

Assignment #2 : Warping and Mosaicing

(Due date: 2/22/07)

1 Introduction

This is a programming assignment, which should be done in Matlab as many of the necessary numerical routines are readily available (e.g., eig for the computation of the eigenvalues and eigenvectors of a matrix). All data necessary for this assignment is available on the course web page.¹

2 Image Warping

2.1 Introduction

In this assignment, we consider a vision application in which a scene is being observed by multiple cameras at various locations and orientations. Figure 1 shows examples of images taken by six different cameras.

Given two cameras, a natural thing to do would be to compute how the scene currently seen from camera 1 would appear from camera 2's point of view. In particular, this would allow one to paste together multiple images which have overlapping regions, even if those images were obtained from different locations. This is also called Mosaicing.

An image mosaic is created by first choosing one camera as the reference frame and its associated image as the reference image. The task then consists of mapping all other images onto the reference frame so that all images can be displayed together with the reference image. In the most general case, there would be no constraints on the scene geometry, making the problem quite hard to solve. If, however, the scene can be approximated by a plane in 3D, a solution can be formulated much more easily even without the knowledge of camera calibration parameters. Figure 2 depicts a typical mosaicing result.

To solve this section of the homework, you will first derive the transformation that maps one image onto another in the planar scene case. Then you will write a program to find this warping and apply it to a pair of test images, which are provided on the course web page. Finally, you will estimate and remove a specific radiometric distortion from the images known as vignetting.

2.2 Projective Transformations of the Plane

To begin, we consider the projection of planes in images. Imagine two cameras C_1 and C_2 looking at a plane π in the world. Consider a point $\mathbf{P}$ on the plane π and its projections $\mathbf{p} = (u_1, v_1, 1)^T$ in image 1 and $\mathbf{q} = (u_2, v_2, 1)^T$ in image 2.

Fact 1 *There exists a unique (up to scale) 3×3 matrix $\mathbf{H}$ such that, for any point $\mathbf{P}$:*

$$\mathbf{q} \equiv \mathbf{H}\mathbf{p}$$

(Here $\equiv$ denotes equality in homogeneous coordinates, meaning that the left and right hand side are proportional.) Note that $\mathbf{H}$ only depends on the plane and the projection matrices of the two cameras.

¹This assignment is based on an assignment by Prof. Jean Ponce at the University of Illinois at Urbana-Champaign and Prof. Martial Hebert at CMU.

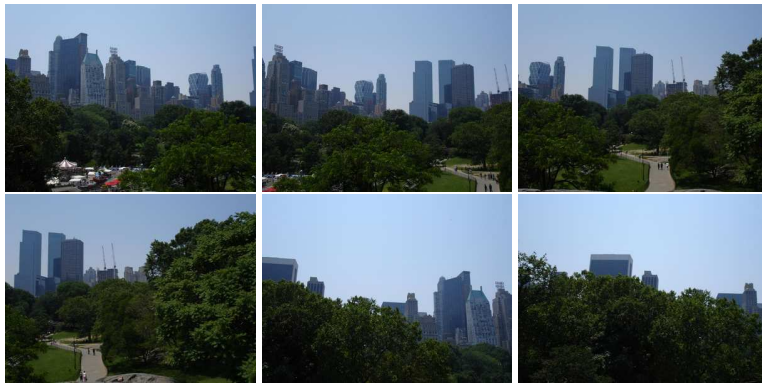


Figure 1: Input images.



Figure 2: Resulting mosaic image.

