GZ720J: Homework 1 - Planar Homographies

Instructor - Gary Overett
TA - Hu Yafei

5 September 2015 - Due Midnight 19 September 2015

Instructions/Hints

Today's homework deals with a very interesting property of camera geometries. The relationship between the planes within the camera scene and the associated mathematics "planar homographies". We spoke in class about the ambiguity between the scene geometry in 3 dimensions and the 2d camera sensor. This ambiguity is partially resolved when observing planes.

1. Start early: The theory questions require you to put time and effort to consider the homography problem in depth. The implementation questions also may take much longer than your expectation. Matlab is a powerful and flexible language. It means you can do things swiftly in a few lines of Matlab code; it also means the program may go wrong without reporting any errors. Even for advanced Matlab users, it is often the case that a whole day can be spent debugging.

2. **Submission:** Your submission should be a **single zip** file `NetID.zip` that at least contains:

   • A Root Level Folder: `<NetID>` - so that we can unpack your work and uniquely identify it!

   • `<NetID>.pdf`: a pdf containing your answers to all written items, including images or diagrams. Your pdf should contain answers for Q1.*.

   • The Matlab files for the implementation portions of the homework.

   • WE provide a LaTeX PDF template and Matlab starting code for you to use in the homework zip file. Please use these templates **exactly**. Do NOT rename files, change function headers etc. This assists us in grading your submissions and understanding your work efficiently. Marks may be deducted for template deviations that make marking difficult.

   • Remove commented code that is part of your prior work. We will deduct marks for leftover code that we find making your work harder to read.

3. The problems in this assignment need not be solved sequentially. Dont get stalled on a question when there are other questions to answer. You should read and understand section 3 first, but the proofs are not required to solve later problems. **Q1.X** are theory questions which lend insight to the homography estimation implementation in **Q2.X** and **Q3.X**.

4. Do not underestimate the time required to implement some of these concepts in Matlab. This implementation is necessary for solving the rest of the problems in the assignment.

5. If you have questions, please post them on the BlackBoard first.

Introduction

In this homework, we will be exploring the usefulness of homographies. Robots often deal with planes, whether in the form of walls, ground, or some other flat surface. When two cameras observe a plane, there exists a relationship between the images captured. This relationship is defined by a $\mathbf{3 \times 3}$ transformation matrix, called a planar homography.

A planar homography allows us to compute how a planar scene would look from second camera location, given only the first image. In fact, we can compute how images of the plane will look from any camera at any location without knowing any internal camera parameters and without actually taking the pictures, all with the planar homography.

Suppose we have two cameras $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ looking at a common plane $\boldsymbol{\Pi}$ in 3D space. Any 3D point $\boldsymbol{P}$ on $\boldsymbol{\Pi}$ generates a projected 2D point located at $\boldsymbol{p_1} \equiv (\boldsymbol{x_1}, \boldsymbol{y_1}, \mathbf{1})^{\boldsymbol{T}}$ on the first camera $\boldsymbol{C_1}$ and $\boldsymbol{p_2} \equiv (\boldsymbol{x_2}, \boldsymbol{y_{2,1}})^{\boldsymbol{T}}$ on the second camera $\boldsymbol{C_2}$. Since $\boldsymbol{P}$ is confined to the plane $\boldsymbol{\Pi}$, we expect that there is a relationship between $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$. In particular, there exists a common $\mathbf{3 \times 3}$ matrix $\boldsymbol{H}$, so that for any $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$, the following conditions holds:

$$\boldsymbol{p_1} \equiv \boldsymbol{H p_2} \tag{1}$$

We call this relationship planar homography. Recall that both $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ are in homogeneous coordinates and the equality $\equiv$ means $\boldsymbol{p_1}$ is proportional to $\boldsymbol{H p_2}$.

1 Planar Homographies: Theory (40pts)

The Q1 questions should not take more than a couple of lines each. In particular, the "proofs" do not require any lengthy calculations. If you are lost in many lines of complicated algebra you are doing something much too complicated (or wrong).

Question 1.1 (10pts)

Prove that there exists an $\boldsymbol{H}$ that satisfies Equation 1 given two $\mathbf{3 \times 4}$ camera projection matrices $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ corresponding to cameras $\boldsymbol{C_1}$, $\boldsymbol{C_2}$ and a plane $\boldsymbol{\Pi}$. You do not need to produce an actual algebraic expression for $\boldsymbol{H}$. All we are asking for is a proof of the existence of $\boldsymbol{H}$. Note: A degenerate case happens when the plane $\boldsymbol{\Pi}$ contains both cameras centers, in which case there are infinite choices of $\boldsymbol{H}$ satisfying Equation 1. You can ignore this special case in your answer.

Question 1.2 (10pts)

Prove that there exists an $\boldsymbol{H}$ that satisfies Equation 1 given two cameras separated by a pure rotation. That is, for $\boldsymbol{C1}$, $\boldsymbol{p_1} = \boldsymbol{K_1}[\boldsymbol{I}\,\mathbf{0}]\boldsymbol{P}$ and for $\boldsymbol{C_2}, \boldsymbol{p_2} = \boldsymbol{K_2}[\boldsymbol{R}\,\mathbf{0}]\boldsymbol{P}$. Note that $\boldsymbol{K}$ is not the same for the two cameras.

