TECHNICAL UNIVERSITY OF DENMARK

DTU ELECTRICAL ENGINEERING

Evaluation of feature extract algorithms for visual odometry

Special Course Report

s182354 XIAOHAN LYU

June 15, 2019

Abstract

Computer vision is a field that study how computers can understand digital image and videos. It is include many method, such as image analysis, understanding digital images. This report focus on a part of image analysis, which is image stitching. This report discuss the principle of each step of image stitching and complete it based on MATLAB and OpenCV.

Keywords

Rigid motion, Feature matching, Image stitching, Feature benchmark, OpenCV.

Contents

Sec.1 Introduction

Image features is one of an important part in image analysis. There are so many image analysis based on feature, like image feature detection, texture recognition, object classification, image stitching and so on. Moreover, due to most of computer image analysis algorithm use feature as the basic element to cacluate, there are many algorithm related to image feature has been developed. Such as Harris for feature detection, SIFT(scale-invariant feature transform), ORB(Oriented FAST and rotated BRIEF) for feature extraction (also can using for feature detection). During this special course some of image analysis algorithm based on feature will be discussed, I will complete some classical algorithm after fully understanding its principle. This special course has two major topics that image stitching and the evaluation of feature extract algorithms for visual odometry. The software I use to do image stitching is MATLAB, while for algorithm evaluation, I use Visual Studio with OpenCV open source library. This report consists all exercises that I complete during I am taking this special course period. I will complete a part of algorithm about image stitching and algorithm evaluation in each exercise, finally combine all of these exercises together. Through doing these exercises, I complete two major topic of this special course.
The subject of this special course is Visual-Inertial sensor fusion for image stiching. I take this special course in the second semester of my studies at DTU. On first seven week I have lecture and exercise each week, and on last six week I do individual assignment. My supervisor are Daniel Haugård Olesen and Xiao Hu, have given me many help during the whole special course period.
I would like to say I have learned a lot from this special course. Also. this course improve my interesting for image analysis and makes me to decide to follow the study line with image analysis field.

1.1 Course Proposal

- Representation of rigid motion.
- Images. Corner features and blob features.
- Feature descriptors and feature matching.
- Robust estimation method: RANSAC.
- Two view geometry: essential, fundamental and homography.
  DLT for homography matrix.
- Benchmark for feature detectors and descriptors, and matching strategy.

Sec.2 Methodology

2.1 Transform Representation

- Task 1: Representation of 3D rigid motion: rotation matrix, Euler-angle and quaternion.
- Task 2:camera calibration.

1-a Transform Matrix and camera calibration

Imagine that there are rigid body in the 3D space. For example, camera. This camera can do such motion like rotation and translation. In order to describe the motion of camera and get information that where the camera it is, transform matrix is necessary. Consider the rotation and translation cases. Camera motion is a rigid motion, which means that a length and angle of same vector didn't have change in each coordinate. This kind of transform is be known as Euclidean transformation. So when camera rotate in the current coordinate, suppose that this camera is a point in this coordinate. We can simplify this process as the vector rotation in two-dimensional plane:

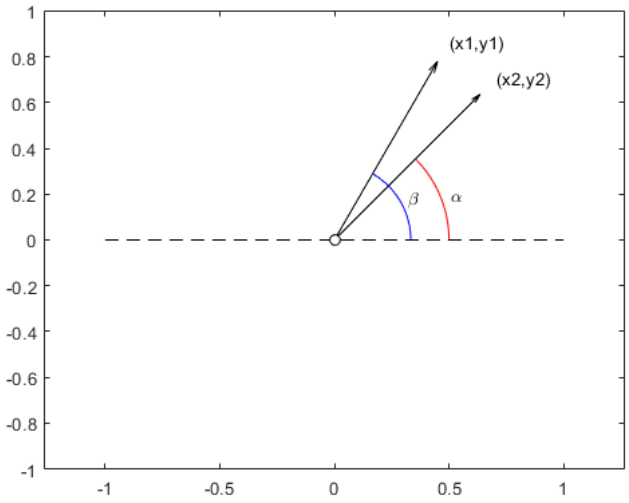

Figure 1: Vector Rotation

From the figure we can obvious the relation of (x_1,y_1) and (x_2,y_2) easily. Set length of vector is S. Thus:

$$x_1 = S \cdot cos(\beta), \quad y_1 = S \cdot sin(\beta), \quad x_2 = S \cdot sin(\alpha), \quad y_2 = S \cdot sin(\alpha).$$

Define θ=β-α, then we have $x_1 = S \cdot cos(\alpha + \theta)$, $y_1 = S \cdot sin(\alpha + \theta)$. Simplify these equation use Trigonometric function and solve it by elimination by substitution, we can describe (x1,y1) by a equation involving (x2,y2):

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & -cos(\theta) \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \tag{1}$$

For translation cases, Suppose that this camera translate from (x_1,y_1) to (x_2,y_2). Define $\triangle x$= x_2-x_1, $\triangle y$= y_2-y_1. Thus we can get:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} \triangle x \\ \triangle y \end{bmatrix} \tag{2}$$

1-b Homogeneous Coordinates

Homogeneous coordinates is a very useful way to help us to deal with image transform problem. Formula which include homogeneous coordinates are more simpler than the one involving Cartesian coordinates.

Consider image scaling. Image only can do transformation in 2-dimentional case. Essentially, image scaling is the scaling of each pixels' vector in the image. That means for every pixel, its scaling can be expressed as equation 1.

$$\begin{bmatrix} k_x & 0 \\ k_y & 0 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

Here (u,v) is pixel in image which is after scaling, (x,y) is pixel in original image. k_x, k_y is scaling ratio.
From above discussion we know that both image rotation and image scaling can be described as the form of matrix multiplication. Generally, image transformation always with rotation, scaling and shifting together. So how to take account image translate to the whole transformation process? In order to compute these case together, homogeneous coordinate is introduced. Enlarge dimension to three-dimensional, so we can get transform equation as following:
Image scaling:

$$\begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4}$$

Image rotation:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & -cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{5}$$

Image translation:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \triangle x \\ 0 & 1 & \triangle y \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Whatever how image will transform, it always can be described as matrix multiplication.

1-c Euler Angle

The way that rotation matrix describe the rotation has 9 unknown variable, while once rotation only has three freedom degree. There are some redundancy in this case. Besides, rotation matrix has constrain in itself: It must be a orthogonal matrix and its determinant must be 1. To solve this problem Euler is introduced. The method to use Euler angle to describe is:

- Rotate with Z-axis and get yaw.
- Rotate with Y-axis and get pitch.
- Rotate with X-axis and get roll.

So we can get a 3-dimension vector $[r, p, y]^T$ to describe arbitrary rotation. The advantage of this method is that it can express the rotation process is very intuitive.

1-d Quaternion

Euler angle has a major defect, which is Gimal Lock. When pitch equals to $\pm 90^\circ$, first rotation and third rotation rotate around same axis. In this case system will miss one freedom degree. This problem is known as singularity. In fact, all the describe methods based on 3-dimension have singularity. Quaternion can solve this problem.
An quaternion's expression is

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{7}$$

Here i, j, k is the imaginary part of quaternion. The i, j, k satisfy the following relationship:

$$\begin{cases} i^2 = j^2 = k^2 = -1 \\ ij = k, ji = -k \\ jk = i, kj = -1 \\ ki = j, ik = -j \end{cases} \tag{8}$$

We can use quatetnion to describe rotation. Suppose there are a rotation that rotate around $n = [n_x, n_y, n_z]^T$ with angle θ. This quatetnion can be express as:

$$q = [cos(\frac{\theta}{2}), n_x sin(\frac{\theta}{2}), n_y sin(\frac{\theta}{2}), n_z sin(\frac{\theta}{2})]^T \tag{9}$$

The new pose is

$$p' = qpq^{-1} \tag{10}$$

1-e Transformation among three expression

Because I just compute the motion by using these equation directly. I do not deduce these equation by my self so the derivation process is omitted. Form quatetnion to rotation matrix:

$$R = \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2q_1q_2+2q_0q_3 & 2q_1q_3-2q_0q_2 \\ 2q_1q_2-2q_0q_3 & 1-2q_1^2-2q_3^2 & 2q_2q_3+2q_0q_1 \\ 2q_1q_3+2q_0q_2 & 2q_2q_3-2q_0q_1 & 1-2q_1^2-2q_2^2 \end{bmatrix} \tag{11}$$

