HW4 REPORT

(Yunke Tian)

(109929662)

## Problem 1

1.a

A=[kf 0 u0;0 lf v0;0 0 1]=[1600 0 400;0 1600 300;0 0 1]

1.b

The quaternion representation of the rotation is:

q=[sin(pi/6)\*u cos(pi/6)]=[ 0.2942 0.3922 -0.0981 0.8660]

and by the Q2M.m function, the quaternion is converted to 3\*3 matrix as:

0.6731 0.4006 0.6217

0.0609 0.8077 -0.5864

-0.7371 0.4326 0.5192

combined with the translation, the 4\*4 matrix is:

1.0e+03 \*

0.0007 0.0004 0.0006 0

0.0001 0.0008 -0.0006 0

-0.0007 0.0004 0.0005 1.0000

0 0 0 0.0010

1.c

Using extrinsic and intrinsic transformations and got the vertices on image:

Columns 1 through 4

402.7120 400.7238 401.4316 400.5578

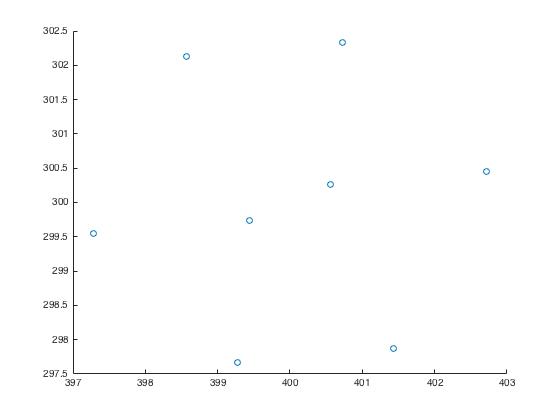
300.4514 302.3300 297.8655 300.2561

Columns 5 through 8

399.4403 398.5703 399.2774 397.2868

299.7431 302.1318 297.6738 299.5484

When scattered on the 2D plane, like the following:



1.d

On the image corresponding to infinite x,y,z are as follows:

x -1061.1 1881.6 2315.6

y 167.7 3287.2 -1507.1

# Problem 2

2.a

For prospective projection camera model, equation is like this:



where, u/w and v/w are known image coordinates, and X, Y, Z are world coordinates of the cross center on the table; A is the calibration matrix: [kf 0 u0;0 lf v0;0 0 1] with 4 freedoms.

For projection camera model, the projection ray is perpendicular to the table plane. Thus there are more constraints to cancel the freedoms.

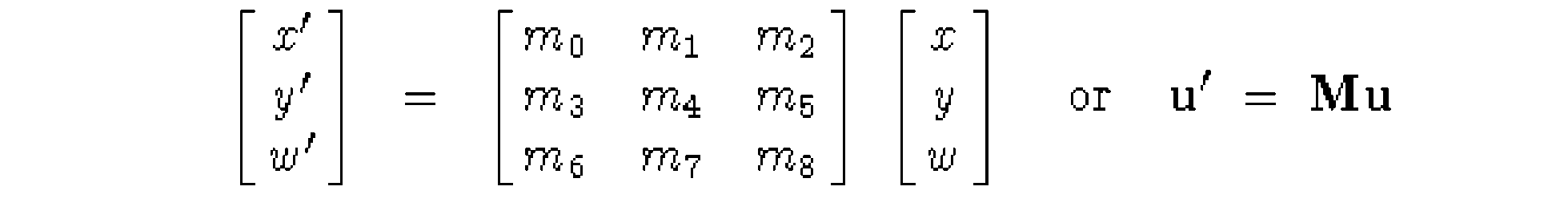
2.b

4 freedoms for the intrinsic parameters, and Xi, Yi, Zi are not known, with 5 freedoms. So there are in total 9 freedoms. But we only have 6 equations, not enough to solve the problem.

# Problem 3

3.b.a

The equation is that:



(here set w=1,m8=1)

make it into a set of equations as:

x’i=m0\*xi+m1\*yi+m2/(m6\*xi+m7\*yi+1)

y’i=m3\*xi+m4\*yi+m5/(m6\*xi+m7\*yi+1)

which can be rewrite as:

x’i= m0\*xi+m1\*yi+m2-m6\*xi\*x’i-m7\*yi\*x’i

y’i= m3\*xi+m4\*yi+m5-m6\*xi\*y’i-m7\*yi\*y’i

and this form can be easily converted into matrix form to solve the solution.

3.b.b

For this problem, I chose 8 feature points widely distributed:

img1:

80 631 522 313 681 445 461 642

412 383 525 495 945 1066 1211 1101

corresponding img2:

46 626 513 295 666 433 450 624

117 83 238 207 663 780 917 810

And got the transform matrix like this:

1.1213 0.0631 -66.8673

-0.0022 1.1836 -363.6975

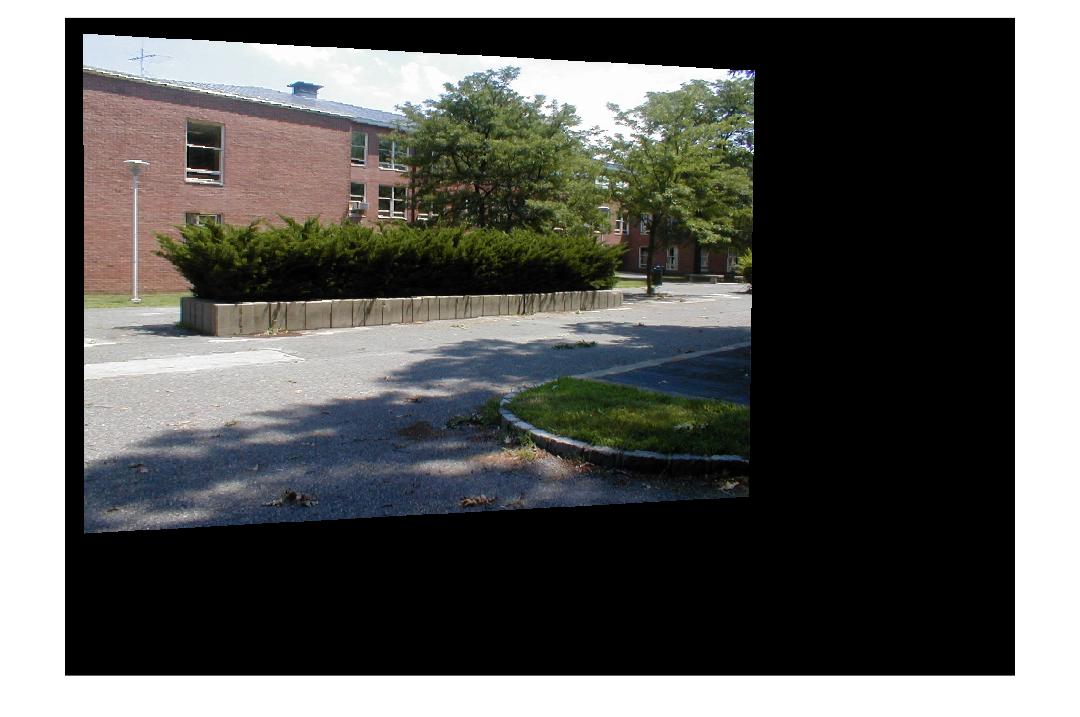
0.0000 0.0001 1.0000

3.b.c

WarpImage function is implemented in the ‘codes’ folder. Also, the backward mapping algorithm was used with bilinear interpolation:

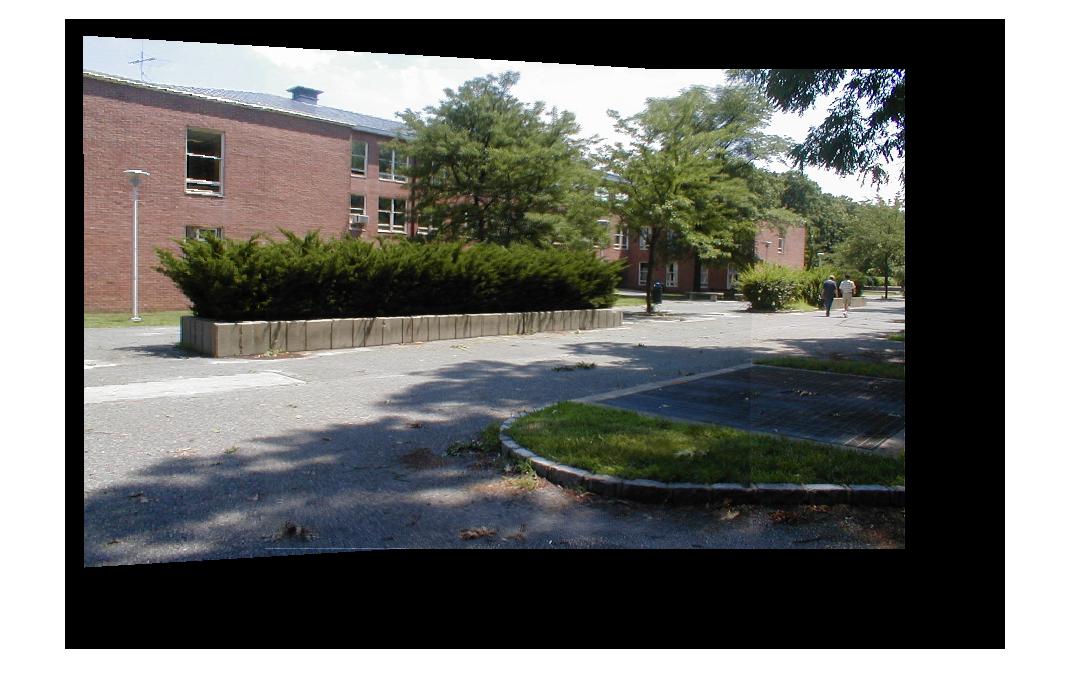
(part of the code): image1warp(x0+i,y0+j,1:3)=left\*up\*double(img1(floor(points(1)),floor(points(2)),1:3))+left\*down\*double(img1(ceil(points(1)),floor(points(2)),1:3))+right\*up\*double(img1(floor(points(1)),ceil(points(2)),1:3))+right\*down\*double(img1(ceil(points(1)),ceil(points(2)),1:3));

Below are the origin pic and warped pic:



3.c

Mosaicing: under the instruction, I successfully yielded the mosaic image of the two:



# Problem 4

For this problem, I first used SIFT function to extract all the feature points of the two pictures. Then I used vl\_ubcmatch to find all the matched feature point pairs. Then is the RANSAC method:

1. Randomly find a set of 4 pairs of feature points;
2. Calculate the transform matrix H;
3. Use H to warp im1, then compare the warped im1 to im2, using least square method;
4. Repeat this 50 times and find the set of pairs with minimum least square, and use the corresponding H as the transform matrix.
5. Use the previously implemented ComputeWarpMapping and WarpImage function to do the mosaic.

The whole process is complex with many details, and I wrote only the sketch code for it, which cannot actually run.