Content

1. Introduction

Visual odometry is the process of determining the position of the robot using a sequence of images. The term of visual odometry was used to show that we estimate the pose of the robot like the wheel odometry but by using the visual information. Visual odometry is an important task in visual robot navigation. Some visual odometry approaches are appearance-based and the others are feature-based. The feature-based approaches are more accurate and faster than appearance-based algorithms. As a result, the feature-based methods are more favorable.(Kassir and Palhang, 2017)

In feature-based visual odometry, the features are detected in a two image frames in a sequence and after that the motion is estimated from calculating feature correspondence in two image frames. A typical feature-based approach first extracts a sparse set of salient image features (e.g. points, lines) in each image and then match them in successive frames using invariant feature descriptors. Next, it robustly recovers the camera motion and structure using epi-polar geometry, and finally refines the pose and structure by reprojection error minimization.(Fu *et al.*, 2017) Estimation can be calculated using a single camera (monocular system) or by using stereo camera. Using stereo images will provide more depth and scale information. In general feature-based visual odometry contains three phases. In the first phase features are detected, in the second phase features are tracked or features are matched, finally in the last phase motion is estimated.

2. Methodology

The approaches used in this project can be sorted into two main categories: monocular vision method and binocular vision method.

The camera and vehicle frames are defined as what is shown in Fig. 2.1.

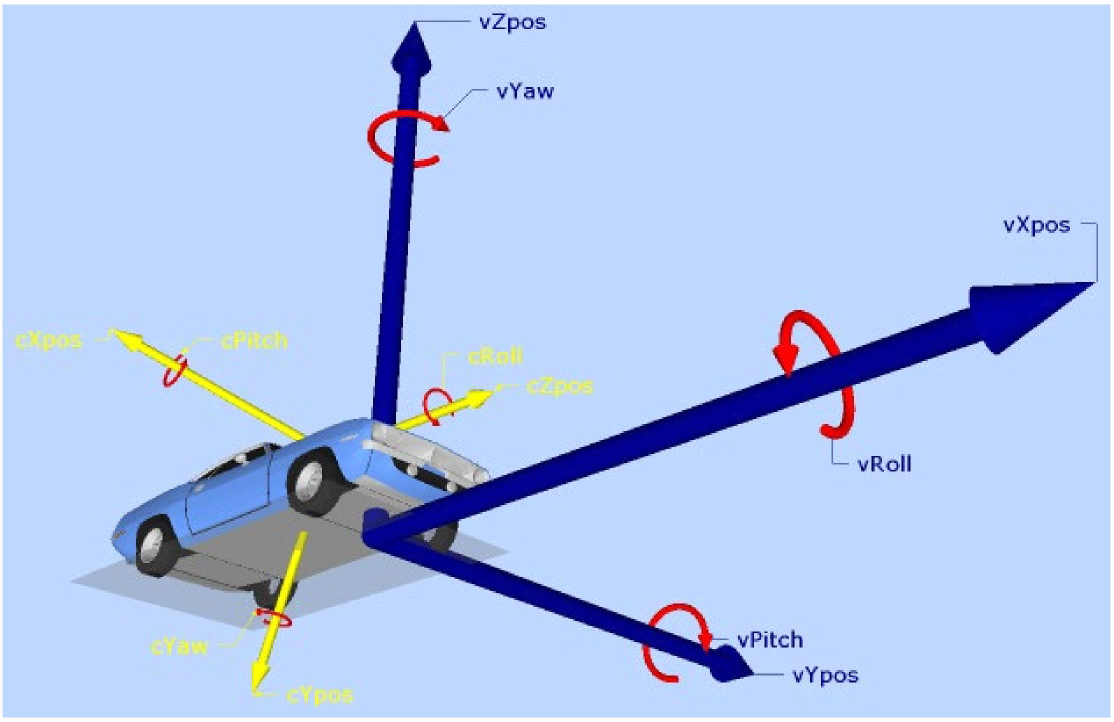


Figure 2.1

2.1 Binocular Vision

In general, four main steps are required to obtain the motion of camera between frames. These steps are: a. feature points detection and matching, b. 3D points triangulation, c. bundle adjustment (BA), d. pose reconstruction

2.1.1 Feature Points Detection and Matching

This step is almost the most common and fundamental step in computer vision and visual-based robots. There are several kinds of popular feature points, e.g. SIFT, Harris corner point, SURF, FAST. Given its lower computational cost, the Harris corner point is applied. And this algorithm is also well suited for the project. Firstly, all frames of both left-hand camera and right-hand camera are detected. For each frame, the 400 strongest corner points are saved for matching. And a 7x7 patch of surrounding pixel values is used as the feature vector.

Next, feature points of left-hand camera and right-hand camera for the same time step are matched according to the descriptor similarity. To lower the chance of incorrect point correspondences, the Random Sampling and Consensus (RANSAC) algorithm is used. In brief, this algorithm chooses eight corresponding points randomly to estimate the fundamental matrix. And this fundamental matrix is tested among all the other corresponding pairs. After iteration, the matrix having the most supporters is returned and all its supporters are termed inliers. After this process, only the inliers of point pairs are kept for future use.

2.1.2 3D Points Triangulation

Since the position of one camera relative to the other is known, it is not hard to reconstruct the 3D position of the point pairs. It is assumed that the optical axes of two cameras are parallel. The geometry is shown in Fig. 2.2.