Question 1.3 (5pts)

Suppose that a camera is rotating about its center $\boldsymbol{C}$, keeping the intrinsic parameters $\boldsymbol{K}$ constant. Let $\boldsymbol{H}$ be the homography that maps the view from one camera orientation to the view at a second orientation. Let $\boldsymbol{\theta}$ be the angle of rotation between the two. Show that $\boldsymbol{H^2}$ is the homography corresponding to a rotation of $\boldsymbol{2\theta}$.

Question 1.4 (5pts)

Why is the planar homography not sufficient to map any arbitrary scene image to another viewpoint? State your answer concisely in one or two sentences.

Question 1.5 (10pts)

We have a set of points $\boldsymbol{p_1^i}$ in an image taken by camera $\boldsymbol{C_1}$ and points $\boldsymbol{p_2^i}$ in an image taken by $\boldsymbol{C_2}$. Suppose we know there exists an unknown homography $\boldsymbol{H}$ such that

$$\boldsymbol{p_1^i \equiv H p_2^i} \tag{2}$$

Assume the points are homogeneous coordinates in the form $\boldsymbol{p_j^i = (x_j^i, y_j^i, 1)^T}$. For a single point pair, write a matrix equation of the form

$$\boldsymbol{Ah = 0} \tag{3}$$

Where $\boldsymbol{h}$ is a vector of the elements of $\boldsymbol{H}$ and $\boldsymbol{A}$ is a matrix composed of the point coordinates.

1. How many elements are there in $\boldsymbol{h}$?
2. How many point pairs are required to solve this system?
3. Show how to estimate the elements in $\boldsymbol{h}$ using the Rayleigh Quotient Theorem mentioned in class to find a solution to minimize this homogeneous linear least squares system.

2 Planar Homographies: Implementation (30pts)

Question 2.1 (15pts)

Implement a function `H2to1=computeH(p1, p2)`. Inputs: `p1` and `p2` should be $\boldsymbol{2 \times N}$ matrices of corresponding $\boldsymbol{(x, y)^T}$ coordinates between two images. Outputs: `H2to1` should be a $\boldsymbol{3 \times 3}$ matrix encoding the homography that best matches the linear equation derived above in Equation 2 (in the least squares sense). Hint: Remember that a homography is only determined up to scale. The Matlab function eig will be useful.

Question 2.2 Mom! I am on TV!! (15pts)

Have you been to a Pirates game at PNC park? Have you ever wondered how cool it would be to have your face show up on the big screen as shown in Figure 1a[1]? With what you have learned so far, this is no longer a fantasy. Assuming the the big screen is planar (which it most likely is), there exist a homography between the plane of the big screen and any (planar) image. Therefore, if we are able to find this homography, then we can map an image on your computer to the big screen as shown in Figure 1d. Please do the following:

a) Create an image with **at least your name on it**. You can modify the provided `pnc_tomap.jpg` as shown in Figure 1b. Please save this image as `pnc_tomap.jpg`.

[1]Photo taken by Shoou-I Yu, Padres v.s. Pirates, 8/10/2012.

b) Find the point correspondences `p1` and `p2` ($\mathbf{2 \times N}$ matrices) between the big screen of PNC park and the image with your name. You can use Matlabs cpselect to select matching points on each image. Hint: What is the minimum amount of point correspondences required to compute a homography? Which point correspondences should you choose? Please save `p1` and `p2` in `Q4.2.p1p2.mat` with this command: `save(Q4.2.p1p2.mat, p1, p2)`.

c) Implement a function:

```
[img_yourname_warped, img_PNCpark_yourname] =
                warp2PNCpark(img_PNCpark, img_yourname, p1, p2)
```

which takes Figure 1a (`img_PNCpark`) and Figure 1b (`img_yourname`) as input,

and outputs Figure 1c (`img_yourname_warped`) and Figure 1d (`img_PNCpark_yourname`). The point correspondences `p1` and `p2` are the ones you found in the previous step. Apply your `computeH` to compute a homography $\boldsymbol{H}$.

Then use the provided `warpH` function to warp the image with your name to the coordinates of the big screen of PNC park as shown in Figure 1c.

`warp_im=warpH(im, H, out_size, fill_value)`

warps image `im` using the homography transform $\boldsymbol{H}$. In this function, the pixels in `warp_im` are sampled at coordinates in the rectangle `(1, 1)` to `(out_size(2), out_size(1))`. The coordinates of the pixels in the source image are taken to be `(1, 1)` to `(size(im,2), size(im,1))` and transformed according to $\boldsymbol{H}$. The empty regions of `warp_im` are filled with `fill_value`. Setting an appropriate `fill_value` can help you quickly find empty regions. The final step will be to combine the output of `warpH` with `img_PNCpark` to get the final output.

