16-822: Geometry-based Methods in Vision (F17) Released: Oct-21., Due: Nov 11th

Homework 3

Lecturer: Martial Hebert *TA: Dinesh Reddy, Nate chodosh, Martin li*

1 Matching the Answers with Gradescope(5 points)

You get 5 points for correctly matching the answer to the respective question.

2 Theory(65 points)

2.1 Affine cameras (25 Points)

In class, we have mostly focused on perspective cameras. For this part of the homework, we consider cameras with center lying on the plane at infinity, especially *affine cameras*. An affine camera is one that has a camera matrix of the form $M = [A \mid b]$ in which A is singular and the last row of M is of the form $(0, 0, 0, 1)$. The following questions explore affine cameras a little more in depth.

- Given a set of points X_i in $3D$, and their projection x_i in the image, what is the projection of the centroid of X_i's? (no equations, just 1 short line.)
- Show that an affine camera maps parallel world lines to parallel image lines; i.e., when using an affine camera, parallel lines in $3D$ project to parallel lines in image plane.[1]
- Show that for parallel lines mapped by an affine camera, the ratio of lengths on line segments is an invariant.

For the following questions, consider a system of 2 affine cameras, with camera matrix M and M': (Last row of each camera matrix is of the form $(0, 0, 0, 1)$.)

- Show that epipolar planes (and lines) are parallel.
- Show that the affine fundamental matrix F defined by two affine cameras is invariant to an affine transformation of the world coordinates.

2.2 Auto-calibration (15 Points)

1. No equations, use a counting argument only: Suppose that we have m images from which we have generated a projective reconstruction. Suppose that the intrinsic parameters (K_i) are fixed throughout all of the images but unknown except for the skew which is known to be 0. How many images are needed to solve for the calibration parameters (auto-calibration toward metric reconstruction)?
2. Can it be solved in a linear fashion in this case?

[1] Actually, the converse is also true; if the camera preserves parallelism of lines, then it is an affine cameras.

3. Suppose that all the calibration parameters in K are allowed to vary across the images, but that we know that the two coordinates of the principal point are equal in each of the images: $x_o = y_o$. How many images are needed to solve the auto-calibration problem? [2]

2.3 Fundamental matrices between three cameras (25 Points)

In this problem we consider the relationship between 3 images. Let us denote by F_{ij} the fundamental matrix between image i and j (from 1 to 3), and by e_{ij} the epipole generated by image j in image i. We will now show that the 3 fundamental matrices induced by 3 cameras are not independent.

- Draw the 3 cameras with e_{ij} labelled.
- Show that: $F_{ij}e_{ik} = e_{jk} \times e_{ji}$ for all triplets. (this can be shown purely geometrically, without involved algebra)
- Based on these 3 quadratic relations, give a counting argument showing that the 3-camera geometry is characterized by 18 degrees of freedom. And thus conclude that the 3 fundamental matrices are not independent.

3 Having fun by learning from implementing! (35 Points)

In this part of the homework, we will explore 3D reconstruction. Especially, we will study a full pipeline of a typical 3D reconstruction system using Structure from Motion (SfM) and Stereo Matching. Slight different from the previous homeworks, this time you do not implement everything from scratch. Instead, you will delve into one of the existing packages for this purpose (Here we use the *SFMedu* system from the computer vision group at Princeton University). You can use other state-of-the art SFM pipelines as well like OpenSFM(in python) and COLMAP(in C++)) for the assignment by analyzing the results and modifying some steps of particular interest and importance. For you, this is also a very good way to understand the material taught in the class (I can say this from first hand experience). You can use any opensource SFM and MVS system for this homework.

3.1 What you are supposed to do:

You need to both implement and answer questions as follows:

- **Run the SFMedu(or similar) system**
  I have already included the SFMedu package in `./code/` folder. Run the SFMedu system by running the `SFMedu2.m` script in MATLAB. Make sure that it runs successfully and produces a `dense.ply` point cloud file. Install MeshLab from `http://meshlab.sourceforge.net`, and open the point cloud in MeshLab and rotate around the model. Take a screen capture of the point cloud and put it in the report as the answer to this question. An illustration is shown in Figure 1.