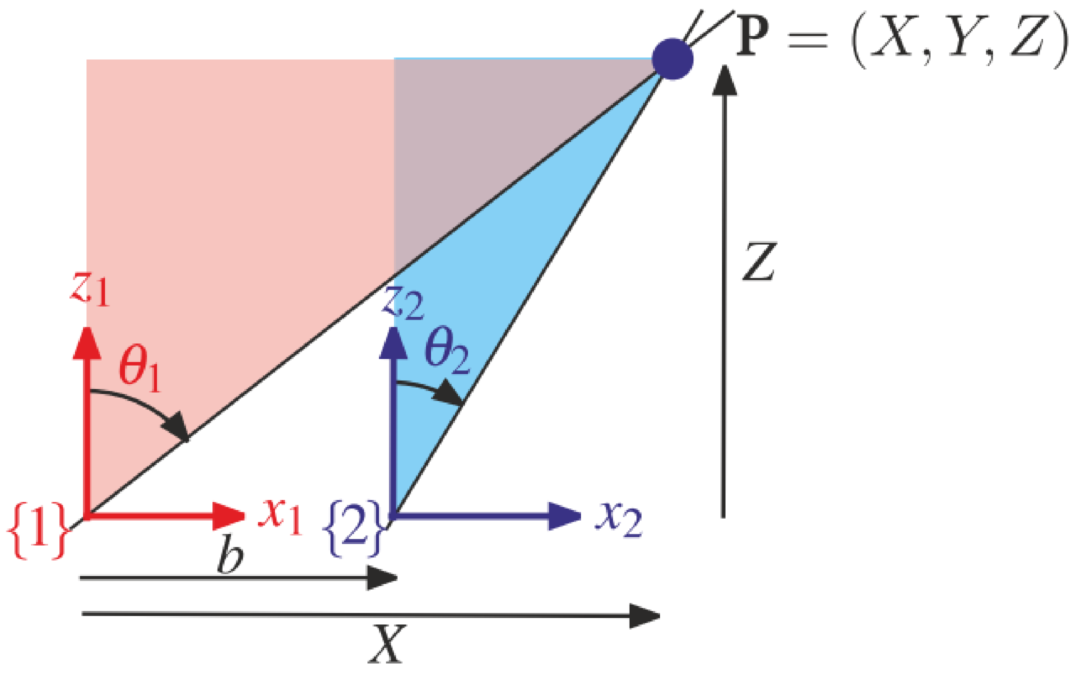


Figure 2.2(Corke, 2017)

According to this geometry, we can get the following MATLAB equations:

```
p1 = p(1:2,:); p2 = p(3:4,:);
d = p1(1,:) - p2(1,:);
 X = b * (p1(1,:) - u0) ./ d;
 Y = b * (p1(2,:) - v0) ./ d;
 Z = f * b ./ d;
```

where p1 and p2 denotes the coordinates of landmarks with respect to left-hand image plane and right-hand image plane. The 3D position of landmark obtained by these equations is respect to the left-hand camera frame.

2.1.3 Bundle Adjustment (BA)

After step 2, all 3D coordinates of landmarks are finally known. It is natural to consider using the 3D-3D approach like iterated closest point (ICP) to determine the change of camera or vehicle pose. However, it works poorly in practice. (Corke, 2017) Thus, bundle adjustment is preferred. The main idea of bundle adjustment is to find the optimal camera pose change that minimizes the error of reprojection of the 3D landmarks at the current time step into the previous time step with respect to the observed landmark coordinates. And the search for optimal pose change is iterated. In this project, BA is done for just two adjacent time steps. The correspondence between landmarks in three images is shown in Fig. 2.3.

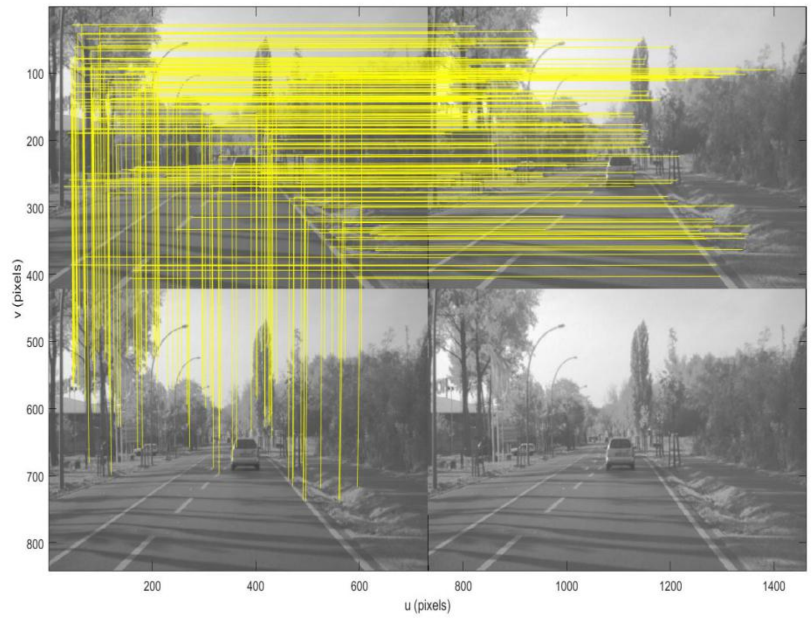


Figure 2.3

One of the key input variables of BA function is the 3D positions of landmarks in current time step. There are two approaches of obtaining them. The first method is straight forward. All 3D coordinates for current frame is derived by stereo vision algorithm as discussed in section 2.1.2. Another method is EPnP. Instead of using stereo vision, the 3D coordinates of landmarks in current frame are derived by their 3D positions in previous frame and projection in current frame. According to (Lepetit and Pascal, 2009), when the number of points used to estimate pose is over 20, there is a chance of over 80% that parameter 'N' in EPnP algorithm is 1. Thus, we set N to 1, which simplifies the algorithm. Besides, the Gauss-Newton optimization is also not applied for simplicity.