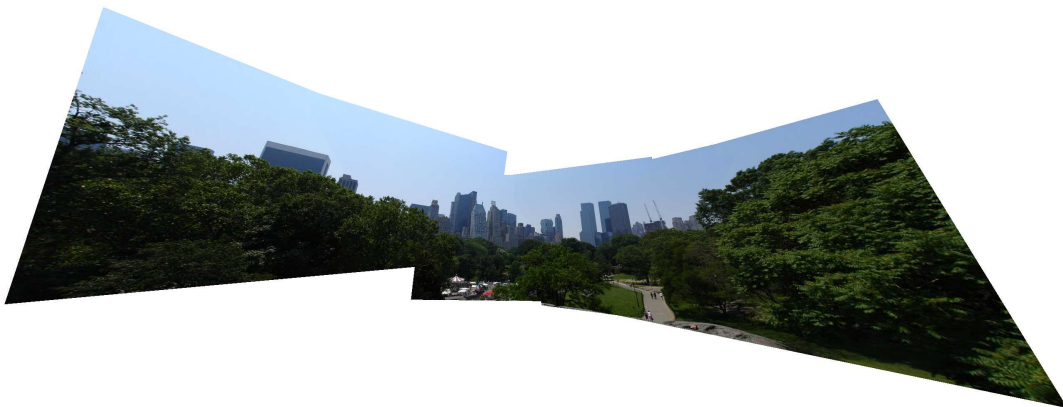


Figure 3: Resulting mosaic image with vignetting correction.

The interesting thing about this result is that by using $\mathbf{H}$ we can compute the image of $\mathbf{P}$ that would be seen in camera C_2 from the image of the point in camera C_1 without knowing its three-dimensional location. Such an $\mathbf{H}$ is a *projective* transformation of the plane, also referred to as a *homography*.

Perhaps even more interesting, a homography can also be used to represent the transformation between images of a genuine 3-D scene for multiple images taken by camera that rotates in 3-D about a fixed point. Do you see why?

2.3 Estimating Transformations From the Image Points

Given a set of points $\{(u_1^i, v_1^i)\}$, $i = 1 \dots N$ in image 1 and the corresponding set of points $\{(u_2^i, v_2^i)\}$, $i = 1 \dots N$ in image 2, show that $\mathbf{H}$ can be recovered from two sets of image points using *homogeneous* linear least squares (since $\mathbf{H}$ is only defined up to scale).

Write a program which computes the matrix $\mathbf{H}$ using the method you just derived. Input arrays should be $2 \times N$ matrices, listing a single image point's coordinates per column. Verify your results by manually selecting two sets of corresponding image points on two of the provided images, and applying the transformation $\mathbf{H}$ to them. (Checkout the function *cpselect* in Matlab's Image Processing Toolbox for help selecting corresponding points.) Note that, throughout this estimation procedure, camera projection matrices did not come into play at all! Hence, no calibration was necessary.

Important hints: Recall that $\mathbf{H}$ is a projective transformation matrix and hence, defined only *up to a scale*. A good way to enforce this is by constraining the squared Frobenius norm (sum of the squared entries) of the matrix $\mathbf{H}$ to be equal to 1. Also remember that $\mathbf{q}$ and $\mathbf{H}\mathbf{p}$ are only *proportional* to each other, or equivalently we have $\mathbf{q} \times \mathbf{H}\mathbf{p} = \mathbf{0}$.

Here are a few implementation tips:

- Use the images *im1.jpg* and *im2.jpg* on the web page as test images. You can read these images using the function *imread* in Matlab. (There are six images there, *im1.jpg* to *im6.jpg*, but you only need test your method on the first two.)
- Image coordinates of the points are the corresponding row and column indices of the image array.
- Beware of numerical ill-conditions: Your estimation procedure may perform better if image coordinates range from -1 to 1 as opposed to from 1 to 800. Consider scaling your measurements to avoid numerical issues².
- For the estimation to work well, a sufficient number of points should be provided. You should select the corresponding points in the two test images manually (or with the help of *cpselect*) and give all your results with your own point set.

3 Image Warping and Mosaicing

Write a program which takes as input an image *lin*, a reference image *Iref*, and a 3×3 homography $\mathbf{H}$ and returns two images as outputs. The first image is *Iwarp*, which is the input image *lin* warped according to $\mathbf{H}$ to be in the frame of the reference image *Iref*. The second output image is *Imerge*, a single mosaic image with a larger field of view containing both the input images. In order to avoid aliasing and sub-sampling effects, consider producing the output by solving for each pixel of the *destination* image rather than mapping each pixel in the input image to a point in the destination image (which will leave holes). Also note that the input and output images will be of different dimensions. Also, consider how to bring together the information from both images into the mosaiced image.

