COMP 558 Assignment 4

Prepared by Prof. Michael Langer
Posted: Tuesday, Nov. 20, 2018
Due: Tuesday, Dec. 4, 2018 (midnight)

Introduction

This assignment covers some of the material from lectures 14-20 on 3D vision. The assignment is designed to help you reinforce some of the key ideas in those lectures, and so we strongly suggest that you carefully review the lecture material prior to doing the assignment.

We also encourage you to discuss the questions with each other and to give each other hints and tips. *However, the solutions that you submit must be your own work.* You are not permitted to copy code from each other.

Instructions

Submit a single zipped file **A4.zip** to mycourses Assignment 4 folder. The zip file must contain:

- a PDF with figures and text for each question
- the Matlab code that you wrote to generate the figures for each question

In order to receive full points, your solution code must be properly commented, and you must provide a clear and concise description of your computed results in the PDF. Be concise and clear! Note that the TA's have limited time resources and plan to spend about 30 minutes grading each assignment.

Late assignment policy: Late assignments will be accepted up to only 3 days late and will be penalized by 10 points per day, e.g. An 80 grade would be reduced to 72 for one day late. If you submit only a few minutes late, we reserve the right to treat this as "one day late".

Question 1 (25 points)

You are given starter code, and two images **c1.jpg** and **c2.jpg** which show views of a simple scene, namely an IKEA shelf unit. The images were shot with a DLSR camera. The camera was rotated and translated slightly between shots. (There is also a small lighting change between shots, but this is irrelevant to the assignment.)

The 3d object coordinates **XYZ** of several points are given to you. These are the corners of three different squares; two squares are holes in the shelf frame and the other square is a yellow post-it note whose edges are aligned with the coordinate frame of the object. The 3D positions of these given points were estimated approximately using a tape measure, and assuming the frame is rectilinear. Units are in mm. The corresponding image pixel projections **xy** are also given. These positions were obtained by hand labeling using Matlab's Data Cursor. The **XYZ** and **xy** data can be read in using script **readPositions.m**.

The script **Q1.m** reads in the two images, and the 3D and 2D point coordinates. It displays the image and marks the 2D image positions **xy**. We would like to model the camera, so that we can related the positions of any points **xy** and **XYZ**, not just the given ones.

Your task: Write (and submit) the function **calibrate.m** that performs a least squares estimate of the 3x4 projection matrix **P**, given coordinates **XYZ** and corresponding image pixels **xy**. The function file has been provided for you with a code stub. (Do not change the arguments or return type.) Your implementation should call the given function **decomposeP.m** which factors **P** into a product of matrices **K, R,** and translation vector **C,** as in the lectures.

You are given **Q1_Tester.m** to help you test whether your **calibrate.m** is correct.

Also note that the given script **Q1.m** uses the matrix **P** to project the 3D points into the image and plots the corresponding points. If your **calibrate.m** function correctly computes **P**, then there should be good alignment between the **xy** and projected **XYZ** points.

Question 2 (25 points)

When shooting the two images in Question 1, the same camera settings were used. All that changed was the orientation and position of the camera – and these changed only slightly. In particular, the camera internals (matrix **K)** should not have changed between shots. However, when you look at the values in the matrices **K1** and **K2** that you obtain from your calibration in Question 1, you will find that the corresponding parameters for **K1** and **K2** are different. For example, the principal points estimated are surprisingly far from the center pixel of the image. This suggests that the estimates of the camera parameters were not so reliable.

Your task in this question is to explore the reasons why. Here are some ideas to get started with, followed by specific tasks for you to carry out.

One reason for the unreliable estimates is that we have chosen only 12 object points. Although the system is over-constrained (i.e. there are more data values in **xy** and **XYZ** than **P** array entries), the position points **xy** and **XYZ** are only an approximation and one really should use more data points than that to effectively reduce these approximation errors.

The fundamental issue here is that different combinations of parameters in the **K, R,** and **C** matrices can produce quite similar **P** matrices, and so the estimates for these parameters turns out to be sensitive to small variations in the **xy** and **XYZ** values**.** For example, in Question 1, you observed that the principal point in the **K** matrix was estimated to be far from the center pixel of the image. One way this can happen is if there are also errors in the estimated **R** or **C** values that happen to cancel out these **K** errors, such that that your calibration can still yield a very good fit to the given data points even though the **K, R,** or **C** values are incorrect. In fact what is happening here is that we you have *overfit* the data.

Your task in this question is to examine two ways in which such fitting errors can occur.

A. Using one of the given images, say **c1.jpg**, show examples (computed with Matlab) of how changing two of the **K, R,** and **C** matrices can produce similar image *shifts*. Specifically, introduce an additional matrix multiply that essentially modifies the **K, R,** or **C** transformation and construct a new projection matrix **P**. Then compute the image positions of the projected **XYZ** points. For example, show how changing the **K** transformation can lead to a similar shift as changing the **R,** and show how changing the **K** transformation can lead to a similar shift as changing the **C** transformation.