The number of iterations is set to 10, which keeps a good balance between accuracy and computational cost.

2.1.4 Pose Reconstruction

The BA functions can directly return the homogeneous transform matrix between two time steps. At each time step, the position of the camera in previous time step is regarded as the reference frame. As a result, the pose of camera at each time step is derived by the following equations:

$$TT_i = TT_{i-1}T_i$$
$$Pose_i = TT_i[0\ 0\ 0\ 1]^T$$

where TT_i is the homogeneous transform matrix between the first time step and time step i. T_i is the homogeneous transform matrix between time step *i* and time step *i-1*. $Pose_i$ is the pose of camera at time step *i*.

2.2 Monocular Vision

For both of the following monocular methods, only the left camera's data of the stereo vision dataset was used to generate the visual odometry. The corner features of last frame and current frame were extracted by the 'icorner' function from the Machine Vision Toolbox and then matched on both frames.

As for the RANSAC function method, it estimates the fundamental matrix by choosing randomly eight corresponding points and creates a model. The model that has the most supporters will be the fundamental matrix we choose.

```
%Calculate fundamental matrix
Fund = matched.ransac(@fmatrix,1e-4, 'verbose');
```

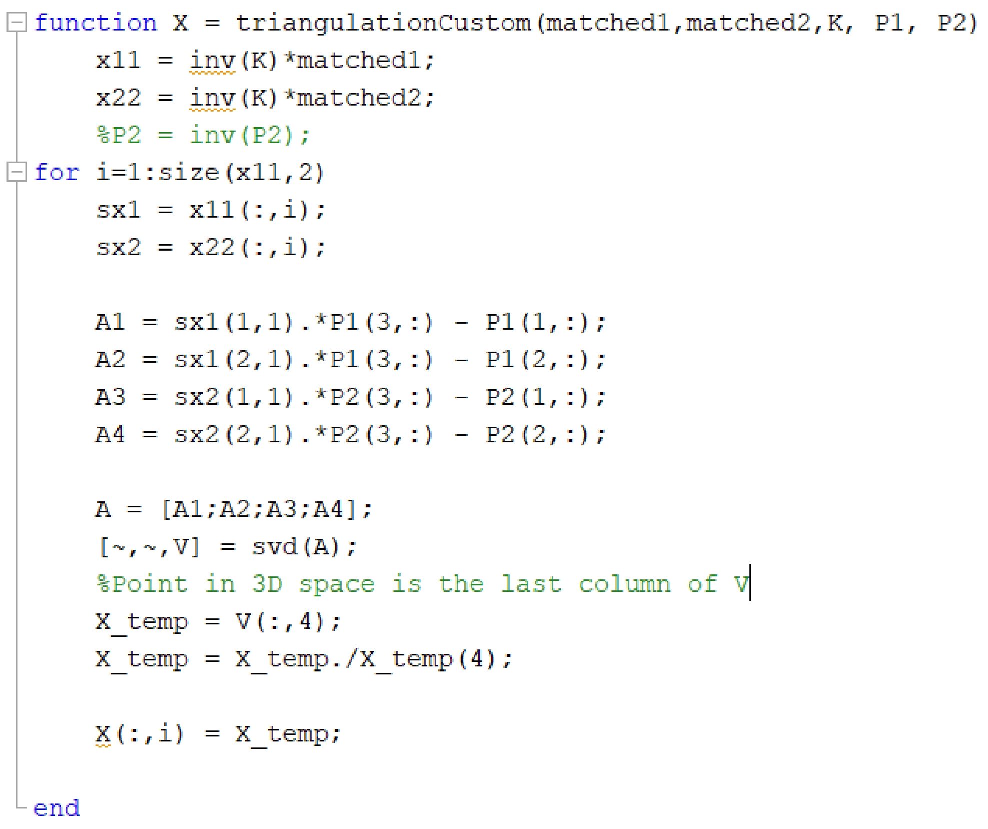
Once we got the fundamental matrix, the essential matrix will be calculated by using the intrinsic matrix. By using the Machine Vision Toolbox, two solutions of rotation matrix R and relative translation T will be calculated.

```
%Calculate essential matrix
E = K' * Fund * K;
sol = cam.invE(E);
sol1 = double(sol(1));
sol2 = double(sol(2));
```

Theoretically the solution that has positive translation on z-axis should be chosen, because the dataset implies that the vehicle does not move backwards. However, the calculated result and generated plot show that this criterion of choosing R and T is not correct. Therefore, there should be four possible combinations of transformation matrix P from the two Rs and two Ts, and each of the transformation matrix P should be tested by some triangulation 3D points.

