第一页为封面页

参赛队员姓名：饶适

中学：人大附中朝阳分校

省份：北京

国家/地区：北京市

指导教师姓名：孙东旭

论文题目：SuperVO: A Monocular Visual Odometry based on Learned feature matching

2020

第二页为创新性申明

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员：饶适　　指导老师：孙贵一旭

2020年9月14日

Contents

SuperVO : A Monocular Visual Odometry based on Learned Feature Matching with GNN

Shi Rao
RDFZ Chaoyang Branch School

Abstract

In this paper, we propose a novel Visual Odometry (VO) system using a feature detector and feature matcher based on neural networks. The networks for feature detectors and descriptors learning consists of a conventional CNN for feature detection and description, and a graph neural network (GNN) final feature matching. The learned feature has several advantages over traditional handcrafted features such as being robust to light variation, scale and etc. By applying state-of-the-art deep learning-based feature matcher-SuperGlue, we developed a new monocular VO framework which can exploit the advantages of deep learning-based feature detector and matcher, which performs better than many other learning-based VO methods.
keywords: Visual SLAM, Visual Odometry, Deep learning-based Feature Detector and Descriptor, Graph Neural Network

1 Introduction

Visual Odometry is the problem to localize the agent based on the change in the perspective. Visual Odometry has many applications such as helping self-driving cars localize in urban environments and navigating to the destinations. It helps the user to interact with the virtual objects and interfaces in Augmented Reality (AR) devices, and assisting autonomous robots finishing their jobs in various environments. Visual SLAM is very similar to the visual odometry, SLAM is about localize the agent while building a 3D map of the environment. SLAM is the work that adds reconstruct the 3D environment on top of the visual odometry. In other words, an accurate Visual Odometry would be the foundation of the SLAM.

There are several methods to solve the visual SLAM and odometry problems. These methods can mainly be classified into the following categories: Feature-based method [1] [2] [3], direct methods [4] [5] [6] and more recently, deep learning-based methods [7] [8].

Feature-based SLAM/VO first find interesting points from one image frame. Then a set of feature descriptors are extracted in order to match with other frames. Once two frames are matching, one can compute the relative motion and 3D structures. Direct methods, however, do not involve any feature detection or matching process. Instead, direct methods compute the camera motion and 3D point clouds' coordinates with just raw image pixels. Deep learning-based SLAM/VO is a brand new subject recently emerged. There are three major methods in deep learning-based SLAM, which are supervised approach, self-supervised approach and hybrid approach (combine traditional methods and learning-based method). Our method can be viewed as one type of hybrid methods since we combine traditional VO with the deep learning-based feature.

Researchers have developed many feature detectors and descriptors using traditional methods such as SIFT (Scale-Invariant Feature Transform)[9], SURF (Speeded up Robust Features) [10] and ORB (Oriented FAST and Rotated Brief) features[11].

SIFT feature is one of the most successful and widely used features. The major components of SIFT are a gradient-based feature detector and a histogram-based descriptor. Although SIFT is robust to scale variation, it is slow to compute, especially when the computing power is restricted. SURF was invented to tackle the computing efficiency problem of SIFT. ORB is a binary feature which consists of an oriented version of FAST [12] and a rotated version of BRIEF [13] feature.

Feature-based SLAM methods, as mentioned above, has several drawbacks, such as not being robust to light variation, invariant to scale and rotation. Moreover, 3D reconstructions from many feature-based SLAM algorithms tend to be too sparse to be applied in real life scenarios, such as self-driving auto mobiles, robotics and reality.

To overcome those issues, we apply modern deep learning-based feature detector and matcher to the traditional feature-based SLAM framework.

2 Related Works

2.1 Feature-Based SLAM/VO

Traditional feature-based SLAM such as Mono-SLAM [1], PTAM [2] and ORB-SLAM [3] estimate camera pose and 3D points' coordinates-based on the handcrafted features.

Mono-SLAM is one of the very early algorithms to recover the 3D trajectory of a monocular camera, and it runs in real-time. The main contribution of this work was to introduce an active approach to mapping and measurement, use of general motion model for smooth camera movement, solutions for monocular feature initialization and feature orientation estimation.

Parallel Tracking and Mapping for Small AR Workspaces (PTAM) is another feature-based method. Its main difference with other SLAM algorithm is that it splits tracking and mapping into two separate tasks. By running the program on a dual-core computer, it can track thousands of landmarks at framerate. One of the problems of the algorithm is its dependence on corner features. A rapid movement of the camera could result in a lost of corner feature and tracking failure.

The most wildly used feature-based SLAM is ORB-SLAM. The algorithm uses ORB as feature detector and descriptors, then apply features for all four stages, including tracking, mapping, relocalization, and loop closing. The ORB-SLAM result the state-of-the-art performance when it was published. One of the improvements could be made about the ORB-SLAM is to make the sparse map denser.