[2] FYI, this particular constraint does not correspond to a realistic physical constraint on the cameras, but it is still a valid constraint in theory.

Figure 1: An illustration of the SFM system: Structure from Motion (SfM) and Stereo Reconstruction.

- **List the Major Steps**
  Read the code carefully, especially `SFMedu2.m`. Summarize the major steps for SfM and dense stereo matching, and write it down in the report as answer to this questions.

- **Principal Point**
  What is the assumption for the principal points in this system? Under what condition will this assumption get violated?

- **Data Structure**
  Graph is a data structure used in the code for bundle adjustment. It has 5 major fields. What are their names? What is the data in each field? How is the data organized? Why we need these 5 fields?

- **Reconstruct Your Own Images**
  Capture/Find two sequences of images (at least 5 images per set) on some interesting objects or scenes, run the system on these images. If it fails, figure out why it fails, and change the code or retake good images until it works. Submit the input images, as well as three screen captures of your reconstruction per image set in the report. Explain all the changes or things you have to do to make it work. Which step do you think is the most unstable one? Explain why. List suggestion for a normal user without any computer vision background about how to take good images to make it work.

- **Compute the Reprojection Errors**
  Write a function

  ```
  function printReprojectionError(graph)
  ```

  that takes a graph (as defined in the SFMedu system), and prints out the current reprojection error, so that you can insert the function into many steps for debugging purpose, to check if the reconstruction errors are getting smaller. A call to this function is at Line 160 of `SFMedu2.m`. Submit the function as code. Also, plot the reprojection errors in the report for the image sequence provided and the two sets of images you captured in the report PDF file.

- **Visualize the Reprojection Points**
  Write a function

  `function visualizeReprojection(graph,frames)`

  that takes a graph (as defined in the SFMedu system), and draw the 3D keypoint point cloud projected onto each image, as well as their observed location. Figure 2 shows an example of the output for your function. You are required to write a function to produce the same kind of visualization. Each observed keypoint is represented by a red $\times$, and the reprojection of its 3D estimated location is represented by a green $+$, and these two points are connected by a blue line. For the 3D points that are projected to the image but not observed from the image, it should be shown as a yellow $\circ$. A call to this function is at Line 162 of `SFMedu2.m`. Submit the function as code, and also include the visualization results of the image sequence provided and the two sets of images you captured in the report PDF file.

Figure 2: An example result produced by `function visualizeReprojection(graph,frames)`.

- **Levenberg-Marquardt**
  `bundleAdjustment.m` uses matlab function `lsqnonlin` to minimize the objective function via the Levenberg-Marquardt algorithm:

```
[vec,resnorm,residuals,exitflag] =
lsqnonlin(@(x) reprojectionResidual(graph.ObsIdx,graph.ObsVal,px,py,f,x),
[Mot(:);Str(:)], [], [], options);
```

Please read all the code and figure out what is the objective function it is optimizing for. Write down the math equation of this objective function in your report.

3.2 What you have to submit:

You should submit three files:

- A report that contains all the images (both the images you captured and your result images) and answers to the questions.
- Your CODE folder that contains all source code (including YOUR readme) for your system, and your "SFMedu2.m" script that takes no parameter as input and runs directly in Matlab to generate the results reported in your PDF file.
- Your IMAGE folder that contains all the input, intermediate and output images.

3.3 Tips:

- Make sure to keep "sanity checks" in your code while you are debugging, by checking an equation that you know should be true. (e.g., $l^T C m = 0$). Keep such code commented when you submit, so that I can see and appreciate it!
- Normalize coordinates of points before doing anything. This is important!
- Remember the transformations that you are estimating are up to a scale.
- Keep in mind all the things Martial mentioned in class!
- Start early...
- Have fun!!