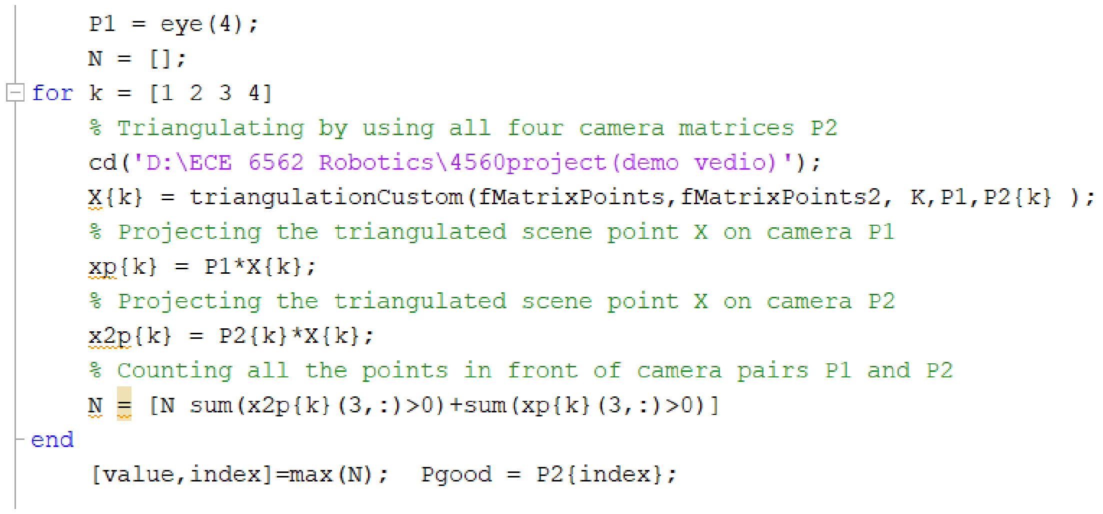
```
R1= sol1(1:3,1:3); R2 = sol2(1:3,1:3);
if (det(R1)<0)
    R1 = -R1;
end
if (det(R2)<0)
    R2 = -R2;
end
t1 = sol1(1:3,4);t2 = sol2(1:3,4);
P2{1} = [R1 t1;0 0 0 1];
P2{2} = [R1 t2;0 0 0 1];
P2{3} = [R2 t1;0 0 0 1];
P2{4} = [R2 t2;0 0 0 1];
```

From the matched points, choosing 30 pairs of matched points then triangulate them by transformation matrices P1 and P2. Since the P1 matrix is at the origin pose, set it as an identity matrix.



```
function X = triangulationCustom(matched1,matched2,K, P1, P2)
    x11 = inv(K)*matched1;
    x22 = inv(K)*matched2;
    %P2 = inv(P2);
for i=1:size(x11,2)
    sx1 = x11(:,i);
    sx2 = x22(:,i);

    A1 = sx1(1,1).*P1(3,:) - P1(1,:);
    A2 = sx1(2,1).*P1(3,:) - P1(2,:);
    A3 = sx2(1,1).*P2(3,:) - P2(1,:);
    A4 = sx2(2,1).*P2(3,:) - P2(2,:);

    A = [A1;A2;A3;A4];
    [~,~,V] = svd(A);
    %Point in 3D space is the last column of V
    X_temp = V(:,4);
    X_temp = X_temp./X_temp(4);

    X(:,i) = X_temp;

end
```

After getting the triangulated 3D points, both of P1 and P2 transformation matrices would be used to transform the 3D points to camera coordinates, and then check which combination will result in the most positive answers on z-axis on both camera 1 and camera 2 (previous frame and current frame). The P2 transformation matrix that has the most votes will be selected as the final solution.



```
    P1 = eye(4);
    N = [];
for k = [1 2 3 4]
    % Triangulating by using all four camera matrices P2
    cd('D:\ECE 6562 Robotics\4560project(demo vedio)');
    X{k} = triangulationCustom(fMatrixPoints,fMatrixPoints2, K,P1,P2{k} );
    % Projecting the triangulated scene point X on camera P1
    xp{k} = P1*X{k};
    % Projecting the triangulated scene point X on camera P2
    x2p{k} = P2{k}*X{k};
    % Counting all the points in front of camera pairs P1 and P2
    N = [N sum(x2p{k}(3,:)>0)+sum(xp{k}(3,:)>0)]
end
    [value,index]=max(N);  Pgood = P2{index};
```

Finally, the R matrix and relative translation T can be found from the solution P2, if the selected solution of T has negative depth on z-axis, choosing the negative T as the final answer. Also, since it is a monocular solution, the relative translation does not have a scale, the T vector should be normalized. Setting the first rotation pose with an identity matrix and position of T as zero vector, we can get the path by literally using the R matrix and T vector.

```
Rpos = rotGood *Rpos;
tpos = tpos + tGood'*Rpos;

path = [path; tpos];
```

The second method was also based on the eight-point algorithm, but the fundamental matrix was not generated from the Toolbox's RANSAC functions, and the solutions of essential matrix E was calculated explicitly.