Finally, apply your method to all six images, using the first image as the reference frame³, to produce a single mosaic, akin to figure 2.

4 Vignetting Correction

As can be seen from figures 2 and 4, even with perfect alignment there are still noticeable seams in the resulting mosaic image. The most significant cause for seaming is vignetting falloff, which can be divided into four distinct

²Hartley normalization is a well known procedure for doing this. Given a set of homogeneous 2D points $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, the Hartley normalized point set is given by $\hat{\mathbf{x}}_i = A\mathbf{x}_i$ where A is a 3×3 linear transformation such that $\{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N\}$ has zero mean and unit variance. In the context of homography estimation, we can write $\mathbf{q} = H\mathbf{p} \Rightarrow A_q\mathbf{q} = A_qHA_p^{-1}A_p\mathbf{p} \Rightarrow \hat{\mathbf{q}} = \hat{H}\hat{\mathbf{p}}$. Thus, we can solve for $\hat{H}$ in terms of $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$ and then solve for H as $H = A_q^{-1}\hat{H}A_p$.

³You may encounter problems if you try to use a reference frame far from the center of the resulting mosaic; in this case points will project to infinity and/or to the back of the reference frame which may not be accounted for in your Matlab code. Note that other projective mappings such as cylindrical or equirectangular allow full 360° mosaics.



Figure 4: (Left) Zoom-in of seam artifact caused by vignetting. (Right) Result after vignetting correction.

contributing effects : natural vignetting, pixel vignetting, optical vignetting, and mechanical vignetting. Natural vignetting refers to the approximate $\cos^4 \theta$ radial falloff that results from geometric optics; this result was derived in lecture. Pixel vignetting is caused in digital cameras by the finite depth of each photon well, which causes decreased photon sensitivity with increased incident light angle. Optical vignetting results from light paths blocked by the lens diaphragm; larger apertures typically result in more significant optical vignetting effects. Mechanical vignetting refers to falloff due to light paths blocked by other camera components such as filters or hoods attached to the front of the lens body. Since each term contributes to the vignetting in a non-trivial (and camera specific) way, it is difficult to derive an expression for it. However, because each of the vignetting terms is (nearly) radially symmetric about the optical axis, we can approximate vignetting as a function $V(r)$ of the distance r from the optical center. The image intensity for the j th pixel in the i th image is then related to the irradiance as,

$$I_{ij} = E_{ij}V(r_{ij}). \quad (1)$$

A commonly used parametric form for $V(r)$ is

$$V(r) = ar^6 + br^4 + cr^2 + 1 \quad (2)$$

where a , b , and c are parameters that can be estimated from the input images. Using this form for $V(r)$, show how to solve for a , b , and c in terms of pixel intensities $\{I_{ij}, I'_{ij}\}$ from corresponding points in the input images and their respective distances from the optical center $\{r_{ij}, r'_{ij}\}$. Using this result, write a Matlab function that returns a , b , and c given input I , r , I' , r' where each of the inputs is an $N \times 1$ vector (i.e., N pixel correspondences). Run your code on at least 2000 randomly selected correspondence points across all input images to estimate the vignetting parameters a , b , and c (assume the optical center is the center point of the image). Finally, correct the vignetting effect in each image using equation 1 and construct a new panoramic image using the corrected input images.

What to Turn In

Submit a hardcopy at the due date with the following.

1. Derivation of the algorithm for estimating **H**.
2. Your code for : estimating **H**, image warping, image merging, and vignetting correction.
3. An image showing the set of corresponding points you selected plotted on the original images.
4. An image showing the result of warping applied to image 2.
5. An image showing the result of combining images 1 and 2.
6. An image showing the mosaic of all six images.
7. The estimated vignetting parameters a , b , and c .
8. An image showing the mosaic of all six images after vignetting correction.

In addition, please email to nalldrin@cs.ucsd.edu a copy of your code. If it is a single file, just send the .m file. If it is contained in multiple files, send a zip or tar file. In the subject line, please put the string: CSE252HW2.