Fundamentals of Computer Vision Project

Implementation Report

Members:

Setareh Cohan - 91105636
Saeid Naderiparizi - 91110836

Contents

1 Project goals

In this project we aim to reconstruct the 3D scene using a set of 2D pictures of the same scene taken from different angles.

The available data consists of a number of pictures from one scene that contain information about camera's characteristics. Having this dataset, we must reconstruct the 3D scene, along with finding camera's position(t) and rotation(R) for each 2D picture, and calculating matrix containing the intrinsic parameters of camera(K).

2 Implementation

2.1 Explanation

In order to reconstruct the 3D scene from a set of 2D pictures, we followed 6 steps:

1. Finding key points in each picture
2. Matching key points
3. Extracting *fundamental matrix* using matched points and RANSAC algorithm
4. Calculating *essential matrix* from *fundamental matrix*
5. Calculating position(t) and rotation(R) of camera from *essential matrix*
6. Finding the 3D point corresponding each couple of matched points using R and t.

In what follows, we will elaborate upon each step throughly.

3 Loading pictures

3.1 Explanation

All pictures were uploaded from a folder and were saved in a *structure* which is called "dataset", and contains fields listed below:

- **image:** Image itself is stored as a 3D matrix, each photo stored in RGB format.
- **descriptor:** Structure containing all information extracted from pictures. This structure is filled when extracting key points and matching them and has two fields: *features* and *valid points*. *Features* consists of all features extracted from a picture and *valid points* consists of key point's SURFObjects.
- **camera_parameters:** Structure which saves camera parameters that are listed below:
  - length: Picture's length in mm.
  - width: Picture's width in mm.
  - f: Camera's *focal length*.
  - length_pixels: Number of pixels in picture's length.

- width_pixels: Number of pixels in picture's width.
- pixels_per_mm_length: Number of pixels in each mm of picture's length.
- pixels_per_mm_width: Number of pixels in each mm of picture's width.

Output of this part of the code is an array of "cells" of "dataset" type that consists of all pictures information.

3.2 Functions

3.2.1 load_data

This function inputs folder's name and uploads all pictures in that folder into "dataset" and fills "camera_parameters" using each picture's information, EXIF file and project description.

3.2.2 get_input

A function called in load_data which returns all pictures of a folder in an array.

4 Calculating matrix representing intrinsic parameters(K)

4.1 Explanation

In this section matrix K was calculated using camera parameters. Parameters used in this part are "focal length" in mm, and lens' length and width in mm and pixels.
As we know, matrix K is defined as below:

$$\begin{bmatrix} f\delta_x & \gamma & h_x \\ 0 & f\delta_y & h_y \\ 0 & 0 & 0 \end{bmatrix}$$

Based on project description, we know that γ is equal to zero.
Moreover, δ_x is equal to the ratio of lens' width in pixels to lens' width in mm, and δ_y is also equal to lens' length in pixels to lens' length in mm.
h_x equals to half of lens' width in pixels and h_y also equals to half of lens' length in pixels.[3][4]

4.2 Functions

4.2.1 build_K

Inputs a structure of "camera_properties" type and builds matrix K using its fields as explained above. Output of this function is 3 by 3 matrix K.

4.3 Output

Calculated K for the camera that has captures pictures of folder S1-C1 was:

$$K = 10^3 * \begin{bmatrix} 5.5792 & 0 & 2.5920 \\ 0 & 5.5667 & 1.7280 \\ 0 & 0 & 0.0010 \end{bmatrix}$$

5 Extracting each picture's key points

5.1 Explanation

In this section, key points of each picture saved in "dataset" were extracted and filtered. Afterwards, *Feature Vector* of remaining key points was returned and saved as the "descriptor" field of each "dataset".

5.2 Functions

5.2.1 extract_features

This function takes a picture as an input and returns "descriptor" structure of that picture. This output is used for filling "descriptor" field of each "dataset".
This function is called for each picture in a dataset.
We have used MATLAB's "detectSURFFeatures" and "extractFeatures" functions.

