriptors from these feature points, visual SLAM could be more invariant to the surrounding environments such as changes in illumination and scale. Specifically, feature-based methods will improve localization accuracy in environments which have illumination variation. Among the feature-based methods, sparse methods, in which we only need to extract and match a few key points, are faster to compute.

Extracting ORB feature using oriented FAST feature detector and rotational BRIEF feature descriptor is an efficient method to find key points in each frame and match these key points. The original FAST detector doesn't detect the orientations and the original BRIEF don't describe rotation features. By adding special functions to a conventional method, people are more efficient in SLAM algorithms.

2. Related works

2.1. Feature based methods
To match points and objects between images, detecting and describing key points is a time-saving method because we only need a few points in the corners or edges in each frame. The principle of these sparse, indirect methods is to minimize the projection error after we recover the camera translation and rotation.

There are a lot of feature detection and description tools. For extracting feature points, many of existing methods such as Harris corner detector and SIFT detector require intensity gradients to identify key points. The requirement of intensity gradients is time-consuming. With FAST feature detector, computers can determine the key points without much calculation. For feature description, while some methods such as SIFT feature descriptor are accurate in transforming the information in the image to several intensity gradients vectors, they are not fast enough to compute in the real world. Oriented FAST and rotational BRIEF make ORB feature one of the best ways to implement visual localization and mapping.

The early application of visual SLAM using ORB features is Parallel Tracking and Mapping (PTAM) [2]. This algorithm uses effective methods of feature matching, triangulation and localization but is less accurate in important aspects such as relocalization, map initialization and loop closure. The enhanced method is ORB SLAM, which solves problems in map initialization, relocalization and loop closure.

2.2. Direct methods
Unlike indirect methods that only extract a fraction of information of the images, direct methods take advantage of all the pixels presented in images. The principle of direct methods is to minimize the photometric error between each frame. An early example of direct method is DTAM (Dense Tracking and Mapping) [3], which provides a dense model to find the accurate poses of camera and map the surroundings. Later, scale-aware algorithms such as LSD [4] were introduced to further enhance the accuracy of direct methods. Then, Direct Sparse Odometry (DSO) [5] was introduced to provide better application of points and frames in augmented reality. While the direct methods often require taking every pixel in the images into consideration, the computation time can be reduced using optimizations. The Semi-direct Visual Odometry (SVO) [6] finds its balance between indirect methods and direct methods. During the process of SVO, people use feature correspondence but not feature extractions and matching, meaning that people select several small pixel patches and measuring the photometric errors between patches. Feature extraction is applied only the current frame is a key frame.

2.3. Deep learning based methods
With the rise of deep learning algorithms recently, deep learning methods have been applied in visual SLAM algorithms. Many of the deep learning methods require no computer vision and two-view
geometry knowledge. GoogleNet is a perfect example of convolutional neural network which regresses the camera pose through training. In 2015, researchers from University of Cambridge use a 23-layer convolutional neural network similar to GoogleNet and train computer to learn translation and rotation simultaneously [7]. Their localization was proved to be accurate when it was tested under 5 scenes in University of Cambridge. Moreover, in 2017, researchers from UC Berkeley and Google developed a synthesis neural network which includes two independent unsupervised training networks [8], single-view depth network and multi-view pose estimation network. Computer can recognize the surrounding environment by making inference from previous pictures.

3. Theoretical approach
In this section, we will demonstrate the pipeline of our feature-based approach by explaining the pinhole camera models used on moving vehicles. We use epipolar geometry methods to recover the camera poses, feature detection and description methods techniques to match key information in images and singular value decomposition to solve equations.

3.1. Pinhole camera model
In visual SLAM, people use pinhole cameras to record the environments around us. There is a predetermined world coordinate at the beginning, and the coordinate relative to the camera is constantly changing. Denote a three-dimensional point relative to the world coordinate as \( X_w \) and the same three-dimensional point relative to the camera coordinate as \( X_c \). The harmonic forms of the world coordinate and the camera coordinate are \( \tilde{X}_w \) and \( \tilde{X}_c \) respectively. The relationship between the world coordinate and the camera coordinate can be expressed by the equation:

\[
\tilde{X}_w = T \ast \tilde{X}_c, T = \left[ \begin{array}{ccc} R & 0^T & 1 \\ \end{array} \right] \tag{1}
\]

In this equation, \( T \) is a 4*4 matrix representing the rotation \( R \) and the translation \( t \) of the camera. Matrix \( T \) is also called “the extrinsic matrix of the camera”.