From rotation matrix to quatetnion, suppose m_{ij} meaning element in ith row, jth column:

$$q_0 = \frac{\sqrt{tr(R)+1}}{2},\ q_1 = \frac{m_{23}-m_{32}}{4q_0},\ q_2 = \frac{m_{31}-m_{13}}{4q_0},\ q_4 = \frac{m_{12}-m_{21}}{4q_0} \tag{12}$$

From Euler angle to rotation matrix:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 1 & sin(\theta) & cos(\theta) \end{bmatrix} R_y(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

The expression of rotation matrix is:

$$R = R_x(\theta)R_y(\theta)R_z(\theta) \tag{14}$$

Relatively, from rotation matrix to Euler angles (using Z-Y-X order) we have:

$$\theta_x = atan2(m_{32}, m_{33}),\ \theta_y = atans(-m_{31}, \sqrt{m_{32}^{+} m_{33}^2}),\ \theta_z = atan2(m_{21}, m_{11}) \tag{15}$$

From quaternion to Euler angles we can compute it with two steps. The transform between quaternion and rotation matrix is computed firstly, then compute transform from rotation matrix to Euler angles.

2.2 Harris

Harris and Stephens developed the algorithm that be known as Harris Corner Detector. Corner is the junction of profile, which is among the first low-level features used for image analysis and in particular, tracking. The main idea of this algorithm is that using a sliding window ,which this sliding window move one pixel each time, compare the difference of pixel gray between before sliding and after sliding, then compute the position which the gray changes significantly. When the windows moving in [u,v], the change degree of pixel gray points has been describe as :

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \tag{16}$$

[u,v] is the offset of windows. (x,y) is the coordinate of windows' center. I(x,y) is the intensity of the pixel at (x,y), hence I(x+u,y+v)-I(x,y) will be the difference in intensities of the window shift. w(x,y) is the window function. Generally windows function has two types: Gaussian and Rectangular.
In order to compute E(u,v) more easily, we need to simplify this function by Taylor series expansion, but only with the first order. Function 1 can be rewriting as following:

$$E(u,v) \approx (u,v)M \begin{pmatrix} u \\ v \end{pmatrix} \tag{17}$$

Here,

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{bmatrix} \tag{18}$$

But Harris algorithm doesn't check corners by using the value of E(u,v) directly. Considering three situations: linear edge, flat, corner. Draw the Ix-Iy figure for all three types of patches

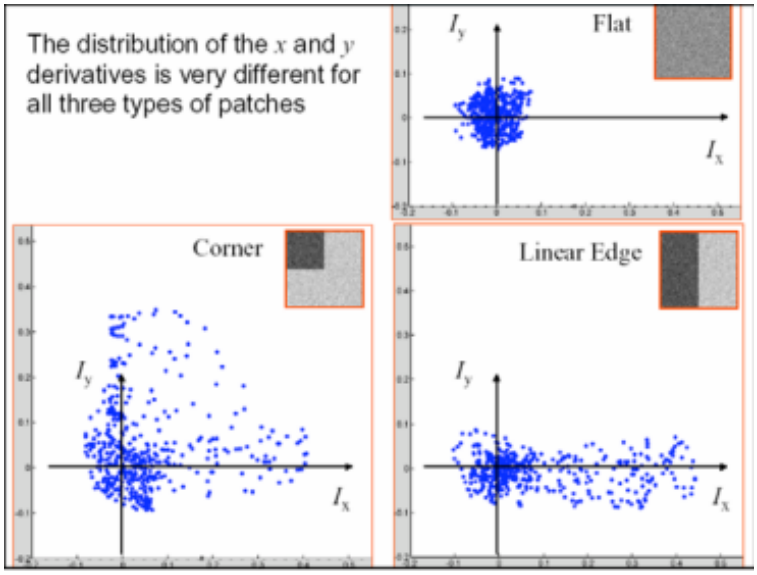

Figure 2: Ix-Iy

It is obvious that their has different features. For pixels in flat area, even their gradients' direction are different, the amplitude are always small. For pixels in edge area, if direction of edge is horizon(vertical), ranges of Iy(Ix) will be large. For pixels in corner area, both Ix and Iy has large range of values. According these features we can classify pixel points with the eigenvalue of M matrix. Calculate an score named R for every windows:

$$R = det(M) - k(trace(M))^2 \tag{19}$$

Here k is an empirical constant, which always between 0.04-0.06.
Large value of R indicates the corner, negative value of R indicates the edge, while small value of R means that this window located in a flat area.

2.3 FAST detector

Features from accelerated segment test (FAST) is a corner detection method, which was originally developed by Edward Rosten and Tom Drummond. Compared with Harris Corner detector, FAST algorithm is more computational efficiency. So it's very suitable for real-time feature tracking.
The main idea of FAST algorithm is select a center pixel point and then detect pixels' gray value around the center pixel point to classify whether this center pixel point is corner. In order to make the concept of "pixel points around center pixel point" more clear, Bresenham circle of radius 3 is introduced.

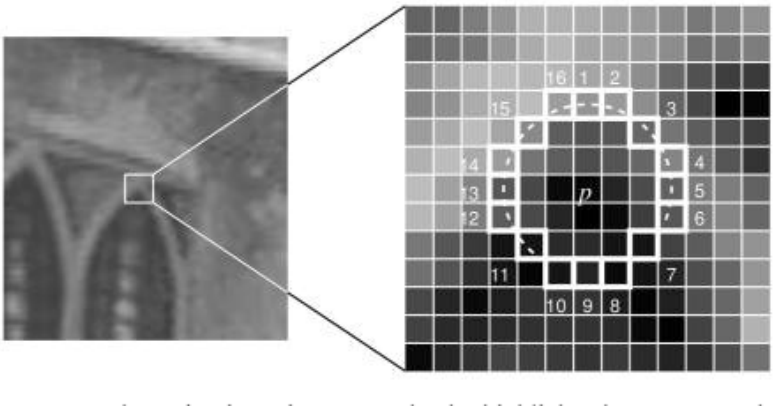

Figure 3: Bresenham circle of radius 3

It is composed of 16 pixels and center pixel point is the center of this circle. Let the intensity of center pixel point be I_p. Now, Define a threshold t, then compute I_1 to I_{16}. If there exists a set of n contiguous pixels in the circle brighter than center pixel, this center point is defined as corner. Repeat above steps for all pixels in this image. Finally using non-maximal suppression to discard inappropriate corners. That's all contents of FAST algorithm. Generally n value will be 12, but when n equals to 9, we can get better results.

2.4 BRIEF Descriptor

Due to the SIFT descriptor being a 128 vector, which is more slowly for real-time feature tracking, BRIEF descriptor has been invention
First of all, Define a patch p of size $s \times s$, where keypoint p is the center of this patch. Randomly sample N pairs(x,y) which located on this patch and compare its gray value each other. In Calonder's experiment, they test five ways for choosing the pairs:

- pairs(x,y) satisfy uniform distribution.
- pairs(x,y) satisfy Gaussian distribution.
- First location x is sampled from a Gaussian , next location is sampled from another Gaussian centered on x.
- Randomly sample pairs from discrete locations of a coarse polar grid introducing a spatial quantization.
- Let x as the center, sample all possible y value on polar coordinate system containing all a set of N points.

Test result shows that second way has small advantages than other three. So in my experiment second way will be choose. Then test the results with τ. Define test τ as

$$\tau(p;x,y) = \begin{cases} 1 & if\, p(x) < p(y) \\ 0 & otherwise \end{cases} \tag{20}$$

From above equation we can see because of method of this algorithm is that compare single pixel intensity, it's very noise-sensitive. Pre-smoothing the patch can reduces the impact of this problem and makes descriptors more stable and repeatable. So before test τ smoothing kernels will be used to reduces the influence of noise. Finally, Combine N binary string to be the N-dimensional bitstring.

$$f_{n_d}(p) := \sum_{1 \le i \le n_d} 2^{i-1} \tau(p;x_i,y_i) \tag{21}$$

2.5 SIFT

SIFT(Scale-invariant feature transform) is used to detect and describe the local interest points in a image. Major SIFT algorithm has six steps.