5.3 Output

Key points extracted from first picture of S1-C1 folder was:

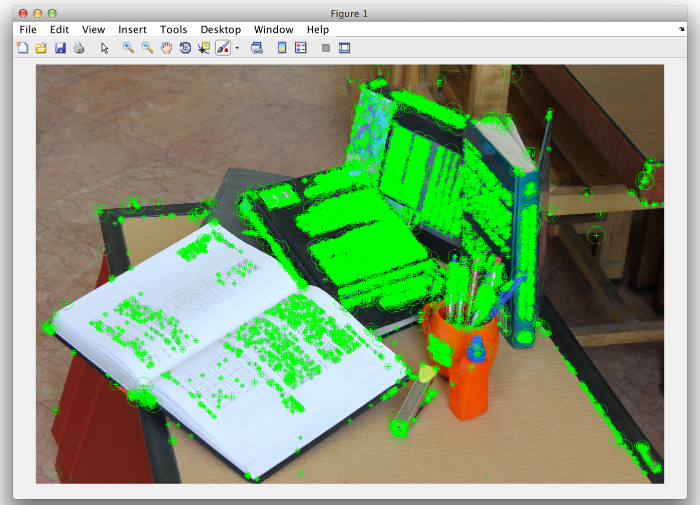


Figure 1: All 12239 key points

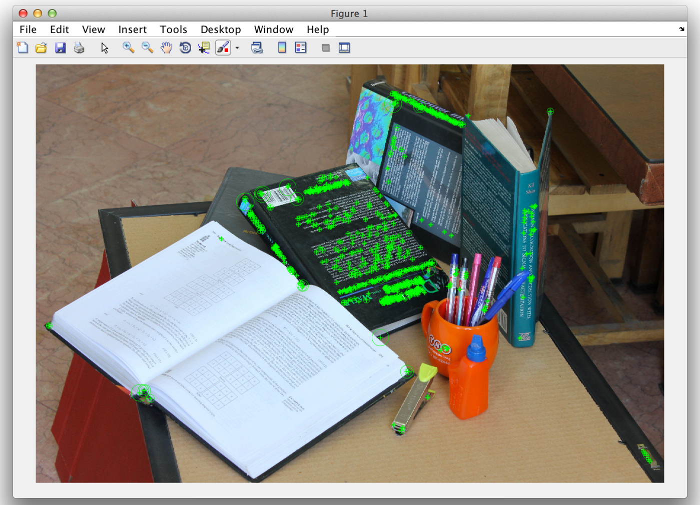


Figure 2: All 700 strong key points remained after filtering

6 Matching key points

6.1 Explanation

For constructing a 3D scene using each couple of pictures, we first needed to extract key points of each of those two pictures which was explained in last section and, match these key points and eliminated ones that have no match. We used MATLAB's "extractFeatures" function for doing so.

6.2 Functions

"extractFeatures" was called by "extract_fundamental_matrix" function which is called by "structure_from_motion" function itself.

6.3 Output

Output of key points matching in first and second picture of S1-C3 folder is delineated below:

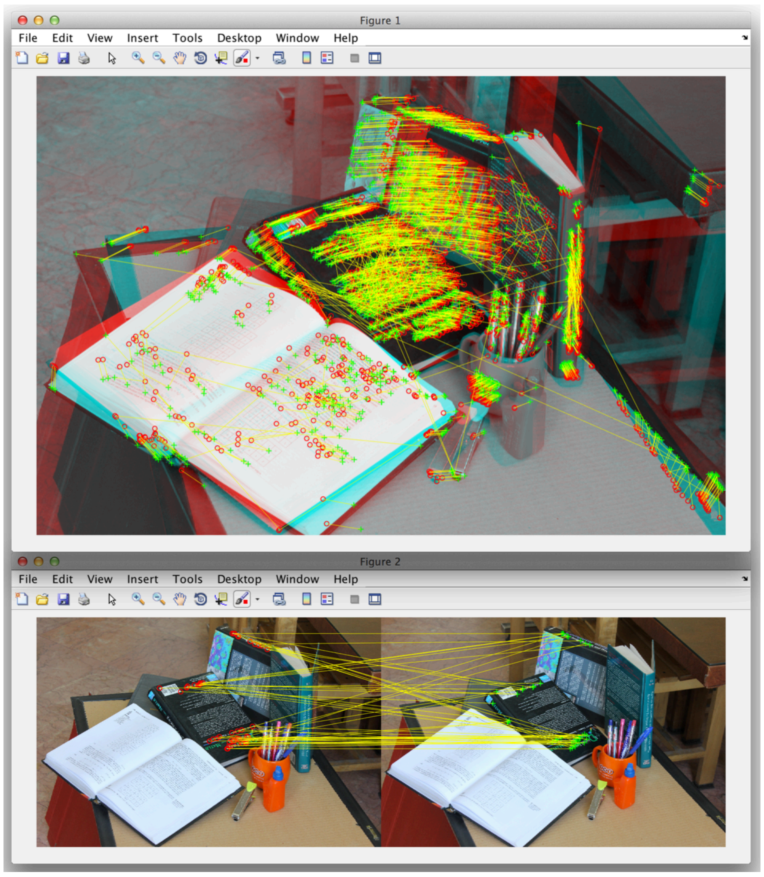


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 2
File Edit View Insert Tools Desktop Window Help

Figure 3: Matching 50 of strongest key points

7 Extracting fundamental matrix using matched points and RANSAC algorithm

7.1 Explanation

Using matched key points, we used RANSAC algorithm for extracting *fundamental matrix*. For doing so, we chose 8 random points in each iteration, and using those and

$$\begin{pmatrix} x_b & y_b & 1 \end{pmatrix} F \begin{pmatrix} x_a \\ y_a \\ 1 \end{pmatrix} = 0$$

we extracted the *fundamental matrix*(F) approximately. After that, we calculated number of inliers for F. Then, using all pair of inliers, F is calculated. We continued these steps until F was found with the desired accuracy.
Since the accuracy threshold and number of iterations were unknown, we used MATLAB's "estimate-FundamentalMatrix" with RANSAC parameter. This function inputs matched key points of each two pictures and returns F and inlier pairs indexes as outputs.[5]

7.2 Functions

As explained above, we used "estimateFundamentalMatrix" in this part of our code.

7.3 Output

Extracted F from matched points of first and second pictures of S1-C1 folder was:

$$F = \begin{bmatrix} 0.0000 & -0.0000 & 0.0005 \\ 0.0000 & 0.0000 & -0.0074 \\ -0.0008 & 0.0076 & 0.9999 \end{bmatrix}.$$

8 Calculating essential matrix from fundamental matrix

8.1 Explanation

We know that

$$F = {K_b^{-1}}^T R[t]_x K_z^{-1}$$

where $[t]_x$ is as

$$[t]_x = \begin{bmatrix} 0 & -Z_0 & 0.Y_0 \\ Z_0 & 0 & -X_0 \\ -Y_0 & X_0 & 0 \end{bmatrix}.$$

Moreover, for *essential matrix* (E) we also know that:

$$E = R[t]_x.$$

Therefore, we can see that having F, we can calculate E from relation 1:[6]

$$E = K_b^T F K_a. \tag{1}$$

8.2 Functions

8.2.1 extract_essential_matrix

We have used "extract_essential_matrix" function which inputs F, K_a and K_b, and returns E as the output.

8.3 Output

Matrix E calculated for first and second pictures of S1-C1 folder was reported as:

$$E = \begin{bmatrix} 0.2936 & -3.7317 & 1.5188 \\ 1.3428 & 0.1402 & -40.7429 \\ -3.7336 & 40.4322 & 0.1422 \end{bmatrix}.$$