2.2 Learning-Based Feature

With the rapid development of deep learning, using deep networks to learn feature detectors and descriptors from the image is becoming more and more popular. Many learning-based features have numerous advantages over traditional features. As a consequence, we now see a trend that learning-based feature are replacing traditional features.

LIFT[14] if an algorithm that uses CNN to extract feature points. The algorithm uses three different networks for the detector, the orientation estimator, and the descriptor. Compare to the traditional features like SIFT, LIFT could offer more correct matches.

Unlike LIFT, which requires supervision for training, SuperPoint is a self-supervised interest point detection algorithm. It computes interest points and descriptors in a single network in real-time and generates more dense correct matches, which is an improvement compare to the LIFT.

SuperGlue is a feature point matching algorithm based on Graph Neural Networks (GNN). It uses SuperPoint as feature detector and descriptors, and it uses GNN for feature matching. As a result, the whole detection pipeline has no traditional method involved.

3 Methods

3.1 Feature-based SLAM/Structure from Motion

As I mentioned before, traditional feature-based SLAM estimate camera poses and 3D points' coordinates based on the handcrafted features. Now, this part will have an explanation for the theory.

3.1.1 3D rigid-body transformation

We will first define a way to describe the transformation of objects in 3D space. We use t for the translation, R for the rotation, and P for the coordinate after the transformation.

$$R = R_x R_y R_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

However, if we just describe the transformation like this, when having multiple translations, the expression will become too complicated just like this,

$$P_1 = R_1 P + t_1$$
$$P_2 = R_2 P_1 + t_2$$
$$P_2 = R_2 (R_1 P + t_1) + t_2 = R_2 R_1 P + R_2 t_1 + t_2 \neq R_2 R_1 P + t_1 + t_2$$

Thus, we will introduce homogeneous coordinates, so that

$$\begin{bmatrix} P_1 \\ 1 \end{bmatrix} = \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P \\ 1 \end{bmatrix}$$

Now the transformation matrix will be denoted as $T = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix}$, and we will be able to represent multiple transformations uniformly.

$$\begin{bmatrix} P_2 \\ 1 \end{bmatrix} = T_2 T_1 \begin{bmatrix} P \\ 1 \end{bmatrix}$$

3.1.2 Camera projection model

Camera projection is a mapping between 3D coordinate and 2D coordinate. The mapping is described as:

$$x = PX$$

x is the 2D image point, P is the projection matrix, and X is the 3D world point. The point will be transformed between different coordinate system, first is the world coordinate system, then is the camera coordinate system, and last is the image coordinate system.

$[R, t]$ transform a point form world coordinate system to camera coordinate system.

$$\tilde{X_C} = T\tilde{X_W} = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{X_W}$$

When using C for the camera's translation in the world coordinate system, we can get $X_C = X_W - C$. The real translation of the camera with relating to the world coordinate system is represented with the inverse of T:

$$T^{-1} = \begin{bmatrix} R^T & -R^T t \\ \mathbf{0}^T & 1 \end{bmatrix}$$

For the camera projection, we will use a pinhole model for deriving the projection formula.

$$x_{im} = K\tilde{X_C} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

K is called the intrinsic camera matrix, which includes the projection mapping and parameters of the camera, like the focal lens. By combining the 3D transformation with intrinsic matrix, we get:

$$\tilde{X_C} = T\tilde{X_W} = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{X_W} \tag{1}$$

$$\tilde{x_{im}} = K\tilde{X_C} = K[R \mid t]\tilde{X_W} \tag{2}$$

The matrix $[R \mid t]$ is called camera extrinsic parameters.

3.1.3 Essential matrix

According to the camera projection model, we can represent the point in two images with:

$$\tilde{x} = K\,[I, 0]\,\tilde{X_w}$$

$$\tilde{x'} = K'\,[R, t]\,\tilde{X_w}$$

in the camera coordinate system, they will be like:

$$\tilde{X_c} = [I, 0]\,\tilde{X_w} = \tilde{X_w}$$

$$\tilde{X_c'} = [R, t]\,\tilde{X_w} = [R, t]\,\tilde{X_c}$$

The essential matrix establishes the relation between X_c' and X_c, and we want to find that. Take the t to the outside:

$$X'_c = RX_c + t$$

Now we take the cross product on both sides, so that

$$t \times X_c' = t \times RX_c + t \times t$$

Since $t \times t = 0$, so

$$t \times X_c' = t \times RX_c$$

$$[t_\times]\,X_c' = [t_\times]\,RX_c$$

We will use the same method with X'_c

$$X_c'^T\,[t_\times]\,X_c' = X_c'^T\,[t_\times]\,RX_c$$

Again, since $X_c'^T\,[t_\times]\,X_c' = 0$. Thus,

$$X_c'^T\,[t_\times]\,RX_c = 0$$

When assuming the K and K' are all known and are equals to identity matrix, the equation will be like:

$$\tilde{x'}^T\,[t_\times]\,R\tilde{x} = 0$$

We define essential matrix E as,

$$E = [t_\times]\,R = t \times R$$

$$E = [t_x]\,R$$

3.1.4 Fundamental matrix

When deriving the essential matrix, we assumed the K as I, but when the camera is uncalibrated, K could not be assumed to be I. And now we need to solve that. Since,

$$\tilde{x} = KX_c$$

$$\tilde{x'} = K'X'_c$$

So,

$$X_c = K^{-1}\tilde{x}$$

$$X'_c = K'^{-1}\tilde{x'}$$

Previously we have:

$$X_c'^T [t_\times] RX_c = 0$$

Now we substitute the X'_c and X_c,

$$\left(K'^{-1}\tilde{x'}\right)^T [t_\times] RK^{-1}\tilde{x} = 0$$

$$\tilde{x'}^T \left(K'^{-1}\right)^T [t_\times] RK^{-1}\tilde{x} = 0$$

We can now have the fundamental matrix, denoted as F:

$$F = \left(K'^{-1}\right)^T [t_\times] RK^{-1}$$

Remember that $E = [t_x] R$, so the relationship between F and E appears to be:

$$F = \left(K'^{-1}\right)^T EK^{-1}$$

3.1.5 Computation of fundamental matrix

The ultimate target of the SLAM was to compute a precise fundamental matrix, which could tell us the translation and the rotation of our camera. We will use an 8-points algorithm to compute it.

Assuming we have M point correspondences:

$$\{x_m, x'_m\} \quad m = 1, \ldots, M$$

Each correspondence satisfies:

$$\tilde{x'}^T F\tilde{x} = 0$$

Writing the equation in detail, which is

$$\begin{bmatrix} x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = 0$$

This gives,

$$\begin{gathered} x_m x'_m f_1 + x_m y'_m f_4 + x_m f_7 + \\ y_m x'_m f_2 + y_m y'_m f_5 + y_m f_8 + \\ x'_m f_3 + y'_m f_6 + f_9 = 0 \end{gathered}$$

We can notice that each correspondence will give one equation. For m correspondences, we have:

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx_M' & x_My_M' & x_M & y_Mx_M' & y_My_M' & y_M & x_M' & y_M' & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_4 \\ f_7 \\ f_2 \\ f_5 \\ f_8 \\ f_3 \\ f_6 \\ f_9 \end{bmatrix} = 0$$

Represent the equation with simpler form:

$$Af = 0$$

For solving f which includes f_1 to f_9, we need 8 equations rather than 9 since the fundamental matrix is only up to scale.

After getting 8 correspondences with the feature detector and the feature matcher, we could find the fundamental matrix with Singular value decomposition (SVD).

$$A = U\Sigma V^T$$

The f we want is the singular vector corresponding to the smallest singular value of A, and it is the last column of V.

3.2 SuperGlue

For finding the precise fundamental matrix, we need more accurate correspondences. The procedure is to detect the key points on two images and then matching the key points. SuperGlue is a neural network that matches two sets of features by finding correspondences. The network takes key points and their descriptor as input and output the matching relationship between key points.

The algorithm contains two parts, the attentional graph neural network and the optimal matching layer. The attentional graph neural network encodes the key points and descriptors into a single vector, and apply the self-attention layer and cross-attention layer multiple times to enforce the vector f. Then f will be passed into an optimal matching layer, which creates a scored matrix. At last, the layer will use the Sinkhorn algorithm to calculate the optimal partial assignment.

3.2.1 Attentional graph neural network

key point encoder: The GNN would encode all key points with their descriptor and position. When combining the position with the descriptor, we can get a better discrepancy among points, and the position information is useful for the later attention process. For each point $\mathbf{x}_i$, we encode it as:

$$\mathbf{x}_i = \mathbf{d}_i + \mathbf{MLP}_{\text{enc}}\left(\mathbf{p}_i\right)$$

The MLP stands for Multilayer Perceptron It was used to embed the key point's position into a high dimensional vector.

multiplex graph neural network: Consider we have a single complete graph, whose nodes are keypoints in two images. The graph has two kinds of undirected edges, one is the self edge $\mathcal{E}_{\text{self}}$, it connects the keypoints inside the image, another is the cross edge $\mathcal{E}_{\text{cross}}$, it connects the keypoints with keypoints in another image.

We use ${}^{(\ell)}\mathbf{x}_i^A$ to represent the element i on the layer ℓ in the image A. The message $\mathbf{m}_{\mathcal{E}\to i}$ is the result of the aggregation from all key points including the $\mathcal{E}_{\text{self}}$ and $\mathcal{E}_{\text{cross}}$. The residual message updated for all i in A would be like:

$$ {}^{(\ell+1)}\mathbf{x}_i^A = {}^{(\ell)}\mathbf{x}_i^A + \mathbf{MLP}\left(\left[\ {}^{(\ell)}\mathbf{x}_i^A \ \|\mathbf{m}_{\mathcal{E}\to i}\right]\right)$$

The $[\cdot \mid \cdot]$ represents the concatenation. A fixed number of layers L with various parameters are aggregate along the self and cross edges. A similar update could be performed on image B simultaneously.