- Smooth by Gaussian and build scale space.
- Compute difference of Gaussian between adjacent scale space.
- Detect local extreme value in scale space. Search all scales which image located in, and interest points that scale invariant and rotation invariant are defined by using Gaussian function.
- Identify the location of interest point. Using a fitting model to ensure the location and scale. It makes interest points more stable for matching and recognition.
- Direction determination. Interest point will has one or several direction based on local gradient direction in image.
- Generate interest point descriptor.

Basically, method of SIFT algorithm is that search interest point in different scale and compute information of these interest point(size, direction, scale), then use these information to get interest point descriptor.

2.6 RANSAC

RANSAC (Random sample consensus) is the way using iteration to estimate mathematics model from the dataset which has noise. This algorithm suppose that there are both inliers and outliers in dataset, only inliers can be used to estimate mathematics, while outlies doesn't have influence for estimation of mathematics. Design RANSAC for line estimation based on above principle.

- From the prior knowledge in algebra we know that two points can determine a line in the plane. So first of all, two points(x and y) are sampled randomly from database and compute its linear equation by using x,y.
- Set a threshold. Then calculate the distance between other point in this dataset and this line.
- Repeat step two for all points in this dataset and record amount of inliers.
- Now the linear equation is best estimation if it has the most inliers.

For plane estimation the process is very similar as line estimation. Just change criterion distance. i.e. For plane estimation, Randomly sample three points to determine plane equation and calculate distance from each points in dataset to this plane.

6-a RANSAC in feature matching

The method that compute homography matrix by using RANSAC as following:

- From Brute-force matcher the matching pairs can be got.
- Compute initial value of homography matrix between two matrix.
- Using RANSAC algorithm to find the 'best' homography matrix.

Last step will be explained more detailed. In order to design the RANSAC algorithm for feature matching, we need to calculate homograghy matrix. This matrix describe mapping relations between two planes. i.e. the relation of the same object in different viewpoints. Here I use DLT (Direct Linear Transformation) to compute homography matrix. Define a point x_1 in first image as (u1,v1,1), a point x_2 in second image as (u2,v2,1). According to the definition of homography matrix, Suppose that homography matrix is

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{22}$$

we can get x_2=Hx_1. i.e.

$$\begin{bmatrix} 0 & 0 & 0 & -u_1 & -v_1 & -1 & v_2u_1 & v_2v_1 & v_2 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_2u_1 & -u_2v_1 & -u_2 \\ -v_2u_1 & -v_2v_1 & -v_2 & u_2u_1 & u_2v_1 & u_2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{23}$$

We can see from the equation 15 that row 3 is a linear combination of first and second row, so third row is discarded, then all element in H matrix divide h_9 to reduce the number of unknown variable. Finally simplify this equation as $Ah = 0$. This equation can be solved by using four points. Randomly select four points from the interest points in each image to get h matrix, then reconstruct H matrix and use it to compute the difference between x_1 and the mapping of x_2 on first image. In order to make matching results more precisely, I also check the difference between x_2 and the mapping of x_1 on second image. Finding a H matrix that makes most points can have a shortest distance with its mapping on another image.

2.7 Image Stiching

Image stiching contains three major part: feature extraction and matching, image registration and image warping and blending.
The first part of image stiching is as I do in section section 3. There are also others algorithm that can be used to do feature matching, such as SURF(speeded up robust features), ORB(Oriented FAST and rotated BRIEF).
Image registration is the process to aligning two or more images of the same scene.[**?**]. Through find a transformation to transform a projective image to another image, which make the points that located in same location in the space can correspond with each other.Because of interest points has been detected in feature matching part, so the way I choose to complete image registration is based on interest point. The basic method is that compute the homography matrix through matching pairs. The method of compute homography matrix has been discussed in section 4.
Do image warping and blending.
For mutiple image stitching, there are some different method to complete it. One way is that do image stitching one by one. Choose two images to stich and extract feature based on stiched image and third image, then stich them. Repeat this procedure until all of image has been stiched to one panoramic image.
This method is very easy to implement. It is also the method that I use in my mutilple image stitching experiment. But an disadvantage of it is if there are large database for image stitch, this method will be very time-consuming because the whole procedure (including feature matching, image registration and image warping and blending) will repeated for every times of stiching.
Another method is to find the image which nearest to center. Supposed that there are n images in database. This method is following these steps:

- Load all images and do feature detection and matching between I(n) and I(n-1).
- Estimate transformation H(n), which maps I(n) to I(n-1).
- Compute the whole size of the final panoramic image. Find a bounding box that every stiched images have a overlap part with it.
- Find center image. The image with the shortest distance between the center of image and the center of the bounding box is center image.
- Compute homography matrix that other images transform to center image.
- Warping and blending.

There are a method using minimum spanning tree. Also supposed that there are n points in database. The major algorithm is following:

- Detect feature and get descriptor.

- Matching for each image, then get an n by n matrix, which its element indicate how many matches obtained.
- Treat this matrix as a graph. Find a tree that can connect all nodes by using minimum cost, which is minimum spanning tree.
- A node in this tree can be set as the root. Thus the shortest path from root node to current can be found. we can get the relationship (for example, homography matrix) between center image and other images from this shortest path.
- Warping and blending.

This method order the image sequence by the number of feature matches. It will firstly compute the transformation relation between two images that has the most number of feature matches.

2.8 Algorithm Performance Evaluation

This performance evaluation will work on OpenCV based on C++.

8-a Data Set

In order to test the performance of different algorithm in different kind of database, I set the kinds of images as more as possible .Due to this evaluation is for visual odometry, it's better to test algorithm with KITTI database.
KITTI is a large computer vision algorithm evaluation database based on autonomous driving scene. It can be used for stereo image evualution, optical flow, visual odometry, 3D object detection an so on. This database includes the real image data that take from city, rural area and freeway. And the raw database is divided into several class: road, city, residential, campus, person. To get this database, two high-resolution color and grayscale video cameras, a 3D leaser radar and GPS localization system is equipped. The sensor parameter can be found in official website.
For evaluation that I take, I use the odomery data set (gray scale). Two of image in this data set is shown as follow:

Figure 4: Scene 1

Figure 5: scene2

Also, in order to make this evaluation more completely, I set other kind of image transformation: image blur (figure 6, 7), light change (figure 8, 9), distortion (figure 10, 11), view change (figure 12, 13) These images allow to analyze the influence of the image transformation and scenes type to feature extract and feature matching algorithm.

