16720: Computer Vision Homework 1

Yuzhang Wu
Andrew ID: yuzhang2
November 8, 2015

Q1.1

The general world frame is $(x, y, z, 1)^T$, and world frame in this problem can be chosen randomly. We choose world frame to set the coordinates of point P in Plane ∏ to be $(x, y, 0, 1)^T$. Let T_1 to be the 1, 2, 4 column of projection matrix M_1, which is a 3×3 matrix. Let T_2 to be the 1, 2, 4 column of projection matrix M_2, which is a 3×3 matrix, too. T_1 and T_2 are assumed to be invertible. Then, the perspective projection equations become

$$\boldsymbol{p1} \equiv M_1\mathbf{P} \equiv \boldsymbol{T_1}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{1})^{\boldsymbol{T}}$$
$$\boldsymbol{p2} \equiv M_2\mathbf{P} \equiv \boldsymbol{T_2}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{1})^{\boldsymbol{T}}$$

So we can get

$$\boldsymbol{T_1}^{-1}\boldsymbol{p1} \equiv (\boldsymbol{x}, \boldsymbol{y}, \mathbf{1})^{\boldsymbol{T}}$$
$$\boldsymbol{T_2}^{-1}\boldsymbol{p2} \equiv (\boldsymbol{x}, \boldsymbol{y}, \mathbf{1})^{\boldsymbol{T}}$$

Therefore

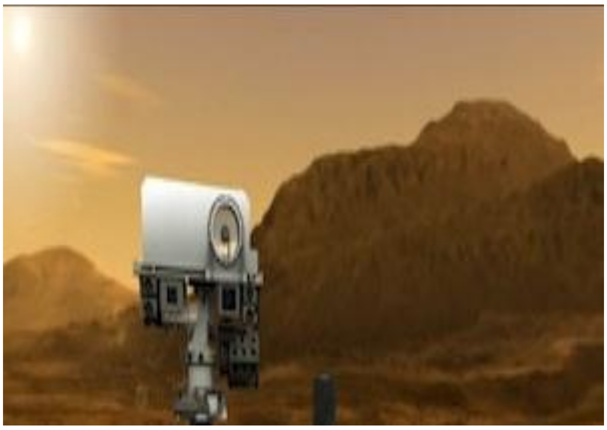
$$\boldsymbol{p1} \equiv \boldsymbol{T_1}\boldsymbol{T_2}^{-1}$$

So there exists a matrix $\mathbf{H} \equiv \lambda\boldsymbol{T_1}\boldsymbol{T_2}^{-1}, \lambda \neq \mathbf{0}$ such that $\boldsymbol{p1} \equiv \boldsymbol{Hp2}$.

Q1.2

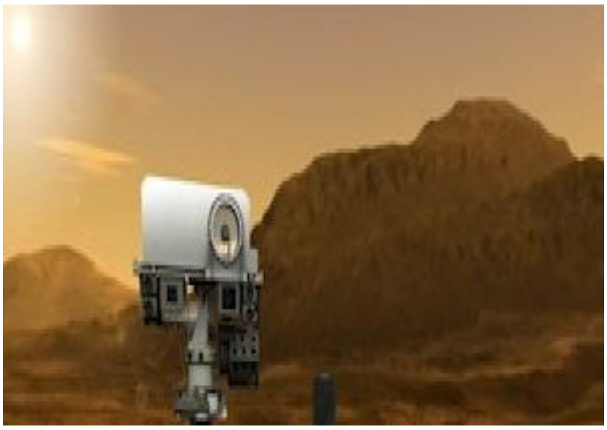
Planar homography requires that the objects in the scene are in the same plane. But in arbitrary scene, it cannot be guaranteed.

Q1.3



QX1

I use the function of homework0 to implement QX1, the function warpImage(inputImage, H) uses the knowledge of affine warp with two simple functions called meshgrid and sub2ind.