From the corresponding matched points of previous frame and current frame, randomly picking eight points and constructing a matrix A. And then we use the SVD method to find the fundamental matrix F. After getting the F, setting the last sigma as zero since the rank of fundamental matrix F is 2.

```
r = randi([1 len],1,8);
fMatrixPoints = p1(:,r);
fMatrixPoints2 = p2(:,r);
  %Points from frame1
x1 = fMatrixPoints(1,1); y1 = fMatrixPoints(2,1);
x2 = fMatrixPoints(1,2); y2 = fMatrixPoints(2,2);
x3 = fMatrixPoints(1,3); y3 = fMatrixPoints(2,3);
x4 = fMatrixPoints(1,4); y4 = fMatrixPoints(2,4);
x5 = fMatrixPoints(1,5); y5 = fMatrixPoints(2,5);
x6 = fMatrixPoints(1,6); y6 = fMatrixPoints(2,6);
x7 = fMatrixPoints(1,7); y7 = fMatrixPoints(2,7);
x8 = fMatrixPoints(1,8); y8 = fMatrixPoints(2,8);
    %Points from frame2
x1P = fMatrixPoints2(1,1); y1P = fMatrixPoints2(2,1);
x2P = fMatrixPoints2(1,2); y2P = fMatrixPoints2(2,2);
x3P = fMatrixPoints2(1,3); y3P = fMatrixPoints2(2,3);
x4P = fMatrixPoints2(1,4); y4P = fMatrixPoints2(2,4);
x5P = fMatrixPoints2(1,5); y5P = fMatrixPoints2(2,5);
x6P = fMatrixPoints2(1,6); y6P = fMatrixPoints2(2,6);
x7P = fMatrixPoints2(1,7); y7P = fMatrixPoints2(2,7);
x8P = fMatrixPoints2(1,8); y8P = fMatrixPoints2(2,8);
```

Given the epipolar constraint, we have: ${x_2}^T E x_1 = 0$, where x_1 and x_2 are the 3D coordinates of feature points. However, this equation is only valid in ideal situation. Therefore, we randomly picking 3 pairs of corresponding points to test the errors for choosing the best F matrix. Running the loop for about 1000 times to choose the best F with the least errors.

```
    r1 = randi([1 len]);
    X_p1 = [p1(:,r1);1];
    X_n1 = [p2(:,r1);1];
    r2 = randi([1 len]);
    X_p2 = [p1(:,r2);1];
    X_n2 = [p2(:,r2);1];
    r3 = randi([1 len]);
    X_p3 = [p1(:,r3);1];
    X_n3 = [p2(:,r3);1];
    e1 = abs(X_p1' * Ftemp * X_n1);
    e2 = abs(X_p2' * Ftemp * X_n2);
    e3 = abs(X_p3' * Ftemp * X_n3);
    e = e1 + e2 + e3;
    if (e < error)
        error = e;
        F = Ftemp;
    end

A  =[x1*x1P x1*y1P x1 y1*x1P y1*y1P y1 x1P y1P 1; ...
  x2*x2P x2*y2P x2 y2*x2P y2*y2P y2 x2P y2P 1;
  x3*x3P x3*y3P x3 y3*x3P y3*y3P y3 x3P y3P 1;
  x4*x4P x4*y4P x4 y4*x4P y4*y4P y4 x4P y4P 1;
  x5*x5P x5*y5P x5 y5*x5P y5*y5P y5 x5P y5P 1;
  x6*x6P x6*y6P x6 y6*x6P y6*y6P y6 x6P y6P 1;
  x7*x7P x7*y7P x7 y7*x7P y7*y7P y7 x7P y7P 1;
  x8*x8P x8*y8P x8 y8*x8P y8*y8P y8 x8P y8P 1;];

  A = double(A);
  [U, D, V] = svd(A);
```

After getting the essential matrix E, the first two of sigma values of E will be set as 1, and the third sigma value of E will be set as 0. Then the rotation matrix R and translation vector T can be calculated explicitly and the 4 combinations of R and T can be constructed.

```
t1 = vex(Ue2*rotz(pi/2)*De2*Ue2');
t2 = vex(Ue2*rotz(-pi/2)*De2*Ue2');

W = rotz(pi/2);
if(t1(3)>0)
    t_temp = t1;
end
if(t2(3)>0)
    t_temp = t2;
end
R_temp1 = Ue2*W*Ve2';
R_temp2 = Ue2*W'*Ve2';
if (det(R_temp1)<0)
    R_temp1 = -R_temp1;
end
if (det(R_temp2)<0)
    R_temp2 = -R_temp2;
end
```

Next, like what we did in the RANSAC method, the four combinations of R and T will be used in triangulation for 30 or 40 pairs of matched points to choose the best combination.

Finally, as is discussed in the first RANSAC method, we can use the best R and T to generate the trajectory.

3. Results

3.1 Binocular Vision

3.1.1 Direct Bundle Adjustment

The trajectory of the vehicle is depicted in Fig. 3.1. Since the cameras are mounted on a road vehicle riding on a flat road, it is reasonable to assume that the trajectory is a 2D path. In general, this trajectory is a straight line, which agrees with the real motion of the vehicle. And the unit of axes is meter.