9 Calculating camera's position(t) and rotation(R) from essential matrix

9.1 Explanation

We first define matrices Z and W as:

$$Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

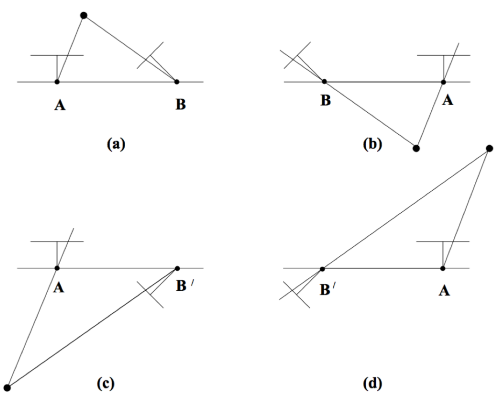
$$W = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then, we normalize E such that its final element (in 3rd row and 3rd column) becomes equal to 1. We name this matrix E_{norm}. With calculating SVD[1] of E_{norm}, vector t and matrix R will be available:

$$E_{norm} = UDV^T \Rightarrow \begin{cases} t = +VZV^T, -VZV^T \\ R = UWV^T, UW^TV^T \end{cases}$$

We can see that what we have at this point, is 4 forms of possible R and ts. However, we know that we should only report one R and one t corresponding each E.

Using class lectures, acceptable R and t are those that cause 3D point appear in front of each camera. To clarify, in the picture below which shows all 4 forms of possible R and ts, only a is acceptable:



For doing so, we calculated the position of our 3D points, using each of the possible R and ts.(Explained in next section) Meaning we had 4 sets of 3D points.
Later, we counted number of 3D points in front of each camera. R and t with the largest number of 3D points in front of cameras were returned as the correct R and t.

[1]Singular Value Decomposition

In order for 3D point X to be in front of a camera, the angle between camera's perspective vector(3rd row of R), and vector $X - C$ with C being camera's position, must be less than 90 degrees.[2]:

$$(X - C).R(:,3) > 0.$$

9.2 Functions

9.2.1 extract_R_t

This function, inputs E and conducts all steps explained above, and returns matrix of all forms of R and t as output.

9.2.2 structure_from_motion

This function, inputs "extract_R_t"'s output and for each of its 4 forms, counts number of 3D points in front of each camera. Then, selects R and t with the largest number of points in front of all cameras and returns them along with the 3D points associated with them.

9.2.3 get_front_points

This function is called from "structure_from_motion". Its inputs are two "dataset"s for each of pictures corresponding to a 3D point, R and t, and corresponding points of two input pictures.
This function uses input information to reconstruct the 3D points and returns those that are in front of both cameras.

9.3 Output

Final output of this section are 3D reconstructed points and the correct R and t.
R and t for first and second pictures of S1-C1 folder were reported as:

$$R = \begin{bmatrix} -0.9884 & -0.1315 & -0.0758 \\ -0.1320 & 0.9913 & 0.0007 \\ -0.0750 & -0.0107 & 0.9971 \end{bmatrix}.$$

$$t = \begin{bmatrix} 1.0000 \\ 0.0940 \\ 0.0087 \end{bmatrix}.$$

10 Calculating the corresponding 3D point for each couple of matched points

10.1 Explanation

Suppose we have the dataset of pictures taken by cameras a and b, R and t of one camera with respect to other one, and corresponding points calculated from matchings.
For the first camera (a), we suppose that it is located int the center of 3-dimensional coordinates and its image plane is parallel to the $x - y$ plane. Therefore, we can say that 3D reconstructed points of this camera have the same x and y as they had in their 2D plane and the z element equals to camera a's focal length in pixels.

Now, for the second camera, we first assume that it has the same location and rotarion as camera a and find the 3D points as above.
All we have to do now is to transfer each 3D point by t and rotate them by R to find the exact 3D points.

10.2 Functions

10.2.1 get_front_points

Function which exactly performs all the steps explained above.

10.2.2 lineIntersect3D

This is obtained from MATLAB's functions.[7] This function has been used to find the location of 3D points by having camera's position and rotation.