In the camera, the coordinates are on a plane, so they should be projected to the three-dimensional space. The harmonic coordinate of the point projected on a 2D image is denoted by \( \tilde{x}_i^m \), and the relationship between the 3D camera coordinate and the 2D image coordinate of the same point can be expressed by the following equation:

\[
\tilde{x}_i^m = K \ast \tilde{X}_c, K = \left[ \begin{array}{ccc} f_x & 0 & c_x \\ 0 & f_y & c_y \\ y & 0 & 1 \\ \end{array} \right] \tag{2}
\]

In this equation, \( f_x \) is the focus length of the pinhole camera on the x-axis and \( f_y \) is the focus lengths on the y-axis. The principle point of the plane can be expressed by a 2D coordinate \( [c_x, c_y] \). The matrix \( K \) is also called “the intrinsic matrix”.

With the pinhole camera model, we can do mathematical calculations to find the intrinsic and extrinsic matrix of the camera and build the mathematical relationships between world coordinates, camera coordinates and the image coordinates. The calculation to solve the matrix will be introduced in section 3.2

3.2. Two-view geometry
In order to draw the trajectories of cars and other objects, I need to compare the differences between pairs of 2D coordinates in consecutive images. For a single point on a particular object, the only information gathered from a camera is the 2D coordinates indicating the position of this point on corresponding planar images. To reveal the relation between the pairs of 2D coordinates in consecutive frames, epipolar geometry is required.

Suppose there are two consecutive images, the 2D coordinate in the first image is \( \tilde{x}_i^m \) and the matching coordinate in the second image is \( \tilde{x}_{i^m}^m \). The intrinsic matrices in these two images are \( K \) and
The pose change between the two images is $T = \begin{bmatrix} R & 0 \newline 0 & 1 \end{bmatrix}$. The following equations show the relation between the two 2D coordinates.

\begin{align*}
\tilde{X}' &= T \ast \tilde{X} \\
X' &= R \ast \tilde{X} + t \\
\tilde{x}_{im} &= K \ast X \\
\tilde{x}'_{im} &= K' \ast X'
\end{align*}

By using matrix calculations, we can get the final equation:

\begin{align*}
X'_{im}^T F \tilde{x}_{im} &= 0 \\
F &= (K'^{-1})^T E K^{-1} \\
E &= t \times R
\end{align*}

$F$ is called the fundamental matrix and $E$ is called the essential matrix. Knowing the essential matrix, we can then recover the camera poses.

However, in real-time circumstances, the corresponding points in two consecutive frames $\tilde{x}'_{im}$ and $\tilde{x}_{im}$ may not satisfy the equation $X'_{im}^T F \tilde{x}_{im} = 0$ due to small errors. The value of $X'_{im}^T F \tilde{x}_{im}$ will be approach 0. Therefore, in the real-time circumstances, we will choose the fundamental matrix $F$ with the smallest value of $X'_{im}^T F \tilde{x}_{im}$ if $\tilde{x}'_{im}$ and $\tilde{x}_{im}$ are corresponding points in consecutive frames, and that is where we need to singular value decomposition.

To apply singular value decomposition, the definitions of eigen values and eigen vectors are necessary. Given a $n \times n$ square matrix $A$, we call $\lambda$ the eigenvalue of $A$ and $x$ the corresponding eigen vector if $A x = \lambda x$. For every square matrix $A$, it can be expressed into the following form:

\begin{equation}
A = Q^T \Lambda Q
\end{equation}

$\Lambda$ is the diagonal matrix with all the eigen values of matrix $A$ and the minimum of $A$ equals to the minimum eigen value of $A$. Then we can calculate the minimum value for $X'_{im}^T F \tilde{x}_{im}$ by singular value decomposition.