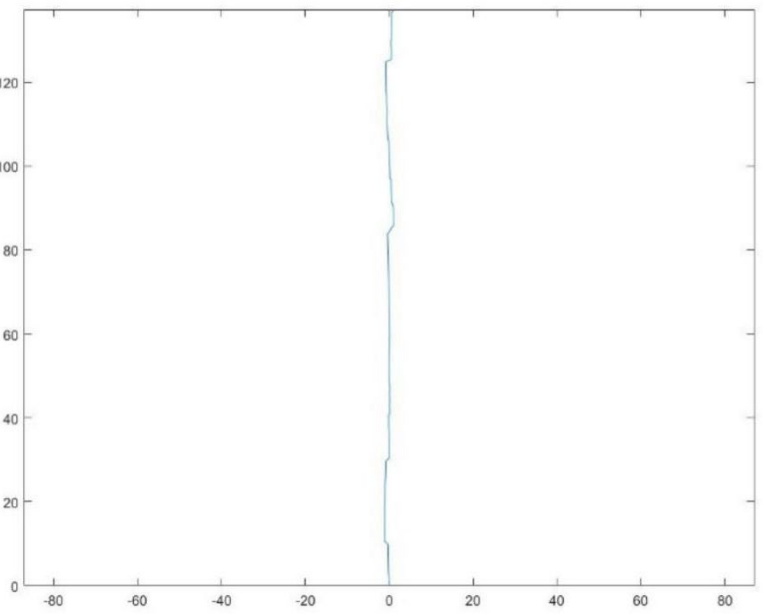


Figure 3.1

3.1.2 Bundle Adjustment with EPnP

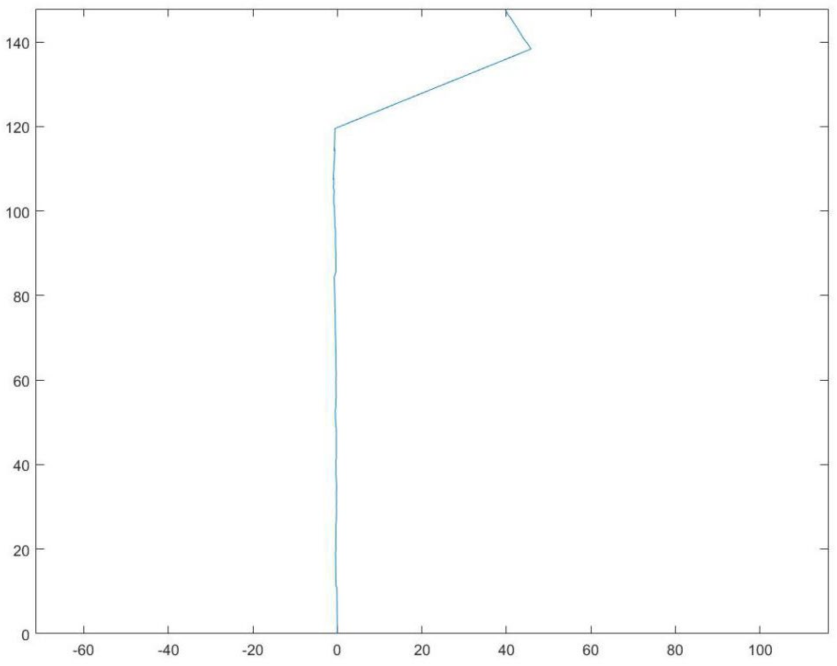


Figure 3.2

As is shown in Fig. 3.2, the path of vehicle is quite smooth at most of time. But there is a sudden tip at the top of the path. This tip is caused by the error in homogeneous transform matrix between frame 227 and frame 228. If we replace the relative motion between frame 227 and frame 228 by the motion between frame 226 and frame 227 before computing the homogeneous transform matrices from first frame to each following frame, the result is satisfactory. And it is illustrated in Fig. 3.3.

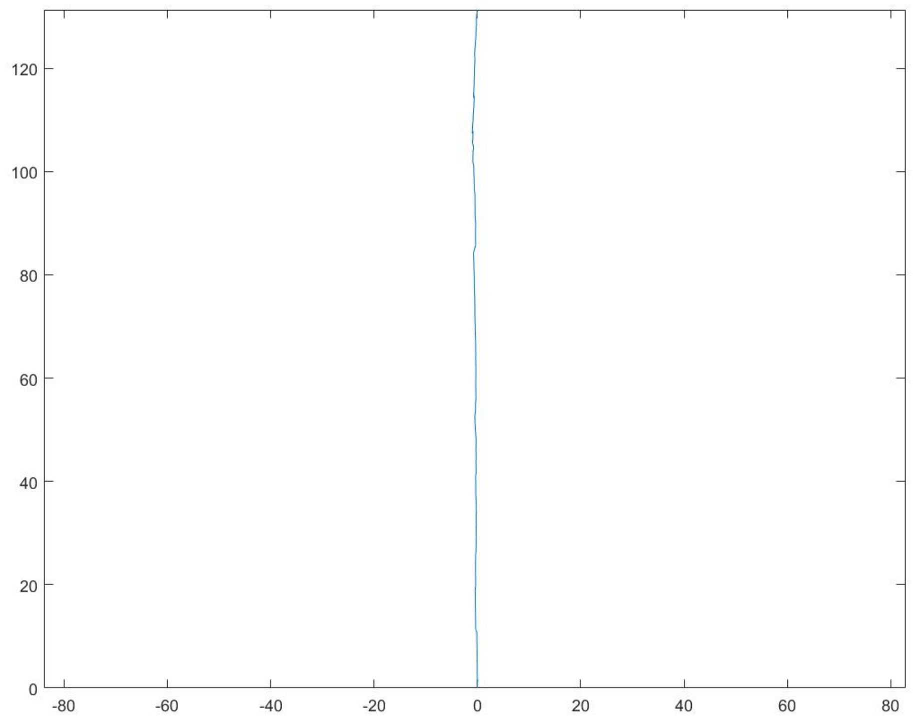


Figure 3.3

3.2 Monocular Vision

The result generated by the first and second method are depicted in Fig 3.4 and Fig. 3.5 respectively.

