EE417 - Assignment 9 - Post-Lab Report

Baris Sevilmis
Lab Date: 18/12/2018
Due Date: 24/12/2018

Overview

Ninth lab assignment of EE417 focuses on 3D structure reconstruction. In the previous weeks, we have focused on correspondence problem and pose recovery. For the pose recovery problem, 8-point algorithm was used in terms of determining Rotation and Translation between the provided views. This week will be second phase such that given the true pose recovery, 3D structure of the provided image will be ensured.

3D Structure Recovery

As mentioned in the Overview, pose recovery must be considered as initial phase in order to recover 3D structure of an image. In order to recover 3D structure out of an 2D image, minimum of 2 different point of views are required, such that their relative poses to each other give information about the depth of different points.

For this lab, pose recovery parameters were provided in terms of focusing on 3D structure recovery, such that Rotation and Translation parameters are given in the following Listing 1:

Listing 1: MatLab Code for Rotation and Translation, Pose Recovery

```
%% Estimated variables in Lab#8
E = [0.0019  -0.3304  0.0018;
     -0.1104  0.0027  -0.9939;
     -0.0045  0.9438  0.0040];

R = [0.9016  0.0030  -0.4325;
     -0.0019  1.000  0.0029;
     0.4326  -0.0018  0.9016];

T = [0.9438;  0.0030;  0.3304];
```

As already known, there are normally two pairs of Rotation and Translation Matrices at the end of pose recovery. However, for simplicity, only the valid Rotation and Translation parameters are given.

Euclidean transformation of a point between two views are given with the following formula, in which γ is a new unknown scale parameter for Translation vector:

$$\lambda_2 x_2 = R\lambda_1 x_1 + \gamma T$$

γ is conclusion of up to scale Translation vector, though it is a new unknown in addition to $\lambda_1 \& \lambda_2$. To reduce amount of unknowns both sides of the equation are multiplied with $\hat{x_2}$, which is skew symmetric form of x_2. Skew symmetric form of x_2 can be seen above

and skew symmetric matrix is produced by a function that can be seen in Listing 2.:

$$\hat{x_2} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

Listing 2: MatLab Code for Skew Symmetric Matrix

```
function x_skew = lab9skew(x)
    x_skew = [0 -1*x(3) x(2); x(3) 0 -1*x(1); -1*x(2) x(1) 0];
end
```

For each correspondence, namely corresponding points between two views, euclidean transformation can be written in the following format including the $\hat{x_2}$ multiplication:

$$\lambda_1^j \hat{x_2}^j R x_1^j + \gamma \hat{x_2}^j T = 0, for j = 1, 2, ..., n$$

j implies correspondence pair number and n indicates total number of correspondences. In order to find scale parameters $\lambda_1^1, \lambda_1^2, ...\lambda_1^n, \gamma$, LLSE, more specifically Linear Least Square Estimation, problem must be solved. Therefore, formula above is transformed into the given format:

$$M\Lambda = \underbrace{\begin{bmatrix} (\hat{x_2}^1 R x_1^1)_{3x1} & 0_{3x1} & \dots & 0_{3x1} & (\hat{x_2}^1 T)_{3x1} \\ 0_{3x1} & (\hat{x_2}^2 R x_1^2)_{3x1} & \dots & 0_{3x1} & (\hat{x_2}^2 T)_{3x1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{3x1} & 0_{3x1} & \dots & (\hat{x_2}^n R x_1^n)_{3x1} & (\hat{x_2}^2 T)_{3x1} \end{bmatrix}}_{(3n)x(n+1)} \underbrace{\begin{bmatrix} \lambda_1^1 \\ \lambda_1^2 \\ \vdots \\ \lambda_1^n \\ \gamma \end{bmatrix}}_{(n+1)x(1)} = 0$$

As it can be seen in the above format, M matrix has size of $(3n)x(n+1)$ and scale matrix Λ has size of $(n+1)x(1)$. When these matrices multplied remaining solution will be equal to size of $(3n)x(1)$. This type of LLSE problem have been solved previously in Camera Calibration Problem and Pose Recovery Problem with Singular Value Decomposition or Minimum Eigen Value Methods. We will be using SVD technique on M matrix. $[U, S, V^T] = svd(M^T * M)$ will compose M into its eigenvectors and smallest eigenvalue corresponds to eigenvector consisting of Λ values. In other words, last column of V^T matrix will contain the Λ vector. Listing 3 ensures essential Matlab Code for the process.