Q2.1

We denote $\boldsymbol{h_1}^{\boldsymbol{T}}$, $\boldsymbol{h_2}^{\boldsymbol{T}}$, $\boldsymbol{h_3}^{\boldsymbol{T}}$, to be the rows of H. From $\boldsymbol{p_1^i} \equiv \boldsymbol{Hp_2^i}$, we can get

$$\boldsymbol{x_1^i = \frac{h_1^T p_2^i}{h_3^T p_2^i}}$$

$$\boldsymbol{y_1^i = \frac{h_2^T p_2^i}{h_3^T p_2^i}}$$

Rearrange

$$\boldsymbol{x_1^i h_3^T p_2^i - h_1^T p_2^i = 0}$$
$$\boldsymbol{y_1^i h_3^T p_2^i - h_1^T p_2^i = 0}$$

We want to solve for H. Even though these inhomogeneous equations involve the coordinates nonlinearly, the coefficients of H appear linearly. Rearranging equation and we get

$$\boldsymbol{a_x^T h = 0}$$
$$\boldsymbol{a_y^T h = 0}$$

Where

$$\mathbf{h} = (\boldsymbol{h11, h12, h13, h21, h22, h23, h31, h32, h33})^{\boldsymbol{T}}$$
$$\boldsymbol{a_x = (-x_1, -y_1, -1, 0, 0, 0, x_2x_1, x_2y_1, x_2)^T}$$
$$\boldsymbol{a_y = (0, 0, 0, -x_1, -y_1, -1, y_2x_1, y_2y_1, y_2)^T}$$

Since **h = (h11, h12, h13, h21, h22, h23, h31, h32, h33)** and **Ah = 0**. However, h only has 8 degrees of freedom. *So the dimensions of A should be m×8, and m ≥8. We need at least 4 point correspondences to solve this problem* and these points cannot be in the same line. To estimate h, the sum squared error can be written as

$$\mathbf{f(h)} = \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{Ah} - \mathbf{0})^{\boldsymbol{T}}(\boldsymbol{Ah} - \mathbf{0}) = \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{h}^{\boldsymbol{T}}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{Ah}$$

Taking the derivative of f with respect to h and setting the result to zero, we get

$$\frac{\mathbf{df}}{\mathbf{dh}} = \mathbf{0} \Rightarrow$$

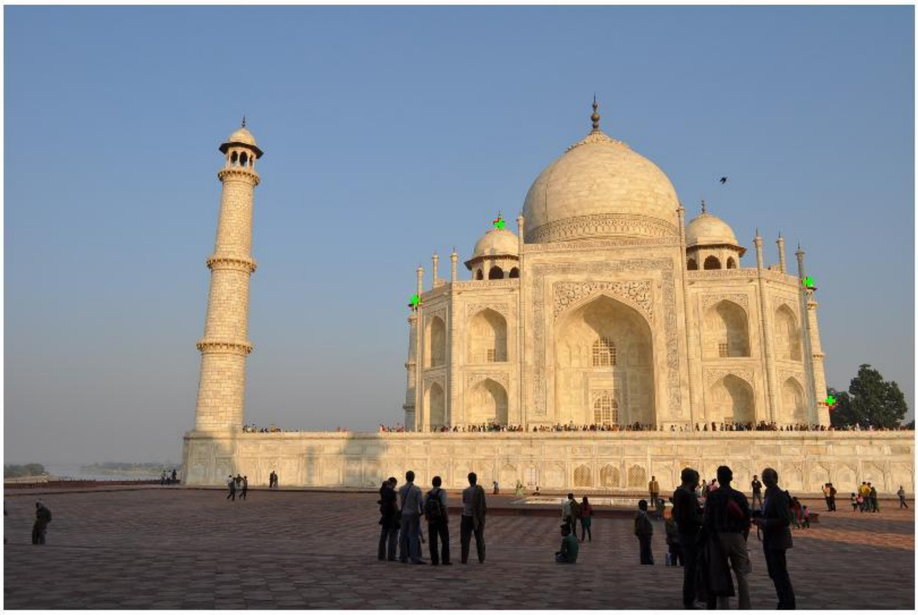
$$\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{Ah} = \lambda\boldsymbol{h}$$

Looking at the eigen-decompostion of $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$, we see that h should equal the eigenvector of $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$ that has an eigenvalue of zero. The solution to this problem is the unit eigenvector corresponding to the smallest eigenvalue.