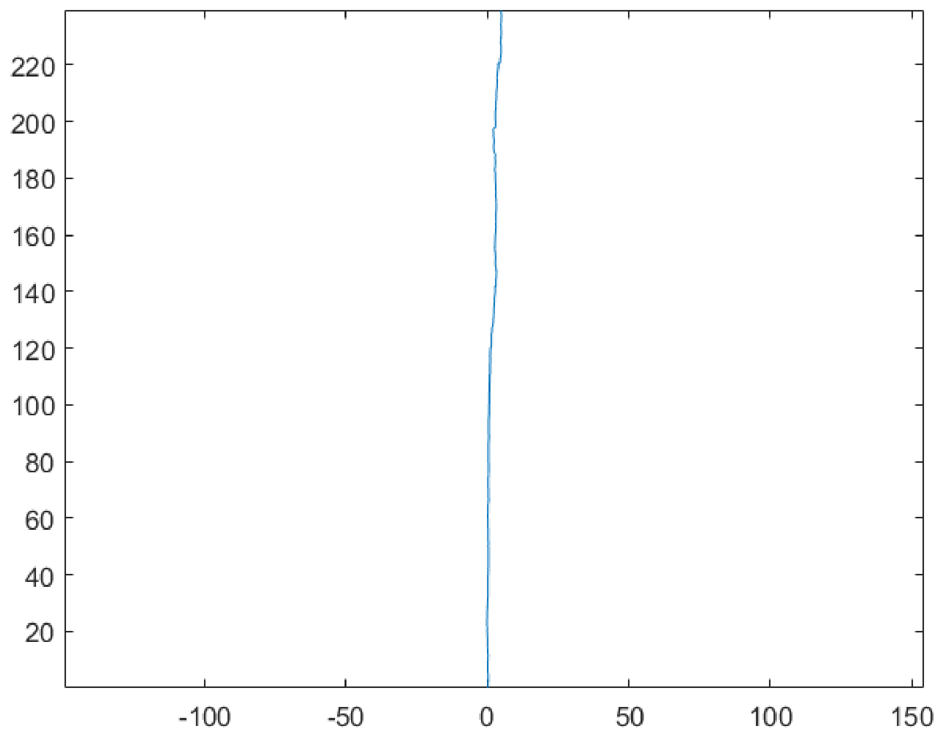


Figure 3.4

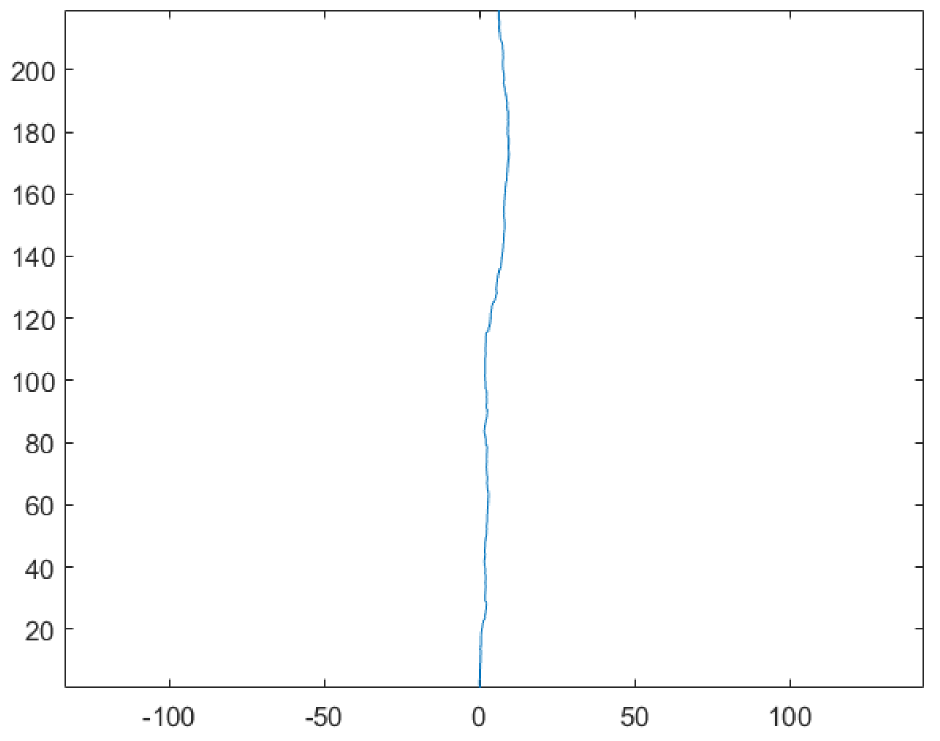


Figure 3.5

As is shown in the figures, the two methods give similar results.