A common problem: There are different data types to represent images for Matlab. When you read an image with imread, the data type used is usually uint8. The range of values for uint8 is 0 to 255, so negative numbers are not available for the uint8 type. If you want to have negative numbers in your matrix, then you need to use a data type of single or double. However, if you are using single or double typed images, then for imshow to display correctly, the values of the image should be between 0 and 1. On the other hand, if you want to use imshow to view an uint8 image, the values of the image should range from 0 to 255. You can convert between single and uint8 data types with the functions `im_single = single(im)` and `im_uint8 = uint8(im)`.

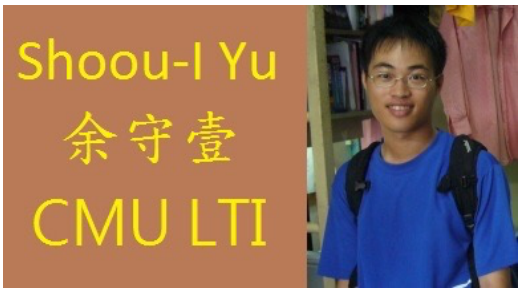
d) Run the provided `q42checker.m`. The `q42checker.m` makes sure that the format of the functions and files follow the specifications of this homework. Please make sure that your code can run through the checker without any errors. The checker will generate an image with a structure very similar to Figure 1d. There will be additional crosses on the images in the top row showing the point correspondences `p1` and `p2`. **Please include this image in your PDF submission. Make sure that the resolution is HIGH ENOUGH (at least 1280 x 908) for the TAs to clearly see the image.**

3 Panoramas (30pts)

We can also use homographies to create a panorama image from multiple views of the same scene. This is possible for example when there is no camera translation between the views (e.g., only rotation about the camera center), as we saw in **Question 1.2**. You will need to use the provided `warpH` function again which you used in **Question 2.2**(c). Use the provided `q5checker.m`, and make sure that your code runs without any error.

(a) Big screen showing players name

(b) Image with your name



(c) Image with your name warped

(d) Big screen with Shoou's name

Question 3.1 (15pts)

For this problem, use the provided images `taj1r.jpg` and `taj2r.jpg` along with the data in `tajPts.mat`. Use Matlabs `imread` function to load an image and the load function to load a mat file. `tajPts` is a $\mathbf{4 \times 1048}$ matrix containing the coordinates of features points in `taj1` in the first two rows and matching feature points for `taj2` in the bottom two rows.

You can visualize a sample of these correspondences using `plotMatches(img1,img2,pts)` Within a few weeks, we will see how to automatically generate them.

Figure 2: Matched features from `tajPts.mat`.

a) Apply your `computeH` using these correspondences to compute `H2to1`, which is the homography from `taj2` onto `taj1`. To validate your result, try to warp all the feature points in `taj2` to the image coordinate of the `taj1`. And check the error between corresponding points by calculating the 2D distance between them. Save `H2to1` as `q5_1.mat` using Matlabs save function. Save all the transformed feature points of `taj1` by `H2to1` as `q5_1_warpedFeatures.mat` using Matlabs save function. Note that this should be a $\mathbf{2 \times 1048}$ matrix. Also, write the Root Mean Squared Error between corresponding points in your PDF submission.

Figure 3: Original images and warped

(a) `taj1.jpg` (b) `taj2.jpg` (c) `taj2` warped into `taj1`

b) Apply the computed homography `H2to1` to `taj2` using `warpH`. Note that since the warped image will be translated to the right, you will need a larger target image. Use `outSize = [size(taj1,1),3000];` Save this figure as `q5_1.jpg`. Produce a simple panorama by laying `taj1` on top of the warped `taj2`. Save this figure as `q5_1_pan.jpg`. Save your code in a Matlab function `[H2to1,warpedImg,panoImg] = q5_1(img1, img2, pts)` which accepts the images and feature points and returns homography, warped image, and panorama image.

Question 3.2 (15pts)

Suppose we want to display all the visible pixels in the panorama. Write a few lines of Matlab code to find a matrix $\boldsymbol{M}$, and `out_size` in terms of `H2to1` and `size(im1)` and `size(im2)` such that,

```
warp_im1=warpH(im1, M, out_size);
warp_im2=warpH(im2, M*H2to1, out_size);
```

produces images in a common reference frame where all points in `im1` and `im2` are visible and `out_size(2)=1280` (`out_size(1)` should be determined by `out_size(2)`). The matrix $\boldsymbol{M}$ should perform scaling and translation only, and the resulting image should fit snugly around the corners of the panorama. Do not select $\boldsymbol{M}$ manually. It should be computed by your code, considering the size of the panorama image. Save this image as `q5_2_pan.jpg`. Save the code used to generate and display the panorama as

`[H2to1, panoImg] = q5_2(img1, img2, pts)`

where `img1` is `taj1` and `img2` is `taj2`.

Figure 4: Final mosaic view. Border lines are only for reference and need not be present in your solution.