### 3.3. ORB feature

When comparing two images, extracting and matching the key points in these two images can be efficient than comparing raw pixel intensities. Key points may be on the edge or corners of the objects in the pictures. For example, if people want to identify a mountain, the best way to describe the mountain is to mark the peaks of the mountains. In my proposed visual odometry methods, I need to extract a few key points from the consecutive frames in order to recover the camera poses using epipolar geometry. The rest of this section will be the introductions to oriented FAST feature detector and rotational BRIEF feature descriptor. The parameters in the ORB feature detection and description are also introduced for further experiments in section 4.

#### 3.3.1. oFAST detector

In a picture, when we want to find a set of sparse points to represent the whole picture, points that have large intensity gradients will be chosen because these points are more likely to represent the edges or corners. While most of the current methods such as The Harris corner detector applies this method by computing the intensity difference when moving from a point $(x, y)$ to $(x + u, y + v)$, which can be represented the equation $E(u, v) = \sum_{x,y} w(x,y) [I(x + u, y + v) - I(x, y)]^2$, it is time-consuming to calculate the intensity gradients. Another problem is that the corners in the image will not yield large $E(u, v)$ if the image is magnified. Therefore, we need to find a method to detect key points without calculating the intensity gradients and worrying about the scale variations. The FAST feature detector satisfies our requirements with the following steps:

- Select a pixel $p$ in the picture with an intensity $I_p$, and choose a threshold $t$, which is denoted as “fast threshold” in ORB parameters.
• Find the 16 surrounding points in a circle.
• Find if there are consecutive n points whose intensities are larger than \( I_p + t \) or smaller than \( I_p - t \). If there is a point set satisfy the requirement, the point \( p \) is a corner in the picture.
• In order to deal with scale variances, we resize the original images using scale factors and find key points again in the resized images. After we have done resizing several times, we can build a pyramid model for key points in different scales. The scale factor is denoted as “scale” and the number of resized images is denoted as “nlevels” in ORB parameters.

Since the original FAST feature detector do not indicate the direction of the corner, we will can use the “Intensity Centroid” to determine the directions of corners in the picture. Generally, we define a moment of a image patch by using the following equation:

\[
m_{pq} = \sum_{x,y} x^p y^q I(x,y)
\]

And the direction of corner can be calculated by the equation \( \theta = \arctan(m_{01},m_{10}) \).

3.3.2. rBRIEF descriptor
After selecting feature points, the next task is to match the key points we have found. The BRIEF (Binary Robust Independent Elementary Features) descriptor outperforms other feature descriptors in computing speeds. We set a binary intensity test \( \tau \) with point pairs \((x,y)\) in the image patch \( p \) whose size is denoted as “patch size”:

\[
\tau(p; x,y) = \begin{cases} 
1, & p(x) < p(y) \\
0, & p(x) \geq p(y)
\end{cases}
\]

By using this test, key points detected are recorded by binary numbers using the following function:

\[
f_n(p) = \sum_{i=1}^{n} 2^{i-1} \tau(p; x_i, y_i)
\]

However, in the BRIEF descriptor, the rotations of camera make it hard to match key points accurately [9]. If the original patch is \( S \), rotating angle is \( \theta \), and the patch after rotation is \( S_{\theta} \), the feature in the image after rotation can be expressed in the following equation:

\[
g_n(p, \theta) = f_n(p) \mid (x_i, y_i) \in s_{\theta}
\]

When describing feature points, the next step is to compare the minimum hamming distance and find possible matches between consecutive points with brute force. The complexity of matching is \( O(n^2) \) which depends on the number of pair tests chosen in the BRIEF feature descriptor.

With oriented FAST feature detector and rotated BRIEF feature descriptor, we can compute much faster while not losing important information-the orientation of the corners.

3.4. RANSAC algorithm
For each pair of consecutive images, we need to recover the translation and rotation using features in pictures. In this section, we will introduce the detailed procedure to choose feature points using RANSAC algorithm and recover the poses of cameras with all the methods mentioned in the previous sections.

RANSAC algorithm is a method to deal with data with a lot of outliers. These outliers will significantly reduce the accuracy of prediction if we take them into consideration. Usually, we find several points as a sample and set a range to determine whether the other points are inliers or outliers according to this sample. After finding a lot of samples, we choose the best sample with the most inliers as the final solution.