The self and cross-attention process is actually simulating the behavior that when human looks at two images. Self-attention can process the keypoints in one image, and cross-attention will find similar points in another image.

Attentional Aggregation: One of the innovations of the SuperGlue was to use the attention mechanism for feature mapping. $\mathbf{m}_{\mathcal{E}\to i}$ is the message that aggregates the message from self and cross-attention, represented by:

$$\mathbf{m}_{\mathcal{E}\to i} = \sum_{j:(i,j)\in\mathcal{E}} \alpha_{ij}\mathbf{v}_j$$

For each layer l, there are projection parameters, which were learned and shared by all the feature points. The $\mathbf{q}_i$ represents feature point i on the image. α_{ij} is the weight, get by $\alpha_{ij} = \text{Softmax}_j\left(\mathbf{q}_i^\top \mathbf{k}_j\right)$. As a result, message $\mathbf{m}_{\mathcal{E}\to i}$ is the similarity between two points, the larger the value the higher the similarity.

After L times of self and cross-attention, we can get the output from GNN. For image A, we are getting the matching descriptor:

$$\mathbf{f}_i^A = \mathbf{W}\cdot(L)\mathbf{x}_i^A + \mathbf{b}, \quad \forall i \in \mathcal{A}$$

3.2.2 Optimal matching layer

The optimal matching layer produces a partial assignment matrix $\mathbf{P}b$. The basic idea was to calculate a scoring matrix $\mathbf{S} \in \mathbb{R}^{M\times N}$ and maximize the score $\sum_{i,j}\mathbf{S}_{i,j}\mathbf{P}_{i,j}$. The score maximizing was solved by the Sinkhorn algorithm.

Score prediction: Directly calculate $M\times N$ matches would have a large amount of calculation. Thus, the Score would be calculated as:

$$\mathbf{S}_{i,j} =< \mathbf{f}_i^A, \mathbf{f}_j^B >, \forall (i,j) \in \mathcal{A}\times\mathcal{B}$$

$< \cdot, \cdot >$ stands for the inner product.

Occlusion and visibility: The SuperGlue has added a dustbins at the last column of the scoring matrix $\mathbf{S}$, and getting $\overline{\mathbf{S}}$. The dustbin could filter the wrong matches or the key points with no matches due to the occlusions on some points which is caused by changes in perspectives.

$$\overline{\mathbf{S}}_{i,N+1} = \overline{\mathbf{S}}_{M+1,j} = \overline{\mathbf{S}}_{M+1,N+1} = z \in \mathbb{R}$$

Key points on image A will be assigned to a matching point on image B or the dustbin, and the dustbin in image A and B have the same numbers of the key points. So, the assignment matrix has the constraints:

$$\overline{\mathbf{P}}\mathbf{1}_{N+1} = \mathbf{a} \quad \text{and} \quad \overline{\mathbf{P}}^\top\mathbf{1}_{M+1} = \mathbf{b}$$

which

$$\mathbf{a} = \left[\ \mathbf{1}_M^\top \quad N\ \right]^\top, \mathbf{b} = \left[\ \mathbf{1}_N^\top \quad M\ \right]^\top$$

3.2.3 Loss

The SuperGlue has used supervised training when giving the ground truth of the matching $\mathcal{M} = \{(i, j)\} \subset \mathcal{A} \times \mathcal{B}$, it minimizes the loss:

$$\begin{aligned} \text{Loss} = &- \sum_{(i,j)\in\mathcal{M}} \log \overline{\mathbf{P}}_{i,j} \\ &- \sum_{i\in\mathcal{I}} \log \overline{\mathbf{P}}_{i,N+1} - \sum_{j\in\mathcal{J}} \log \overline{\mathbf{P}}_{M+1,j} \end{aligned}$$

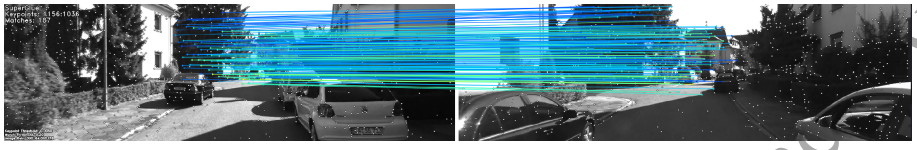


Figure 1: A example of SuperGlue matching on KITTI dataset

4 Experiments and Results

In this part, I will introduce the experiment I have done and compared the result with the other methods.