Listing 3: MatLab Code for Scale Recovery

```
M = zeros(3*size(p1,2), size(p1,2)+1);
counter = 1;
for ii = 1:3:3*size(p2,2)
    p2_skew = lab9skew(p2(:,counter));
    M(ii:ii+2, counter) = p2_skew*R*p1(:,counter);
    M(ii:ii+2, end) = p2_skew*T;
    counter = counter + 1;
end
[U,S,V] = svd(M'*M);
lambd = V(:,end);
```

However, Λ is recovered up to a scale, since any multiple of Λ are also solutions satisfying the equation. Next step of algorithm is to multiply scale values with their corresponding points in the first view. Listing 4 demonstrates the MatLab Code for the World Coordinate recovery.

Listing 4: MatLab Code for World Coordinate Recovery

```
pts = zeros(size(p1,1)+1, size(p1,2));
correct_scale = lab9Scale(Tc2c1, T, lambd(end));
for jj = 1: size(p1, 2)
      pts(1:3,jj) = correct_scale*(lambd(jj).*p1(:,jj));
      pts(4,jj) = 1;
end
p1_world = (inv(Hc1)*pts);
```

Original World Points and Two Point of Views are depicted in Figure 1, and Reconstructed World Points are demonstrated in Figure 2.

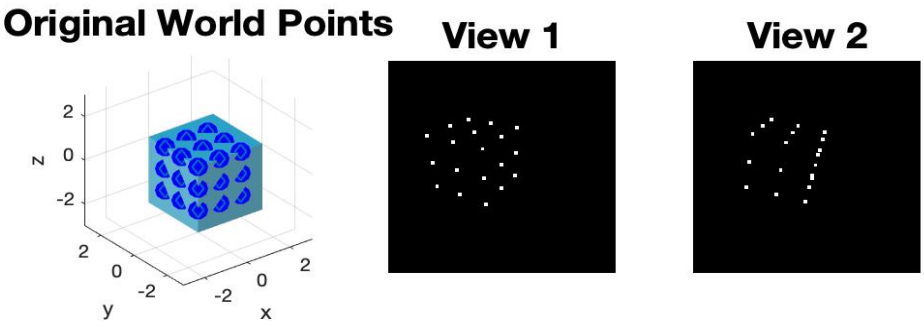

Figure 1: Original Points and Views

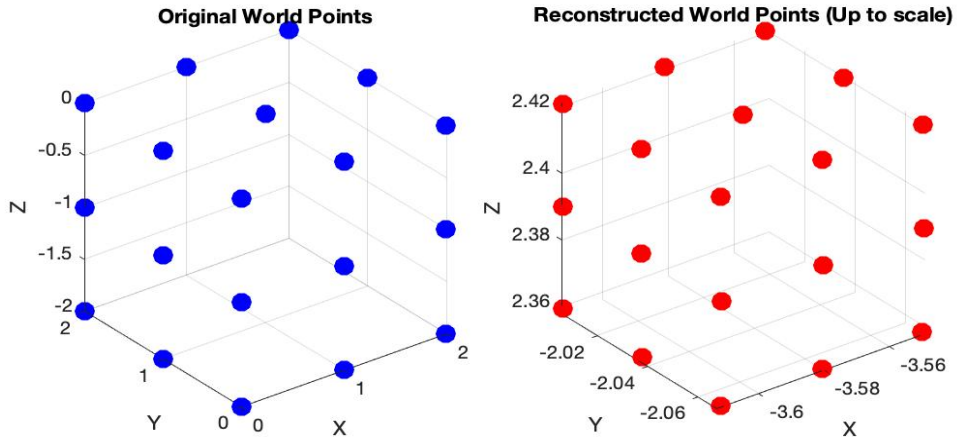

Figure 2: True World Coordinates vs. Reconstructed World Coordinates

It should be clear up until now that reconstructed world points are up to a scale, and therefore not 100% accurate. Figure 2 demonstrates this error as well, and this error is calculated with two different error criterion which are the following:

- Absolute Error Criterion
- Least Squared Error Criterion

Listing 5 provides MatLab code generated for the computation of these errors.