B. Show examples of how similar image *expansions (scaling)* can arise from changing the **K** and **C** transformations**.** (**R** does not produce expansion, so we don't consider it here.)

For both (A) and (B), provide figures in the PDF showing the original plotted points given in the starter code, and the transformed points which are due to image shifts or image expansions, respectively. Discuss your results. *It is fine if your results are qualitative only for this question.*

Submit a script file **Q2.m** which you use to perform these computations.

Question 3 (25 points)

You might conclude from the Questions 1 and 2 that it is impossible to perform calibration accurately, since values of **K, R,** and **C** are inherently ambiguous. Such a conclusion is too strong, however; with enough data points and with accurate estimation of point locations, one *can* estimate the **K, R,** and **C** parameters. Indeed the **Q1_Tester** code showed that it is possible to perform calibration accurately if there is no noise. [ASIDE (optional): To fully convince yourself, try adding a small amount of noise to **xy** and **XYZ.** You should see that the estimated parameter values change only slightly. Morever, you should find that the estimates are better if you add more points (e.g. another object).]

Explain why it is possible in principle to resolve the qualitative ambiguities between the **K, R,** and **C** parameters which you observed in Question 2. Consider separately the cases of (A) namely **K** versus **R,** and **K** versus **C**, and (B) **K** versus **C**.

Support your argument in each case with a simple quantitative example that *concisely* illustrates your arguments. Write (and submit) a script **Q3.m** which generates figures that you will use in your examples. This script can be based on the modified **Q1_Tester.m** code, which sets up a simple scene and projection scenario.

Question 4 (25 points)

This question will give you some basic experience with homographies. You will study what happens when a camera is rotated about the projection center, that is, rotation with no translation. Such a pure rotation motion is used, for example, to stitch images to make panoramas. To add a bit more complexity, we also ask you to combine the rotation with a change in camera resolution and focal length.

Write (and submit) a Matlab script **Q4.m** that does the following.

First, read in the **c1.jpg** image.

Second, 'hard code' the matrix **K**. The reason for doing so is, as we saw in Question 1, we cannot trust the estimated values of **K**. To define **K**, take the principal point to be at the center of the image sensor; let the focal length be 50mm; let the sensor size in the x and y direction be 23.6 mm and 15.8 mm, respectively. Also, assume that the pixels cover the sensor area, and that there is no skew.

Third, write the code for computing and displaying an image that would be obtained by a *second camera* that satisfies all of the following:

- The second camera has the same location (projection point position) as the first camera, but it is rotated relative to the first camera's y axis by **theta** degrees, where **theta** is a variable that you will set in your **Q4.m** code. (Note that Matlab's cos() and sin() function assume the angle is in radians, so you will need to convert.)

- The second camera has half the focal length (25mm) of the first camera. This increases the field of view width in the images in the second camera. Since we don't know what is outside the field of view of **c1.jpg**, we will assume that all pixels outside the field of view of the first camera are black.

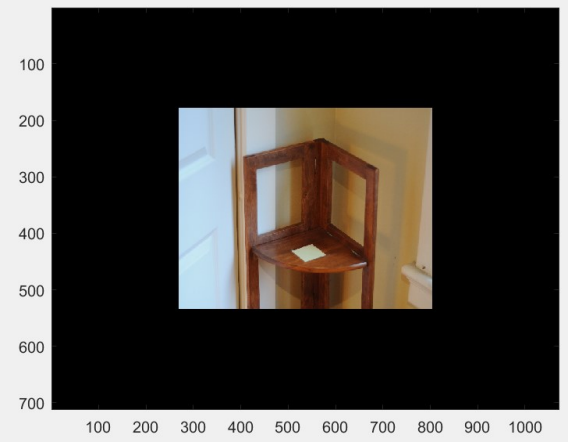
- The second camera has the same physical sensor size, but it has half the resolution. That is, the second camera has half as many pixels in both the x and y directions as the first camera.

Hints for Question 4:

- Construct the homography that maps from pixel locations in the second camera image to pixel locations in the first camera image. To choose the RGB values in the second camera image, take each pixel location in the second camera image and find the corresponding location in the first image, and take the RGB from there. The mapped (x,y) locations will not be integers, so you can just roundoff rather than interpolate. The mapped (x,y) location might not be within the field of view of the first camera image, and so in that case the RGB of the pixel should be black.

- The homography map is not simply a shift in the pixels, since the viewing direction has changed, i.e. rotation. Instead the image frame should have a trapezoidal shape.

- When **theta = 0**, you should get the image below. For a non-zero **theta,** you will get a different image. In the PDF that you submit, include images for the cases **theta** = 10 and 20 degrees.



Good luck and have fun!