4.1 Model and setup

The code is based on a repository from GitHub which uses the FAST feature detector and optic-flow for feature matcher. On top of that, I change the feature detector to SuperPoint and feature matcher to SuperGlue. The whole procedure looks like this:

The first frame will be read, and the feature points in the image will be detected by the SuperPoint. The feature points' position and their descriptor will be stored. Then the second frame and all the feature points in the first frame will be sent into SuperGlue, and we can get the feature points in the second frame, which matches the feature points in the first frame. All the pairs of feature points will be used to find an essential matrix, and then the pose could be recovered with the essential matrix we found.

Something that might be confusing is that as I mentioned in 3.1.5, we only need 8 pairs of points to retain the fundamental matrix, but we are giving far more key points to calculate the fundamental matrix in practice. That is because I am using the Random Sample Consensus (RANSAC) algorithm to filter outliers and solving the optimum Fundamental matrix. The RANSAC algorithm would randomly pick 8 pairs of correspondences from all the feature points and compute the fundamental matrix F. Then, it will pick the points that lie in a range from epipolar constraints. Represent by:

$$\tilde{x'}^T F \tilde{x} \leq \text{ Threshold}$$

By repeating this procedure multiple times, we can choose the F with the maximum inliers as our optimum fundamental matrix.

4.1.1 Hyper parameters

For the SuperPoint detector and SuperGlue matcher, we have many hyper parameters to adjust. During the experiment, I found that the matching threshold had a massive effect on the result. The model tends to have the best result on KITTI when the matching threshold is set between 0.35

to 0.40. A threshold above or between this interval will make the odometer perform worse. As shown in figure 1, the first result is retained when setting the matching threshold to 0.45, and the second result it retained when setting the matching threshold to 0.35. The first result has a better predicted trajectory for the blue section, where the second result has an better prediction for the green section. The environment and the threshold together determine the quality and quantity of the matches.

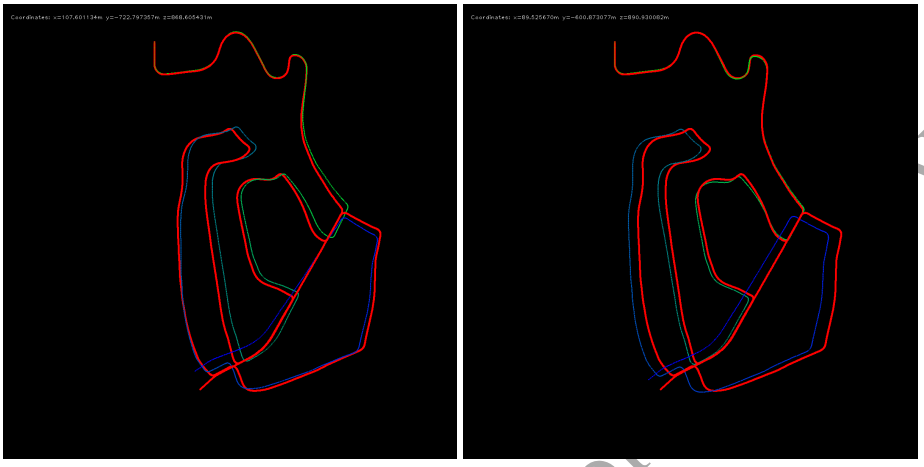


Figure 2: The matching threshold's effect on the predicted trajectory

Although a lower matching threshold could give us more matching points, for the result, our model did not gain a better prediction on the trajectory. One possible reason is that a lower threshold would give more poor-quality matches, and the fundamental matrix would be less accurate. For a higher threshold, the matches would be so less and the RANSAC algorithm could not perform well to give an optimum fundamental matrix. Conclude from the experiment, I eventually set the matching threshold to 0.40.

There are also other hyper parameters I adjusted for the neural network, but I won't talk about them here. If you want to know my choice on other parameters, all of them could be checked in my source code.

4.2 Dataset

The test was based on the KITTI dataset, which is a dataset used for computer vision benchmarks. It includes data to evaluate stereo vision, optical flow visual odometry, etc. For the visual odometry dataset, KITTI provides 20 sequences of images with 10 of them have the ground truth. The ground truth includes the transformation in three dimensions and the rotation. In the KITTI dataset all the images were in a driving perspective on the road. The image was collected by a vehicle with a stereo camera, but we can use images from only one camera for our monocular visual odometry.

For one sequence in the dataset, there were about thousands of images. Our target was to use the SuperVO to recover the transformation and rotation of the camera from the sequence of images. The ground truth of the data includes the transformation and rotation in three dimensions and also a time stamp for comparing the sequence of the frames. As I mentioned before in 3.1.5, the fundamental matrix we calculated will give us the value of transformation and rotation. We can directly compare the difference between the ground truth and the result.

4.3 Evaluation Criteria

When evaluating the result of visual odometry, one of the criteria we can use is called absolute trajectory error (ATE). The ATE is the direct difference between the estimated pose and the real pose, which can very intuitively reflect the accuracy of the algorithm and the global consistency of the trajectory.