11 Building a pointCloud object from 3D points and showing it in Meshlab

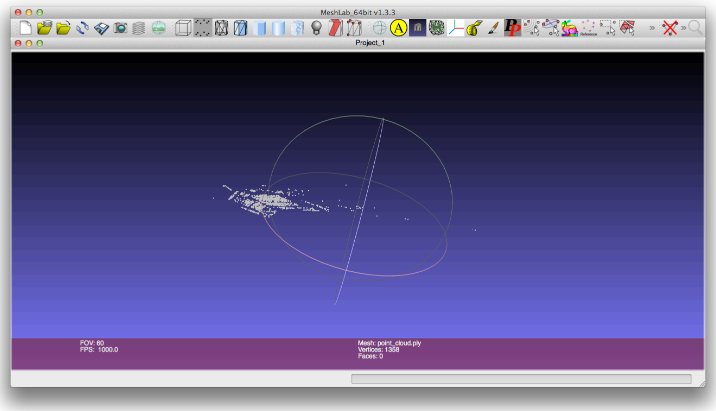
11.1 Explanation

Lastly, using all 3D points, we built a object with pointCloud type. This object was saved in a file using pcwrite command. Afterwards, we showed this object using meshlab software.

11.2 Output

Some of the outputs form meshlab software are brought below:

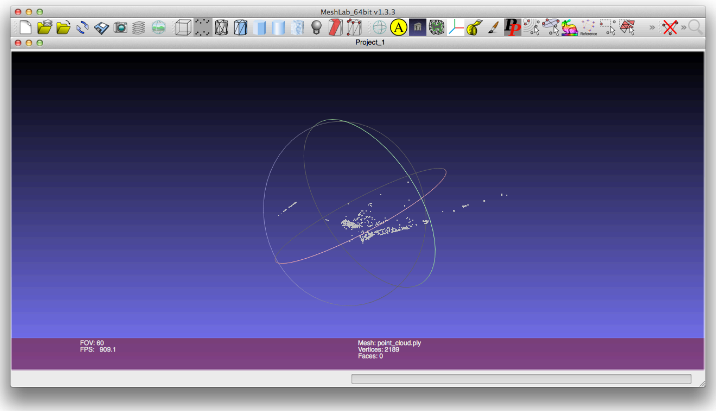
11.2.1 Reconstruction of first and second pictures of folder S1-C1



FOV: 60
FPS: 1000.0

Mesh: point_cloud.ply
Vertices: 1358
Faces: 0

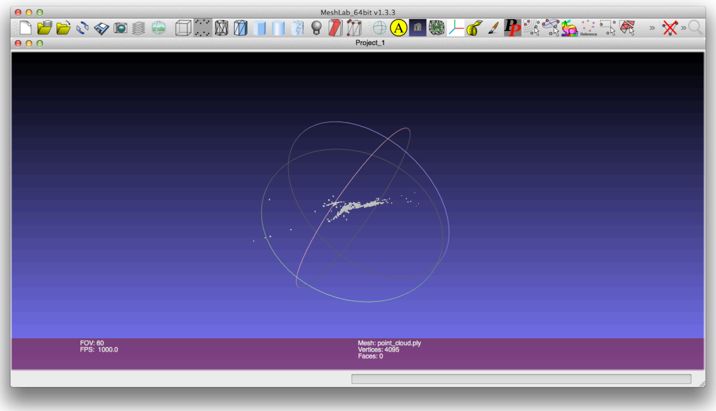
11.2.2 Reconstruction of first to fourth pictures of folder S1-C1



FOV: 60
FPS: 909.1

Mesh: point_cloud.ply
Vertices: 2189
Faces: 0

11.2.3 Reconstruction of all pictures of folder S1-C1



FOV: 60
FPS: 1000.0

Mesh: point_cloud.ply
Vertices: 4095
Faces: 0

12 References

References

[1] Lecture 7_3CV,
Course Lectures.

[2] Vision group of Princeton
`http://vision.princeton.edu/courses/COS429/2014fa/slides/precept04/COS429_ps4_precept.pptx`

[3] Stackoverflow: Essential matrix from fundamental matrix,
`http://stackoverflow.com/questions/23943602/essential-matrix-from-fundamental-matrix-in-opencv`

[4] PRoVisG: Camera calibration,
`http://cmp.felk.cvut.cz/mars/challenge.pl?menu=calib`

[5] Mathworks: Estimating fundamental matrix,
http://mathworks.com/help/vision/ref/estimatefundamentalmatrix.html

[6] Lecture 5_3CV,
Course Lectures.

[7] Mathworks: Intersection point of lines in 3D space,
`http://mathworks.com/matlabcentral/fileexchange/37192-intersection-point-of-lines-in-3d-space`