In our proposed visual odometry methods, we use 8-point RANSAC algorithm. 8 feature points are selected in each of the two frames and then we find the best set of points that most accurately describe the translation and rotation between the two frames. According to epipolar geometry, for corresponding points in two consecutive frames \( x' \) and \( x \),

\[
x'^T F \bar{x} = 0
\]

However, in the reality, the right-hand side of the equation will not be zero because of measurement errors. Therefore, the best set of points that most accurately describe the translation and rotation has the minimum value of \( x'^T F \bar{x} \),which can be solved using the single value decomposition
method in the previous section. After we get the minimum value of the terms \(x^T F \bar{x}\), we can recover the fundamental \(F\) and essential matrix \(E\). Then we can get the poses of camera.

4. Experimental results
After designing my visual odometry method, we conduct experiments on KITTI dataset, a large-scale outdoor environment dataset which includes fast translations and rotations. The fast rotations and translations of this dataset make it difficult for us to yield accurate examples in our experiments. We choose sequence 01-10 that include a various pattern of trajectories include lines, curves, and loops.

4.1. Implementing proposed visual odometry
Initially, we extract 4000 feature points from each image frame and use RANSAC to find the best solution. The result is shown in figure 1. The accuracy of the prediction is not satisfactory, and we need to find a way to improve the accuracy of the experiment.
The first thought we get to improve the accuracy of our results is to increase the sample size from each frame, i.e. the number of features points extracted from each frame. We increase 1000 feature points at each time from 4000 to 11000. The following experiments took hours to finish and the results of are shown in table 1, which records the average ATE between our predicted trajectory and ground truth position.

**Table 1. Average ATE in Sequence 01-10 with Different Numbers of Features.**

| sequence | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10000 | 11000 |
|----------|------|------|------|------|------|------|-------|-------|
| 1        | 191.45 | 296.33 | 100.08 | **72.81** | 193.36 | 118.94 | 77.66 | 105.86 |
| 2        | 218.84 | 172.42 | 173.39 | 237.64 | 176.63 | 134.15 | **92.93** | 106.4 |
| 3        | 33.9   | 30.85 | 18.2  | 36.05 | **14.14** | 14.57 | 16.21 | 24.54 |
| 4        | **4.57** | 4.96 | 5.44 | 5.5  | 4.97 | **4.57** | 5.51  | 4.82 |
As shown in figure 3 and table 1, the average ATE decreases significantly as we increase the size of feature points set except for sequence 07 and 08. However, the average ATE still fluctuates when the size of feature points set increases to 11000 in some sequences. The minimum average ATE for each sequence is marked bold in the table. The best trajectories predicted are shown in figure 2, and the plot of average ATE in each sequence when extracting different amounts of feature points is shown in figure 3.
Figure 2. The Best Trajectory Predicted

Figure 3. Average ATE Plot
Unlike the ORB SLAM [7], our visual odometry method doesn't use optimizations such as key frame selections and loop closures. We try to minimize the estimation error of trajectories by improving the ORB features. Although the result may not seem accurate, we can find how to combine the best ORB feature parameters and improve visual localization method. The most important ORB features are number of features, scales and number of levels. The other influencing parameters taken into consideration in section 4.2 are “fast threshold” and “patch size” mentioned in section 3.

4.2. Parameter tuning
In this part, we will explain our hyperparameter tuning experiment and try to find the best sets of hyper parameters for accurate trajectory estimation. The method we use in the experiment is grid search, which means that we make steady increments in all the five parameters and try to find the best parameter combinations.

Firstly, we need to run the ORB feature based algorithms with default hyperparameter in each sequence that is mentioned in the previous part. As shown in table 2, the accuracy of the predictions is far from what we expect.

| Sequence | Average ATE |
|----------|-------------|
| 1        | 418.45      |
| 2        | 459.15      |
| 3        | 42.06       |
| 4        | 4.81        |
| 5        | 192.46      |
| 6        | 144.4       |
| 7        | 63.22       |
| 8        | 147.54      |
| 9        | 77.25       |
| 10       | 76.6        |

Next, we are going to run grid search on sequence 01, 03, 04, 05, 06, 07, 09, and 10, and use sequence 02 and 08 to test the best parameters yielded from grid search. The following five parameters are adjusted and tested:

- The number of features points ranges from 1000 to 9000 with an increment of 2000.
- The scale factor ranges from 1.1 to 1.9 with an increment of 0.4.
- The number of levels ranges from 4 to 12 with an increment of 4.
- The patch size value is chosen at 16, 31, and 47.
- The fast threshold is chosen at 10, 20, and 30.