Let's express all the poses we retained with our visual odometry as

$$P_1, \ldots, P_n \in SE(3)$$

And we express the ground truth poses as

$$Q_1, \ldots, Q_n \in SE(3)$$

The Δ assigns for the time interval. It should be noted that the estimated pose and ground truth are usually not in the same coordinate system, so we need to align the two first. We calculate a conversion matrix $\mathrm{S} \in \mathrm{Sim}(3)$ from the estimated pose to the real pose.

The frame i's ATE equals to

$$F_i := Q_i^{-1} S P_i$$

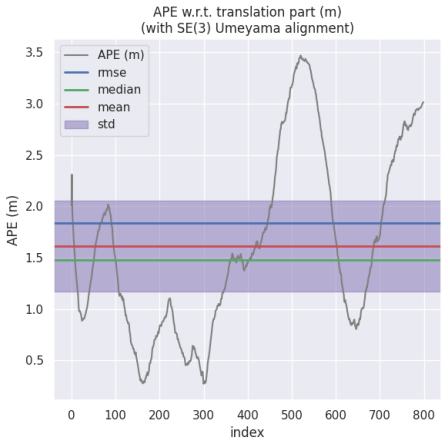
Then we can use the root mean square error (RMSE) to count this error. Of course we could even use the average or median to retain a single value for our ATE. At last, we have our ATE as

$$\mathrm{RMSE}\left(F_{1:n}, \Delta\right) := \left(\frac{1}{m} \sum_{i=1}^{m} \left\| \mathrm{trans}\left(F_i\right) \right\|^2 \right)^{\frac{1}{2}}$$

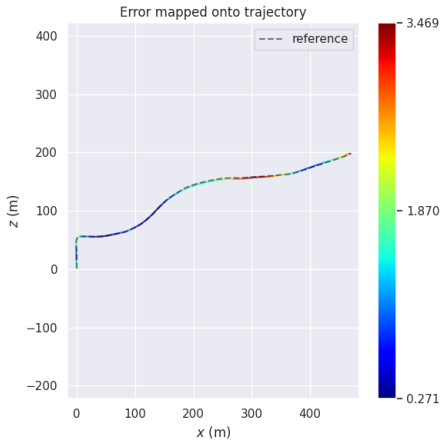
ATE is also called Absolute pose error (ATE), the tools I used later would call ATE as APE, but they are actually the same.

4.4 Experiment Results

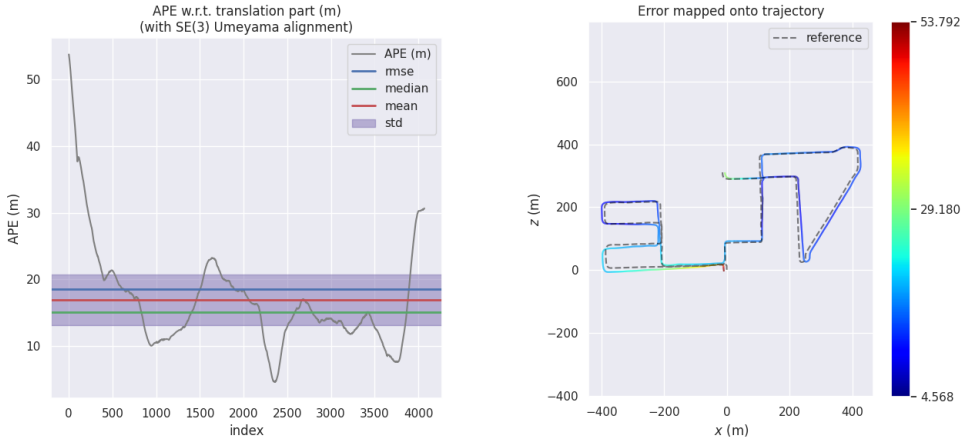
Here are some results on the KITTI dataset. All the data are ensured to use the same parameters and program, so the result of the model could be evaluated more consistently. The visualization tool I used is evo[15], which is an open-sourced tool written with Python.



(a) 03 APE

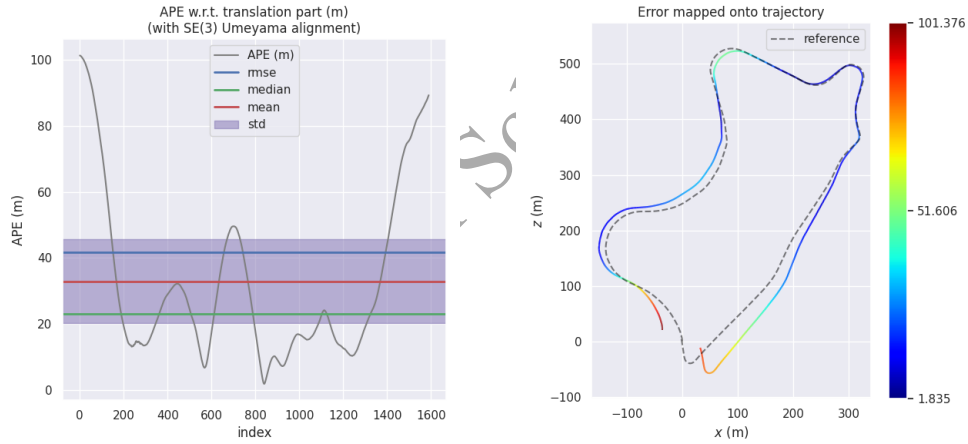


(b) 03 Trajectory



(c) 08 APE (d) 08 Trajectory

Figure 3: The APE and predicted trajectories



(a) 09 APE (b) 09 Trajectory

The SuperVO tends to have a better result when the trajectory is simple and shorter, but when dealing with a relatively complex path, SuperVO still shows its robustness as the last image in Figure 2 shows.