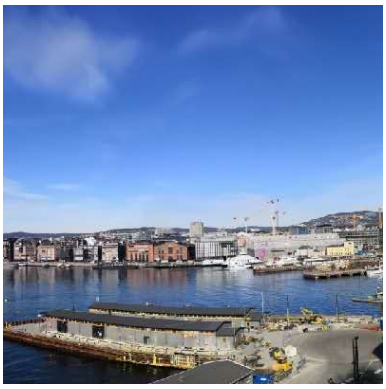

Figure 6: Original image

Figure 7: Image Blur

Figure 8: Plane

Figure 9: Lighter Plane

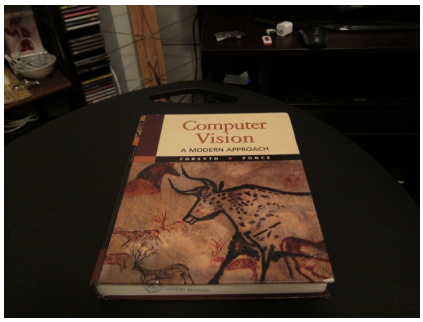

Figure 10: view 1

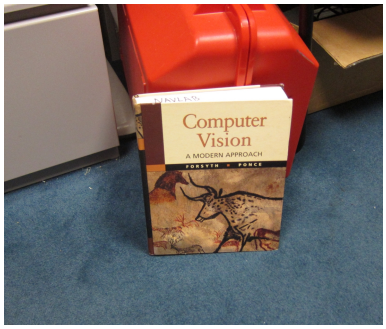

Figure 11: view 2

Figure 12: Texture 1

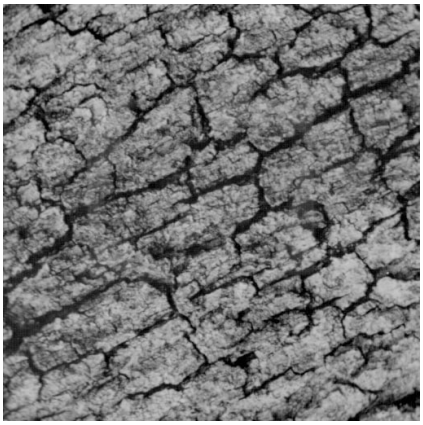

Figure 13: Texture 2

Figure 14: Tiles

Figure 15: Distort Tiles

8-b Detectors and Descriptors

In this evaluation I select three algorithm : SIFT, SURF and ORB. As I discuss in the Harris and FAST feature detection, Harris do not produce a good performance and FAST do not have rotation invariance and scale invariance, so this two algorithm is throw away directly. We still have some other options.
SURF(speeded up robust features) is an improvement of SIFT. It follows the same principle as SIFT except some details. Theoretically, the computation speed of SURF will be more quickly than SIFT.
ORB (Oriented FAST and Rotated BRIEF) is an improvement that combine FAST detector and BRIEF descriptor. Due to both they do not satisfy rotation invariance, ORB is developed in order to solve this drawbacks of FAST detector and BRIEF descriptor.
These state-of-the-art algorithm are also very popular, except these there are also some other methods such as HOG and DOG, this report won't discuss the details about it.

8-c Evaluation criterion

I use two method to evaluation the performance of these algorithm.
First one is to check "good matches". Two corners are matched if their distance d_1 and d_2 are less than a threshold. Meanwhile, descriptor in reference image is also corresponding with a descriptor in the transformed image. Count the number of these matches and then make an accuracy formula:

$$survival\ rate = \frac{good\ matches}{whole\ matches} \tag{24}$$

This is the first criterion to evaluate the performance. Because of the feature detected algorithm that are used for visual odometry, viewpoint will change with the time. matches will be less and less from first view to last view. Otherwise, feature matching method also would lose features. So it is necessary to evaluate features survival rate.
Second one is to check "correct matches". Recall the experiment in RANSAC there still are some mismatch even using distance test. So second criterion is that define a ratio to see how will the corners correspond under the transformation. Here I use homogaphy to calculate the relationship between two images. Then do things as same as RANSAC experiment, find inliers from the iteration, and make an accuracy formual:

$$accuracy\ performance = \frac{Inliers}{whole\ interest\ points} \tag{25}$$

Also there are some other methods to validates whether matches are correct. For example, epipolar geometry. In some experiments[**?**] the epipolar constraints be used and through build a triplet (initial feature-auxiliary feature-proposed match match) to verifies all epiploar conditions.

Sec.3 Results and Discussion

3.1 Harris Corner Detection

- Task: Complete the detection using Harris corner detector based on MATLAB.

Using sobel filter to compute gradient. In order to make result visualization, Figure 2 has been plot:

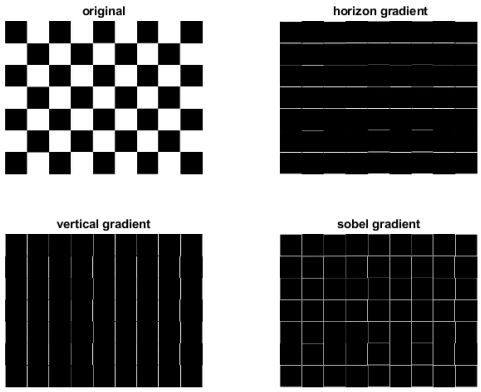

Figure 16: Sobel Gradient

Compute M matrix and get score R. Recall subsection 2 in section 2, $R = det(M) - k(trace(M))^2$. I set k value as 0.04. Then set a threshold(named α) and make all R value larger than this threshold is corner. In this experiment I choose the very popular way to set this threshold:

$$R_{corner} \geq \alpha \cdot R_{max} \tag{26}$$

Draw corners as figure 3. From figure 3 we can see the result is not very good. When I zoom in one of a corner area in this picture(shown as figure 4), it is clear to see that multiple pixel in adjacent locations are detected as corner.

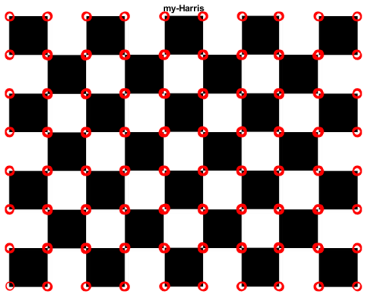

Figure 17: Corners

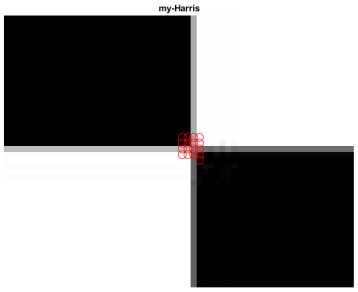

Figure 18: Better Corners

In order to solve this problem, non-maximal suppression is introduced. The way is using a score function as the criterion to select the best corner, The main method is:

- Build a window which normally size is 3 by 3 or 5 by 5 and detect whether if there are more than one corners in this window.
- compare their V values(equation 5) each other, then discard the one with lower V value. choosing the one which has highest V value.

$$V = max \begin{cases} \sum(pixel\ values - p) & if(value - p) > t \\ \sum(p - pixel\ values) & if(p - value) > t \end{cases} \tag{27}$$

- Repeat this procedure until all corners in adjacent locations in this image are compared.

After adding non-maximal suppression method, the results is shown as following.

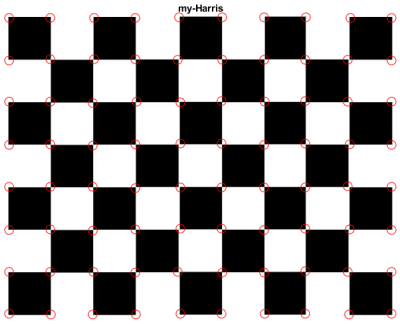

Figure 19: Better Corners

From the figure 5 we can obvious see that the result is much better than before.

1-a Discussion

Change parameter in Harris algorithm and observe the result. I found that increasing k value result in the decline of R value, which will decreases the sensitivity of corner detection. That means number of corners that are detected are decline, Vice versa.
α value is increased, the result would lose some corners (figure 21). So it is important to find a good threshold to get a good result.

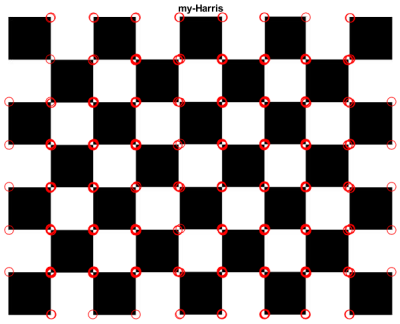

Figure 20: A Worse result

3.2 Feature Macthing

- Task1: Feature matching based on FAST detector and BRIEF descriptor.
- Task2: Feature matching based on SIFT and KD-tree by using VLFeat.

2-a Task 1

In order to make compute process more fast, I divided the process that find a set of n pixels into two parts:

- Compute I_1, I_5, I_9, I_{13} firstly. If at least three of pixels are above $I_p + t$, then got to next part. If not, discard this center pixel.
- Compute all intensity of pixel in the circle, center pixels which has at least nine pixels in the circle are above $I_p + t$ are corners.