The results in this experiment will be compared to our previous results in section 4.1. The minimum average ATE values we have found are presented in table 3. From the result, we can see a significant decrease in average ATE compared to table 1 and table 2, especially in sequence 01, 03, 06 and 07. After we change the single parameter that indicates the number of features extracted in each sequence, we can reduce the average ATE values to a relatively small one. Furthermore, when we tune the five parameters using grid search, we can also significantly reduce the average ATE values. However, we discover that the minimum average ATE values have different corresponding hyperparameter in different sequences. Therefore, in our further experiments, we set out from the minimum average ATEs and search for a better combination of parameters in all the 10 sequences. The best results occur when we choose the hyperparameter in which the number of features is 9000, scale factor is 1.5, number of levels is 4, patch size is 15, and fast threshold is 30. Additionally, there are some other hyperparameters that yield the similar results.
Table 3. Average ATE for Each Sequence in the Best Parameter.

| Sequence | number of features | scale | number of levels | Patch size | fast threshold | minimum average ATE |
|----------|-------------------|-------|------------------|------------|----------------|---------------------|
| 1        | 5000              | 1.1   | 8                | 31         | 10             | 69.31               |
| 3        | 9000              | 1.5   | 12               | 15         | 10             | 10.03               |
| 4        | 1000              | 1.5   | 8                | 15         | 30             | 4.47                |
| 5        | 9000              | 1.5   | 8                | 15         | 10             | 54.99               |
| 6        | 9000              | 1.1   | 8                | 47         | 30             | 27.41               |
| 7        | 9000              | 1.9   | 4                | 31         | 20             | 26.45               |
| 9        | 5000              | 1.5   | 8                | 15         | 20             | 22.6                |
| 10       | 9000              | 1.1   | 12               | 15         | 20             | 23.45               |

After choosing the parameters, we test it in sequence 02 and 08 and results in all the ten sequences are presented in figure 4. From the result, it is feasible to find some hyperparameters that performed accurately in all the sequences.
5. Conclusion

In this paper, we provide a new visual odometry method based on ORB features. The method involves the application of epipolar geometry, ORB features and RANSAC algorithm.

To improve the accuracy of my algorithm, we conduct experiments on different combinations of ORB parameters. We can find the best combination of parameters to estimate the trajectory in each sequence. However, some of the best hyperparameters in each sequence are not suitable for other sequences. Therefore, we also find the hyperparameter that has the relatively accurate prediction for all the 10 sequences.

Further improvements on our algorithm may be more accurate and efficient methods to tune the parameters. In addition, the speed of my algorithm may not be fast enough to use in the real time, especially when we increase the number of feature points.

References

[1] Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics, 31(5), 1147-1163.
[2] Klein, G., & Murray, D. (2007, November). Parallel tracking and mapping for small AR workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (pp. 1-10). IEEE Computer Society.
[3] Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2011, November). DTAM: Dense tracking and mapping in real-time. In 2011 international conference on computer vision (pp. 2320-2327). IEEE.
[4] Engel, J., Schöps, T., & Cremers, D. (2014, September). LSD-SLAM: Large-scale direct monocular SLAM. In European conference on computer vision (pp. 834-849). Springer, Cham.

[5] Engel, J., Koltun, V., & Cremers, D. (2017). Direct sparse odometry. IEEE transactions on pattern analysis and machine intelligence, 40(3), 611-625.

[6] Forster, C., Pizzoli, M., & Scaramuzza, D. (2014, May). SVO: Fast semi-direct monocular visual odometry. In 2014 IEEE international conference on robotics and automation (ICRA) (pp. 15-22). IEEE.

[7] Kendall, A., Grimes, M., & Cipolla, R. (2015). Convolutional networks for real-time 6-DOF camera relocalization. CoRR abs/1505.07427 (2015).

[8] Zhou, T., Brown, M., Snavely, N., & Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1851-1858).

[9] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. R. (2011, November). ORB: An efficient alternative to SIFT or SURF. In ICCV (Vol. 11, No. 1, p. 2).