4. Discussion

4.1 Error in Feature Points Extraction

To apply bundle adjustment, it is assumed that all landmarks are fixed in the world. The implicit epipolar constraint inside the RANSAC algorithm is adopted to make sure that only feature points meet the assumption are kept. However, as is presented in Fig. 4.1, some points on the moving car in front are included in the inlier set.

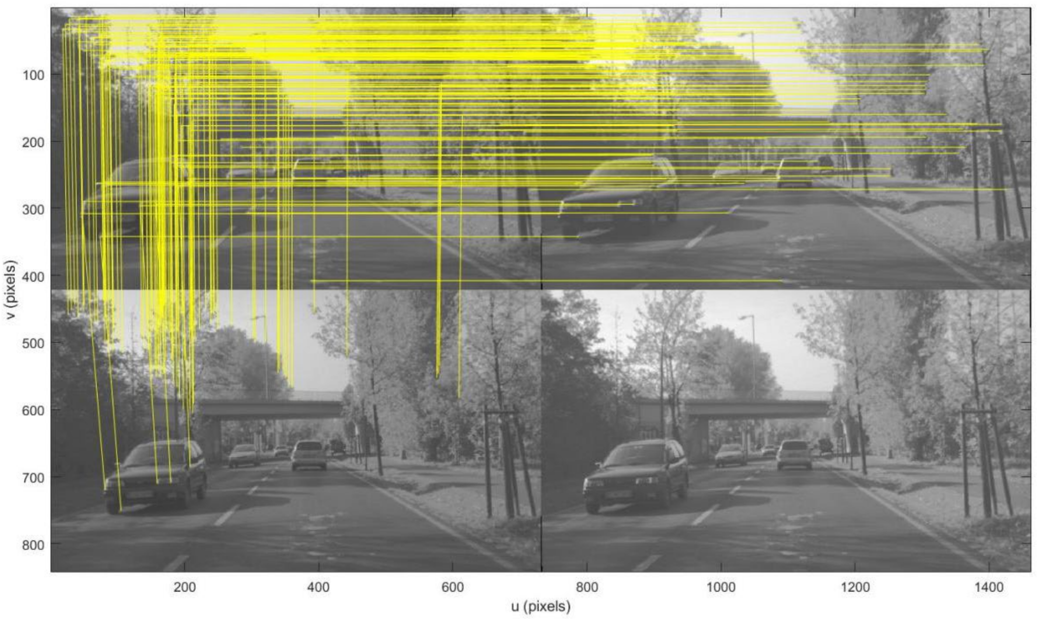


Figure 4.1

Obviously, this introduces systematic errors. A more complicated bundle adjustment algorithm which can detect and reject such points is required.

4.2 Robustness of EPnP

As is discussed in section 3.1.2, the accuracy of vehicle pose can be greatly affected by only one inter-frame error in building the homogeneous transform matrix. And the Gauss-Newton optimization is not considered. Therefore, future work needs to be done on improving the robustness of this algorithm.

4.3 Quality of Monocular-Vision-Based Results

There are another two tests for checking the quality of R and T results and one test for analyzing the quality of fundamental matrix.

For finding the correct combination of R and T, we did triangulation for every combination. We counted how many times the 3D points projected by the best combination in the camera frame has a negative value on z-axis, which means that the best combination still generates a poor result. For this test, normally the RANSAC method has a value of 5 iterations out of all 251 iterations, while the eight-point algorithm has a value of 20.

Another test is to check if the translation vector T has a very small absolute value on z-axis, which means the vehicle doesn’t move forward. And it implies that the fundamental matrix is in ill condition. The RANSAC method always does not have this kind of situation but this happened about 20 times among a series of 251 frames for the eight-point algorithm.

For the eight-point algorithm, the error of testing fundamental matrix is about 0.02 for each iteration. There are also another two tests for testing the R and T result. However, the error of RANSAC method normally is about 0.07 for each iteration. The reason behind is that this error test is the criterion for choosing the fundamental matrix for eight-point algorithm but only a test not the standard for RANSAC method. The result shows that the RANSAC has a better final solution on finding R and T, so the test can help on finding fundamental matrix, but this method still cannot be comparably robust and accurate to the RANSAC method.

Bibliography

Corke, P., 2017. Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised (Vol. 118). Springer.

Fu, Z., Quo, Y., Lin, Z. and An, W., 2017, September. FSVO: Semi-direct monocular visual odometry using fixed maps. In 2017 IEEE International Conference on Image Processing (ICIP) (pp. 2553-2557). IEEE.

Corke, P. (no date) *ALGORITHMS*.

Kassir, M. M. and Palhang, M. (2017) 'Novel Qualitative Visual Odometry for a Ground', pp. 182–187.

Lepetit, V. and Pascal, F. M. (2009) 'EP n P : An Accurate O (n) Solution to the P n P Problem', (June 2008), pp. 155–166. doi: 10.1007/s11263-008-0152-6.

Appendix

To run all the programs, one needs to change the path for dataset in the ‘iread’ function at first.

Vodemo.m is for the direct BA method, epnp.m is for the BA with EPnP, ece6560proj.m is for the RANSAC method and mono.m is for the second monocular method. triangulationCustom.m defines the function ‘triangulationCustom’ used in monocular methods.