Results is shown as follow:

Figure 21: Image 1

Figure 22: Image 2

Now generate a brief pattern. N value is selected as 128 and, for reducing the noise-sensitive, size of smoothing kernels is 9×9 pixels. This kernels size produce good performance than others.[**?**]

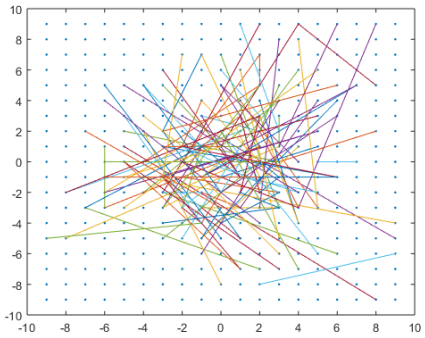

Figure 23: BRIEF Pattern

Then using Brute-force matcher to find matching pairs. This is a simple way that select a descriptor in first image and perform the distance test with each descriptors which in second image, then return to two descriptors that have the closest distance. Repeat this process until all descriptors in first image are matched.
Distance test method is Hamming distance. The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.[] More specifically, descriptors based on BRIEF are bitstring. By using Hamming distance, it's very easy to calculate distance between each other. Plot matching pairs as following. We can see there are some wrong matching pairs even using distance test. To minimize the number of wrong matching pairs, some optimization will be introduced in next section.

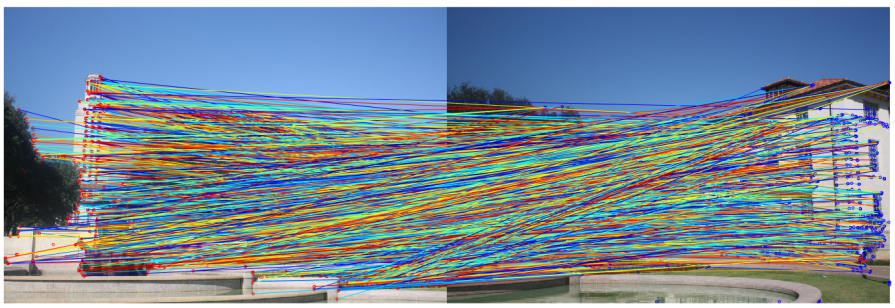

Figure 24: Feature Matching

2-b Task2

Using VLFeat open source library to complete feature matching based on SIFT. This is a open source library of computer vision. I complete the SIFT feature matching by call the function in this package. The input image is color, but the SIFT detector function require gray scale image. So I convert the test image form color to gray. Then extract SIFT keypoints and descriptors. Randomly select 50 features and plot it. When I get descriptor, overlay the descriptor by using a plot function. The visible results are shown as following.

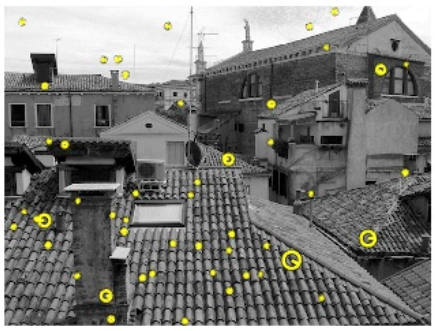

Figure 25: Test Image 1

Figure 26: Test Image 2

Import other image data set and use SIFT algorithm to match feature. From the figure 27 we can see there are some mismatch. The matching function I use in this library is to find the closest distance between descriptors by using L2 norm, which also called Euclidean distance.

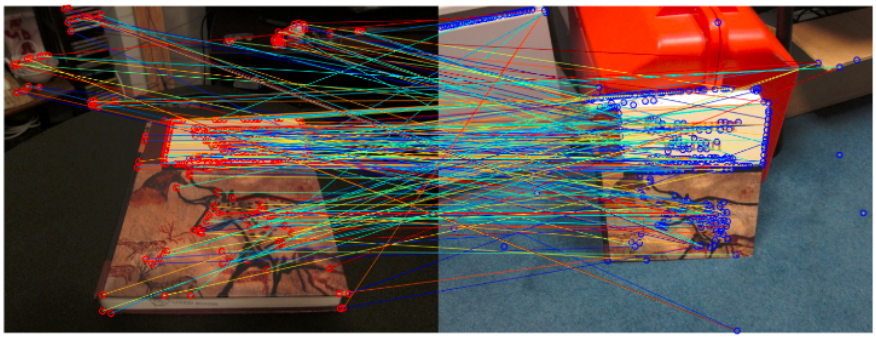

Figure 27: SIFT Matching

3.3 Estimation based on RANSAC

- Task 1: complete RANSAC for line estimation and plane estimation.
- Task 2: Use RANSAC to do optimization of feature matching.

3-a Task 1

In my experiment, in order to improve computational efficiency, I use homogeneous coordinate to compute the distance from points to line and from points to plane. The line which determines by two points x_1, x_2 is ,

$$l = x_1 \times x_2 \tag{28}$$

For the plane case, imagine a set of n points determine a plane ($ax+by+cz=0$). Thus,

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ & \cdots & \\ x_n & y_n & z_n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{29}$$

Equation 15 can also be written as the form

$$Ax = 0 \tag{30}$$

Then plane equation can be compute by finding nullspace of A.
For the same purpose as before, improving computational efficiency, a kind of adaptive RANSAC is introduced. The main idea of it is according to the proportion of inlier in whole dataset to adapt change number of iterations. Using w to describe the probability that select an inlier from the dataset:

$$w = \frac{number\ of\ inlier}{number\ of\ dataset} \tag{31}$$

Generally, w is unknow. Suppose that model estimation needs n points and w times iteration. Thus w^n is the probability of all n points are inliers, $1 - w^n$ is the probability that at least one point in these n points are outlier. So $(1 - w^n)^k$ is the probability that in k times iteration, at least one of n points we select are outlier. Set p is the probability that this algorithm is useful, we can get:

$$1 - p = (1 - w^n)^k \tag{32}$$

Thus we can get k:

$$k = \frac{\log(1 - p)}{\log(1 - w^n)} \tag{33}$$

In my experiment I set p value as 0.99. The advantages of RANSAC is that it can robust estimate model. But the result is limited by number of iteration. the probability of getting trusted model is proportional to number of iteration, While if iteration's number is too less, then we can't get precise model from dataset. Also, we need to set threshold relate to problem. Besides, RANSAC only can estimate one model from one dataset.
After adding above method, the line estimation result is shown as figure 13.

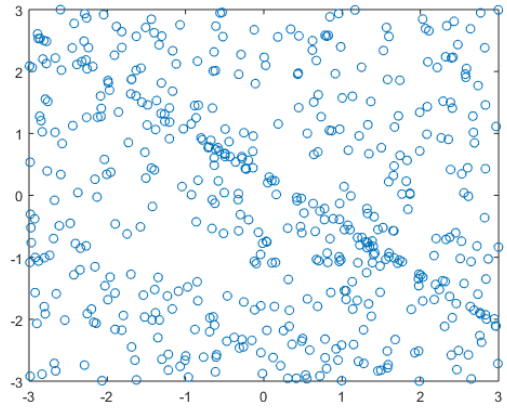

Figure 28: Line Dataset

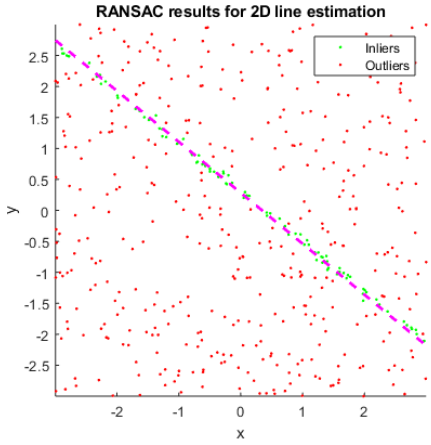

Figure 29: Line

Plane estimation result shows by figure 15.

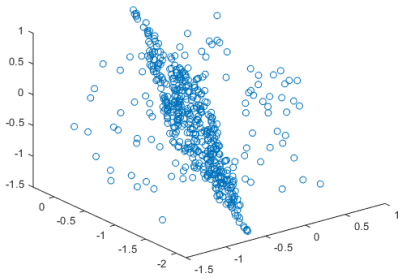

Figure 30: Plane Dataset

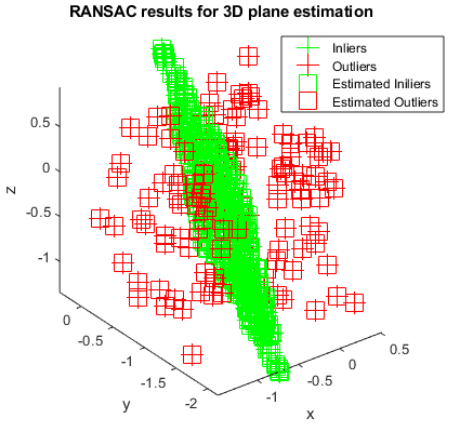

Figure 31: plane

3-b Task 2

In order to make estimation result more stable, normalization can be used. The procedure is very simple, just makes interest point to shift to its center point and then scale transform it to make mean distance of each point to center to $\sqrt{2}$. The criterion of inliers is the sum of two distance: x_1 to projection of x_2 in image 1 and x_2 to projection of x_1 in image 2. Here I set threshold as 1. Results shown as figure

Figure 32: Feature Matching with RANSAC

Compare with the matching results in section 3, It is obvious that by using RANSAC algorithm almost all mismatching pairs are discarded.