Q2.2

The procedure of the function compute has been stated in Q2.1. There is one thing should be mentioned that ***eig*** is replaced by ***svd*** because the matrix to be processed is semi-positive definite and ***svd*** is better than ***eig*** when dealing with semi-positive definitive matrix.

Q2.3



The Average Error is 1.666775 which means the average length between two pixels and it can be denoted as

$$\mathrm{e} = \frac{1}{n}\sum_{i=1}^{n} ||p_1 - Hp_2||_2$$

MATLAB expression for the average error (in pixels)

```
p2to1(3,:) = [];
parallax = p1 - p2to1;
```

```
error = 0;
for i = 1:4
    x = parallax(1,i);
    y = parallax(2,i);
    error = error+x^2+y^2;
end
error = sqrt(error)/4;
```

Q2.4

```
% get height and width of two images
height1=size(img1,1);
width1=size(img1,2);
height2=size(img2,1);
width2=size(img2,2);
% right-up corner of warped image
rightUpCorner=(H2to1*[width2;1;1]);
rightUpCorner=rightUpCorner./rightUpCorner(3);
% right-down corner of warped image
rightDownCorner=(H2to1*[width2;height2;1]);
rightDownCorner=rightDownCorner./rightDownCorner(3);
% left-up corner of warped image
leftUpCorner=(H2to1*[1;1;1]);
leftUpCorner=leftUpCorner./leftUpCorner(3);
% left-down corner of warped image
leftDownCorner=(H2to1*[1;height2;1]);
leftDownCorner=leftDownCorner./leftDownCorner(3);
% width and height of the final image
maxwidth=max([rightUpCorner(1),rightDownCorner(1),width1]);
minwidth=min([leftUpCorner(1),leftDownCorner(1),0]);
width=maxwidth-minwidth;
maxheight=max([leftDownCorner(2),rightDownCorner(2),height1]);
minheight=min([rightUpCorner(2),leftUpCorner(2),0]);
height=maxheight-minheight;

% scale
scale=1280/width;
out_size = round(scale*[height,width]);
% Define a function to create an affine scaling matrix
Scalef = @(s)([ s 0 0;
              0 s 0;
              0 0 1]);
Transf = @(tx,ty)([1 0 tx;
                 0 1 ty;
                 0 0 1]);
```

```
% Finally get the M
M = Transf(-scale*minwidth,-scale*minheight)*Scalef(scale);
```

Q2.5

I first make the two images warped. Specifically for the overlapping areas, I use the method listed which let the weight of two images' values different. And finally to eliminate the edge noise, I use median filter. The result is shown as this.

QX2

I find that the bad quality of panorama always occur in the overlap areas of the final image. To make the two images fuse better, I try to implement the task with some morphology knowledge. Specifically, for the overlap part, I implement an expansion operation to make it look corresponding.

Q3.1

The mugs in the two images cannot be coincident. There appears two mugs in the new picture and the objects on the screen which are behind the mugs cannot match as well due to the refraction.

Q3.2

Method:

Since homography is subject to the objects must be in the same plane, the mugs will cause refractions and so that it will stand out. My solution to detecting transparent objects is trying to find which pixels in two pictures will not obey the planar homography and the pixels cannot correspond to each other should be the transparent part.

I first get the panorama of these pictures as Q3.1 requires and then calculate the differences between panorama and warped-picture. Through experiments, get the appropriate threshold. Use Gaussian filter to make the image fluent and eliminate the noise point. Finally, warp the image to the original size.

Threshold:

There are two parts needed threshold in my function. One is the threshold to ensure the differences which means the transparent objects. I just make many experiments to find the best value. Another part is about the threshold in Gaussian filter, I turn to some papers to learn how this parameter mean and try to find proper threshold.

QX3

The main point to get the homographic transformations is to get the H, which means the relationship between two images. To get H, we need at least 4 corresponding points to calculate it. Two things can be the challenge when creating panorama, first we should select good points and make sure the points we choose are corresponding. Another thing is that we need to blend the overlap part of the panorama picture well to make them look natural.