Listing 5: MatLab Code for Error Calculation

```
abserror = sum(sum(abs(P_W-p1_world)));
llseerror = sum(sum((P_W-p1_world).^2));
disp(Absolute Error Criterion: );
disp(abserror);
disp(Least Squared Error Criterion);
disp(llseerror);
```

From our current reconstructed world points, Figure 3 provides errors according to both of our criterions.

Figure 3: Absolute Error and LSE

As it is obvious, error values indicate that scaling are not decent, although they are up to an universal scale. In the case of our lab assignment, we have additional information to estimate our object in a more correct scale. Normally, true translation vector is not ensure, however in our lab assignment we already know the true world scale. One can simply divide T_true values to estimated $\hat{T}$ value in order to obtain true γ such that euclidean transformation takes the following form:

$$\lambda_2 x_2 = R\lambda_1 x_1 + \frac{T_{true}}{T}T$$

$$\lambda_2 x_2 = R\lambda_1 x_1 + T_{true}$$

As true γ is obtained, one can simply get the scale coefficient k from the following equation by simply dividing γ_{true} to $\hat{gamma}$:

$$k\hat{\lambda_1^1} = \lambda_1^1$$

$$k\hat{\lambda_1^2} = \lambda_1^2$$

$$...$$

$$k\hat{\gamma} = \gamma$$

$$k = \frac{\gamma}{\hat{\gamma}}$$

This k value can multiplied with our image coordinates to obtain world coordinates as in Listing 4. On the other hand, MatLab code for finding k is depicted in Listing 6.

Listing 6: MatLab Code for Correct Scale Calculation

```
function k = lab9Scale(Tc2c1, T, G)
    Gc2c1 = Tc2c1 ./ T;
    k = Gc2c1(1) / G;
end
```

Figure 4 demonstrates corrected scale version of world coordinate estimation. Nonetheless, it is clear that coordinate estimation is much more accurate than previous estimation depicted in Figure 2.

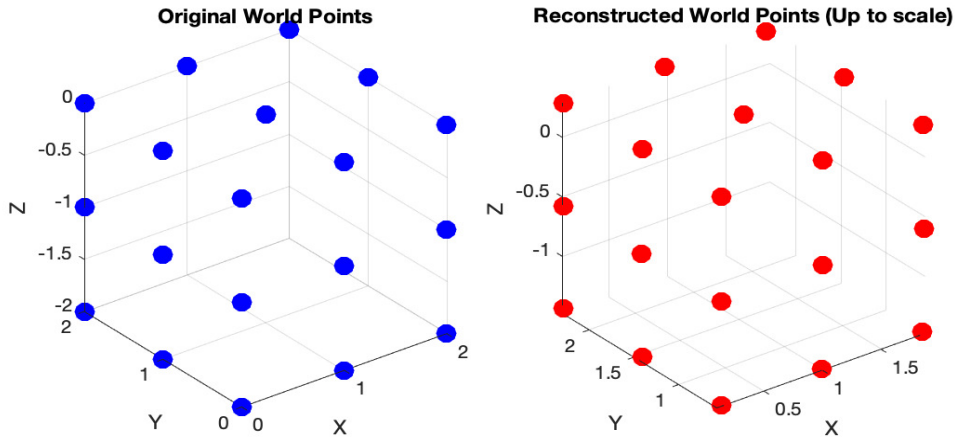

Figure 4: True World Coordinates vs. Reconstructed World Coordinates - Correct Scale

Lastly, error is computed again between true world coordinates and estimated world coordinates as can be referred in Figure 5. Both of the errors are much lower than the error values in Figure 3.

```
Command Window
  Absolute Error Criterion:
     18.0145

  Least Squared Error Criterion
      7.6426

fx >>
```

Figure 5: Absolute Error and LSE - Correct Scale