3.4 Image stitching

- Task 1: Two image stiching use FAST feature detection and BRIEF feature description.
- Task 2: Complete multiple image stitching by using feature detection and BRIEF feature description.

Firstly do feature matching as before, then compute homography matrix. Because of basic DLT algorithm use no more than 4 matching pairs to compute H matrix, there are a method make result more better, which is normalization. The main idea of it is that transform all elements of A matrix to the similar scale. It can be complete by follow Two steps:

- Normalize the set of matching pairs by compute the similarity transform T and T'. It can translates the centroid and scales such that the average distance from the origion is $\sqrt{2}$.

- Denormailze to get the homography : $H = T'^{-1}\hat{H}T$

I stitch two images firstly, and then compute the matches with third image. The results is shown as following.

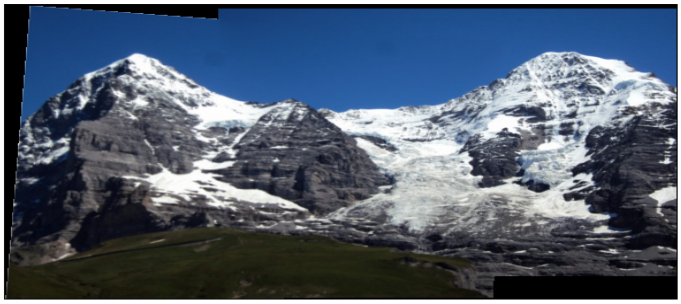

Figure 33: Two Images

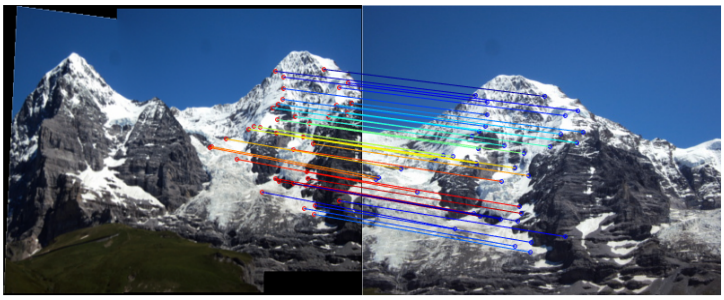

Figure 34: Matching

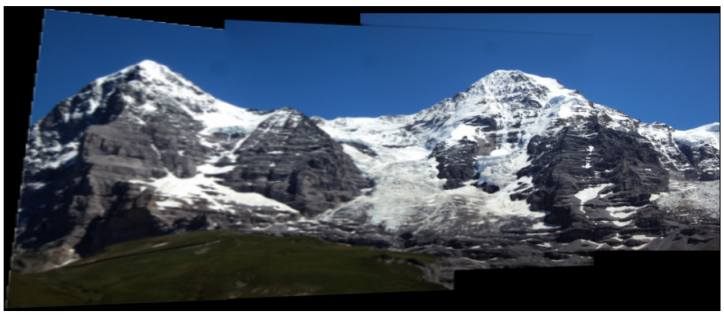

Figure 35: Stitching

Figure 36: Multiple Image stitching

For three image stitching, I use two different ways which already discuss in last section (2.7): one of method is stitching image one by one, another method is that find the center image. These two method perform the same result.

3.5 Discussion

The result I got first time is as figure 37.

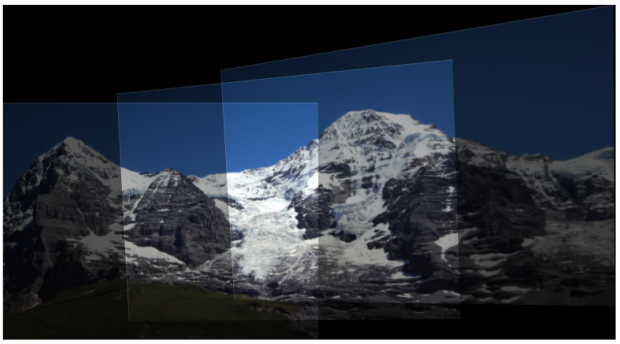

Figure 37: Feature Matching with RANSAC

After I divided the weights that adding in the calculation process, the whole image shows the same intensity. That's because although the intensity are almost in raw image, but the weights change their intensity. The weights are used at image blending.

3.6 Evaluation Results

In this part I use bar chart to describe result and do some analysis.

6-a Scale Changes

In this section we evaluate the matches for image scale change and image rotation. For collection process of KITTI data set , camera are fixed on the car and takes continuous picture when the autonomous car moves at road, so the image that has been taken do not have rotation and images which has large interval will totally different. For the evaluation I choose two images as a group, a total of ten groups be used the evaluation. To analyze result, I compute the mean of five groups. The evaluation of survival rate, accuracy are shown in figure 32 and figure 33.

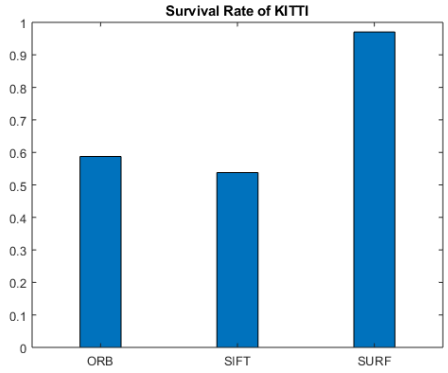

Figure 38: Survival Rate of KITTI

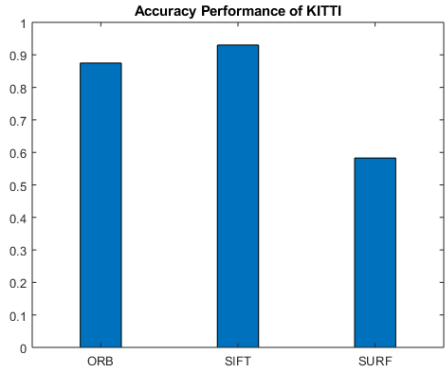

Figure 39: Accuracy Performance of KITTI

In survival rate performance, SURF produce the good performance, almost all matches are survive. But is doesn't has same performance in accuracy evaluation. It seems that SURF algorithm only got more than half a bit of the correct matches, while SIFT got most correct matches, which is about 95% .
For the speed performance, SIFT and SURF spend almost same time to get matches form data set. The time that ORB needs is longest, which is almost 7 seconds.

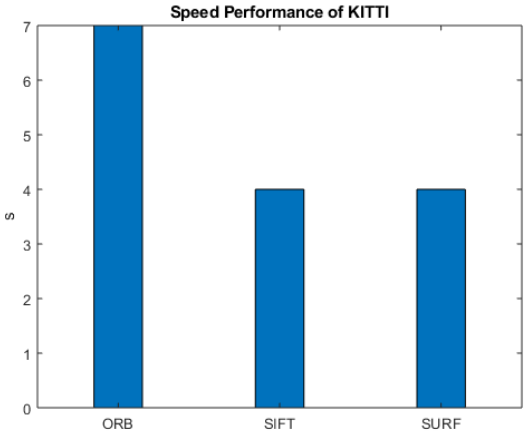

Figure 40: Speed performance for KITTI

The rotation degree in figure 12 and figure 13 is 48°. We can observe that SURF gives best results on survival rate. The confusing result is that the accuracy of SIFT is more than 1. Recall the second criterion, I use inliers' number to divide the whole number of survival matches. So in my opinion this results occur because even using homography to enhance distance test, there are still some mismatch.
The image has a significant influence on the accuracy of ORB algorithm. Although ORB has the shortest responding speed, it is almost no correct matches.