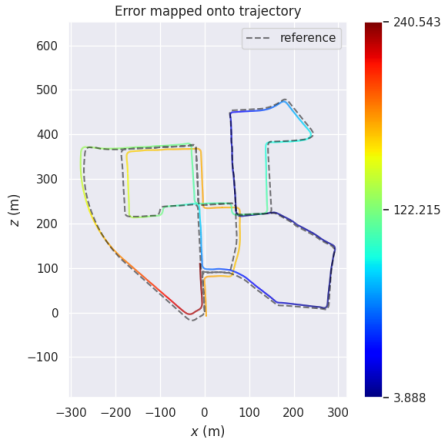
4.5 Comparison

When comparing with other learning-based methods, the SuperVO gets a better average error. Though it did not win the ORB2 with the traditional method on average error, it did win the ORB2 in sequence 04, 05, 06, 08 and 09. Since the KITTI datasets only offer 11 sequences with ground truth, we can't test on more data to compare the ORB2 without SuperVO.

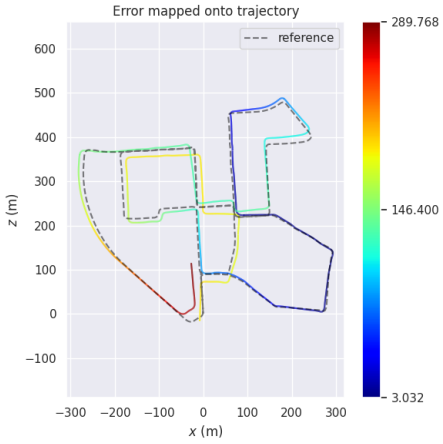
	00	01	02	03	04	05	06	07	08	09	10	Avg.Err
SuperVO	77.26	**58.56**	101.52	1.10	0.36	**26.9**	16.18	21.21	**18.57**	22.15	9.27	32.15
SFM-Leaner	93.04	85.9	70.37	10.21	2.97	40.56	12.56	21.01	56.15	15.02	24.70	34.21
VISO2	79.24	494.6	70.13	52.36	38.33	66.75	40.72	18.32	61.49	52.62	57.25	53.721
Depth-VO	64.45	203.44	85.13	21.34	3.12	22.15	14.31	15.35	29.53	52.12	24.70	33.22
FAST + LK	146.27	81.68	38.72	1.19	0.26	61.39	14.91	11.00	65.25	48.97	10.54	43.65
ORB2	40.65	502.2	47.82	0.94	1.30	29.95	40.82	16.04	43.09	38.77	5.42	26.48

Table 1: The comparison with other method's ATE, part of the data came from [16]

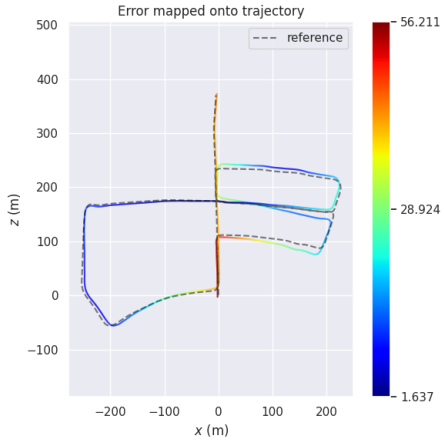
FAST+LK is another traditional algorithm. It does not use keypoints' descriptors to do the matching. It uses another kind of algorithms called optic-flow. Lucas–Kanade (LK) is one of the optic-flow algorithm. The optical flow method uses the changes in the time domain of the pixels in the image sequence and the correlation between adjacent frames to find the corresponding relationship between the previous frame and the current frame, thereby calculating the movement of objects between adjacent frames A method of information. The FAST+LK methods will use the FAST to detect the position of key pixels and then use LK to find those pixels' position in the next frame and solve the fundamental matrix. Here are some of the comparisons between FAST+LK and SuperVO. The SuperVO's results are on the left-side, FAST+LK's results are on the right-side.