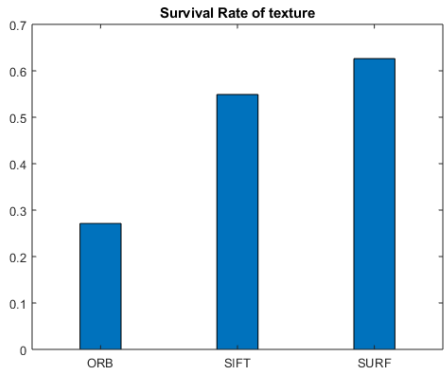

Figure 41: Survival Rate of Texture

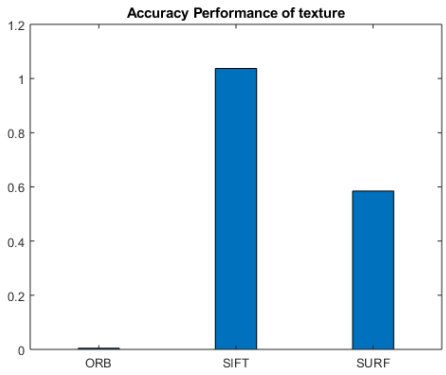

Figure 42: Accuracy Performance of Texture

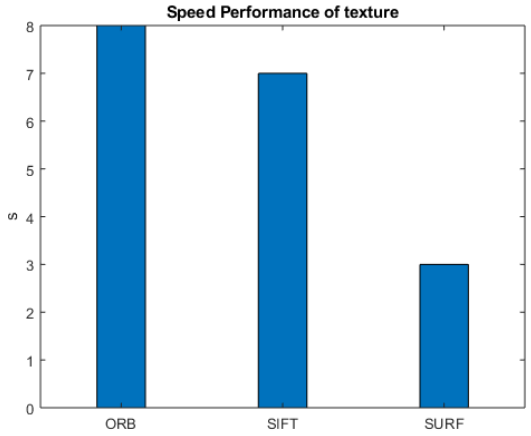

Figure 43: Speed performance for Texture

Figure 14 and 15 is the database that combined rotation and scale change. From the following figure it is obvious that although SURF gives the better survival rate, accuracy performance of SIFT is higher than others. In this time, computational speed is almost same.

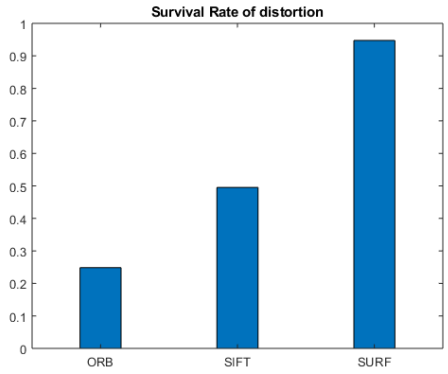

Figure 44: Survival Rate of Distortion

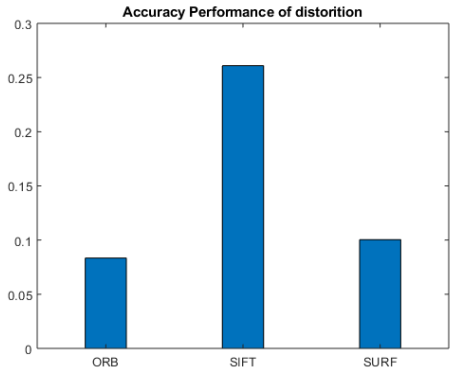

Figure 45: Accuracy Performance of Distortion

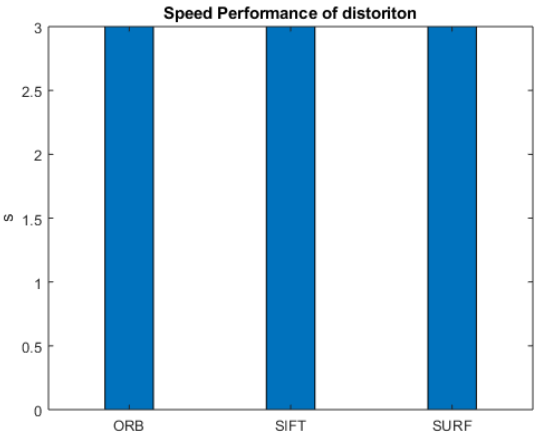

Figure 46: Speed performance for Distortion

From above result we can have the conclusion is that for image that has scale change, SURF always can reserve more

features and SIFT generally has high accuracy for feature matching. ORB algorithm doesn't has a good performance in this case. In summary, the best feature tracking algorithm for database which has most scale change image is SIFT.

6-b Image Blur

This section is evaluation for blur. Blur was introduced by changing the camera focus. In the evaluation based on image blur, the survival rate and accuracy shows similar performance as KITTI data set: SURF got more matches than others, but SIFT produce the highest accuracy. ORB algorithm produce commonly performance, but better as before.

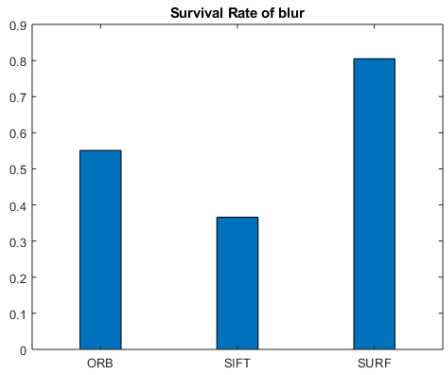

Figure 47: Survival Rate for Image Blur

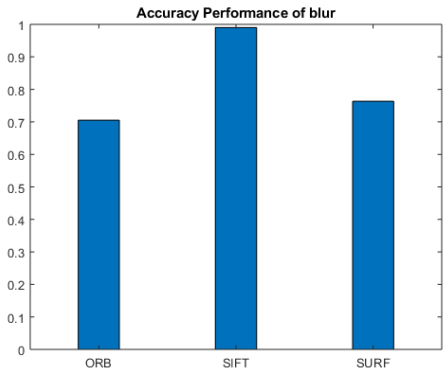

Figure 48: Accuracy Performance for Image Blur

But to computational speed, SURF algorithm costs more times to get matches under this data set. It needs more than three times than ORB and SIFT.

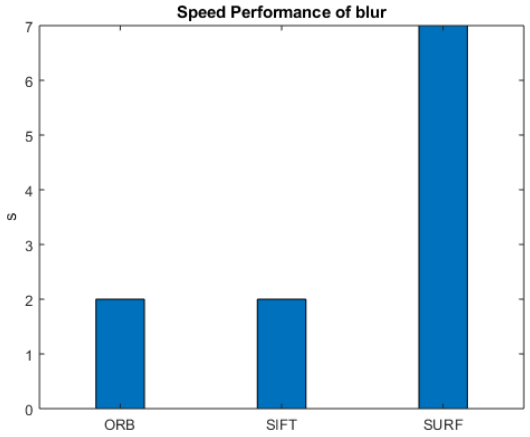

Figure 49: Speed Performance for Image Blur

Actually three algorithms are all affected by image degradation. For other types database all three algorithms can get thousands of interest points but in this database they only get hundreds of interest points.
In summary, for database which has most of image blur, it's better to use SIFT to do feature tracking.

6-c Illumination Changes

In this section I evaluate the images which have different intensity. We can see as same as before, SURF has highest survival rate among three algorithms. But in this data set, ORB algorithm got more correct matches than others. However, ORB consume more times in order to produce this good accuracy.

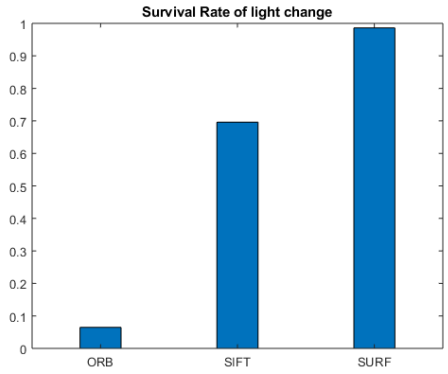

Figure 50: Survival Rate of Light Change

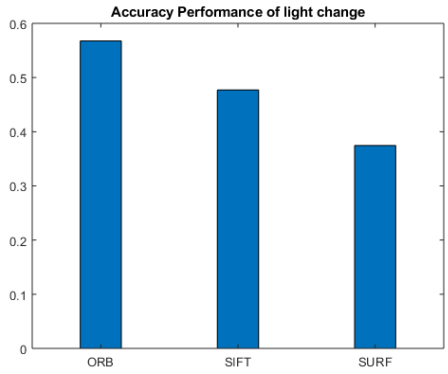

Figure 51: Accuracy Performance of Light Change

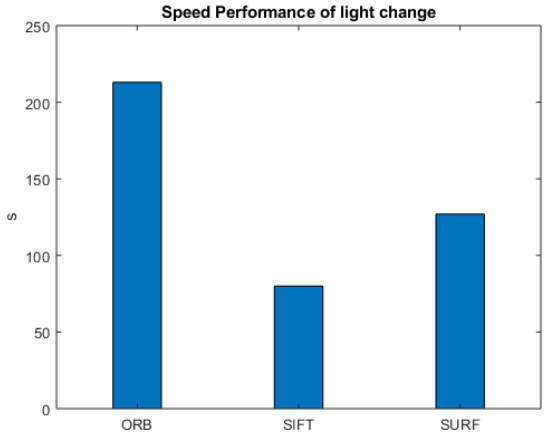

Figure 52: Speed Performance of Light Change

For feature tracking in visual odometry, with the moving of camera there are a part of feature would lose. So it' better to get as more as possible in a image, so that there are more features can be matched. Besides, In this data set the computational time is more higher than evaluation with other data set. That's because I take this data set by my mobile phone. So it is more larger than other data set.

6-d Affine Transformations

In this section I evaluate the performances for view changes of less than 90 degree. This introduced the perspective transformation which can locally be approximately by an affine transformation. There are also some scale and brightness changes in this data set. From the figure we can see the SURF produce the highest survival rate and SIFT produce the highest accuracy. ORB algorithm has lowest performance among three algorithms.

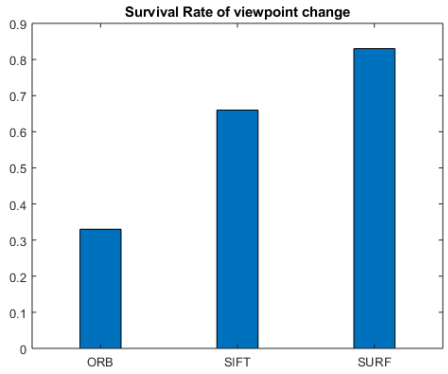

Figure 53: Survival Rate of Viewpoint Change

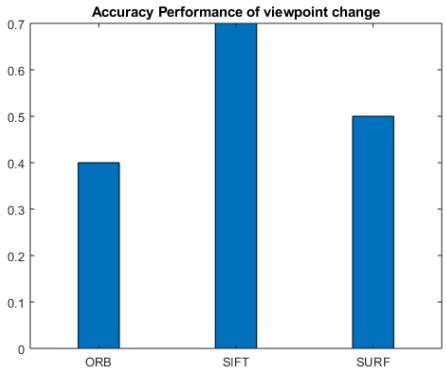

Figure 54: Accuracy Performance of Viewpoint Change

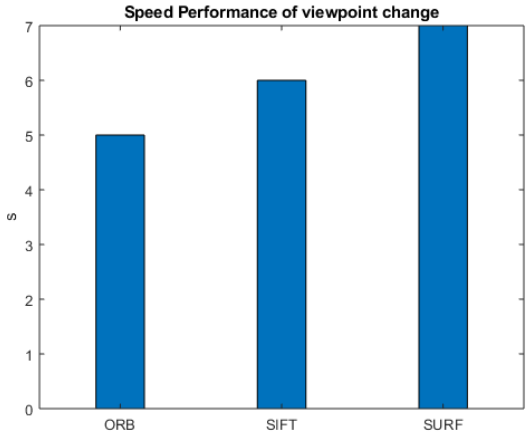

Figure 55: Speed Performance of Viewpoint Change

Sec.4 Conclusion

4.1 Experience

In this part I will make a summary for the period of doing experiment.
From the completion of Harris Corner Detector, the advantages of Harris Corner Detector can be summarised.

- Harris algorithm is easy to calculate.
- Interest points extracted from Harris algorithm are stable, because Harris Corner Detector is insensitive for image rotation, light changes, noise and viewpoint changes.

Harris algorithm has some limitations. For example,it doesn't have scale-invariance and affine-invariance. Also, this algorithm are limited by threshold.
In feature matching experiment, compare with two matching result, although my experiment result is that the combination of FAST and BRIEF is higher than the accuracy of SIFT, from the result of others experiment by using more larger database and more optimal matching method, the performance of SIFT is better than the combination of FAST and BRIEF. Actually, They both have advantages and disadvantages.
For FAST detector, the computation speed is more quick than other corner detect algorithm, but when image has noise, the robustness is not very good. And its result is mostly rely on the threshold. Also, due to FAST detector doesn't generate multi-scale feature and doesn't have direction information, so FAST detector doesn't have rotation invariance.
For BRIEF descriptor, It speed up the descriptor generation process, and its binary descriptor makes feature matching more easy. But, as same as FAST detector's disadvantage, BRIEF descriptor doesn't satisfy rotation invariance. So feature matching that based on FAST detector and BRIEF descriptor doesn't have rotation invariance.
For SIFT algorithm, the advantages are:

- It is insensitive for rotation, scaling and the change of illumination. Its interest point can keep stable under the noise, affine transformation and the change of viewpoint.
- It is very suitable for matching feature based on massive database.
- For several number object it still can generate huge number of interest points.
- High computation speed.

SIFT algorithm still has drawbacks. To obtain high accuracy interest points by using SIFT, images should have enough texture. If image has big flat area, results tend to have many mismatch. There are some other feature extraction and feature matching algorithm, I evaluated these algorithm and will discuss their performance at next section.
RANSAC are utilized to eliminate the mismatch and the effect is very significant. For more specifically, RANSAC can not only be used as the optimization of feature matching, it also can be used to other area. RANSAC algorithm is actually a voting mechanism. Through the voting we can get a model that most data satisfy this model.It are suitable for all database that has disturbance to get the effective sample. But there are still some drawbacks in RANSAC algorithm.

- Its iteration times doesn't have upper limit. Generally, the more iteration leads to more higher precise model. If set the upper limit of iteration, it is possible that the model is not optimal, or even wrong model.
- A threshold that is relevant to this model must be set.
- It only can get one model from one database. If there are more than one model are suitable for this database, RANSAC cannot find other models.

RANSAC has some improvement method. Except the one that I used to find iteration times, it also can make some criterion to select the threshold. Supposed that outliers are satisfy Gaussian white noise, its mean is zero and error residuals are satisfy n-dimensional chi-square distribution. Then error threshold can be selected according to the equation

$$t^2 = x_n^{-1}(\alpha)\sigma^2 \tag{34}$$

Here α is confidence probability. But this improvement didn't solve its drawbacks, which just improve the computational efficiency and model accuracy.
For image stitching, The main difficulty I encountered was the misunderstanding of image warping and blending. After doing image warping and blending, I need to build the panorama and mapping all image need to stitch in this panorama. The blending and warp just makes images warp, and makes images are more smoothly connected.

4.2 Performance Evaluation Conclusion

I compared the classical feature extraction and feature matching algorithms on a benchmark designed to assess their performance for visual odometry. I found that from the overall result SIFT algorithm gives the best performance than others in each kinds of databased. SURF produces similar performance in some database. Amongst scale changes, image blur and illumination changes. But its responding speed is slower than SIFT. ORB cannot resist image rotation, but it gives similar performance in scale changes and image blur.
A part of my experiment conclusion is contrary to the results of other classical benchmark experiments, that's because my database is not large enough. After test significant amount of images the result can be more robustness.
However, based on my evaluation result, I would to say SIFT algorithm is more suitable for visual odometry than SURF and ORB.