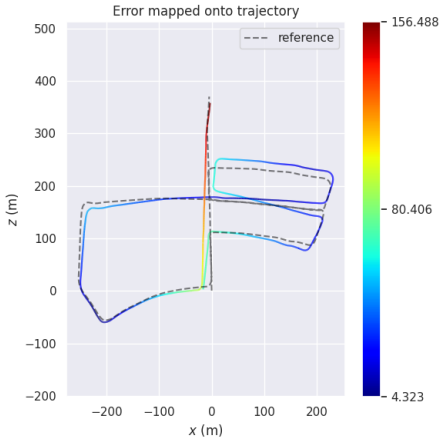
(c) 00 by SuperVO



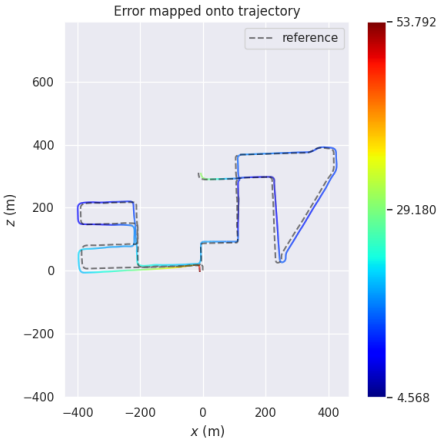
(d) 00 by FAST+LK



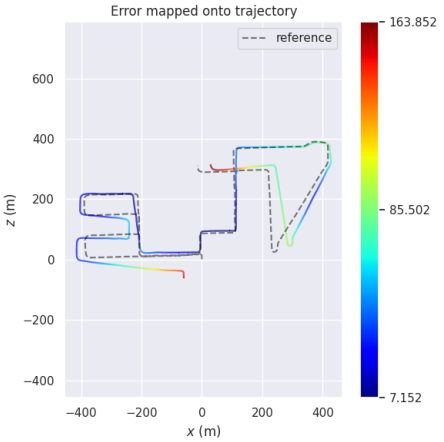
(e) 05 by SuperVO



(f) 05 by FAST+LK



(g) 08 by SuperVO



(h) 08 by FAST+LK

Figure 4: The trajectory predicted by SuperVO and FAST+LK

5 Conclusions and future work

In conclusion, the SuperVO shows its ability to predict the complex trajectory when comparing with other learning-based VO methods and the FAST+LK methods. By introducing the learning features, the SuperVO could have better robustness when facing a change in environment lighting, which is one of the weaknesses of the traditional feature detecting and optic-flow. The SuperVO has proved the advantages when brought the learning-based algorithm to the Visual Odometry.

The main problem of the SuperVO is the over concentrate on the relative transformation between two frames. One of the reasons that ORB methods could perform better then SuperVO in some sequences is its ability to attract key frames. By introducing key frames, ORB could avoid the noise brought by the slight change between two frames and recover the trajectory on a larger perspective. I believe that after adding the key frame mechanism, the SuperVO could achieve an even better result.

Also, I expect that the self-attention layer in the SuperGlue could have a connection with the recovery of the fundamental matrix since the Visual Odometry has an assumption on the static environment. I believe the self-attention used in the SuperGlue could solve the problem of getting the wrong trajectory when there are moving objects in surroundings by filter the key points that have relative movements.

References

[1] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. 4, 5

[2] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007. 4, 5

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 4, 5

[4] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, 2011. 4

[5] J. Engel, T. Schops, and D. Cremers. LSD-SLAM: Large-scale direct monocular slam. In *ECCV*, 2014. 4

[6] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018. 4

[7] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 4

[8] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*, 2018. 4

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, 2004. 4

[10] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features authors. In *ECCV*, 2006. 4

[11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011. 4

[12] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV*, 2006. 5

[13] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *ECCV*, 2006. 5

[14] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *ECCV*, 2016. 5

[15] Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017. 13

[16] Huangying Zhan, Chamara Saroj Weerasekera, Jiawang Bian, and Ian Reid. Visual Odometry Revisited: What Should Be Learnt? *arXiv e-prints*, page arXiv:1909.09803, September 2019. 15

[17] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 18

6 Acknowledgment

I would like to acknowledge all the people who had helped me with my project, especially to the Magic-Leap team. Their work in the CVPR 2020, [17] "SuperGlue: Learning Feature Matching with Graph Neural Networks", gives me the idea to use the GNN matching for a visual odometry system. The result of the SuperGlue was fantastic, and I was amazed by the result when using the SuperGlue matching for fundamental matrix recovery. Besides, thanks to my friend Er Hu, who told me that the SuperGlue's ability in giving high-quality matching key points. My research has used a lot of resources on the GitHub. I really appreciate their contribution to the community. I will keep studying hard and become one of the members to provide high-quality open-source projects.

I would like to thank to my parent, when I have trouble with a lack of computing power, it is my father who borrowed an RTX2080Ti from his colleague. My teacher Dongxu Sun is the person who had offered me a lot of advice when writing the paper.

This research was absolutely tough for me, and I really spent a lot of time on it. The paper was just finished on the last day. Once again, thanks to all my friends and teachers and to all the people who had helped me.