CSE 252A Homework 1

Yingyan Hua and Yanli Wang

Problem 1. Perspective Projection

Solution:

The ray expression $Q = \begin{bmatrix} 2 \\ -3 \\ 0 \end{bmatrix} + t \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$ can also be written as $Q = \begin{bmatrix} 2 \\ 2t - 3 \\ t \end{bmatrix}$. Since all points on line Q are represented by choosing different values of t, we can convert the Euclidean coordinate of each point to homogenous coordinate using the expression of Q.

For any point(x,y,z), the Euclidean to homogenous coordinate transformation is:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow k \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

so for any point on line Q its homogenous coordinates are

$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \rightarrow k \begin{bmatrix} 2 \\ 2t - 3 \\ t \\ 1 \end{bmatrix}.$$

The projective transformation for each point on line Q is

$$\begin{bmatrix} x_t{}' \\ y_t{}' \\ z_t{}' \end{bmatrix} = k \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f' & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2t - 3 \\ t \\ 1 \end{bmatrix} = k \begin{bmatrix} 2 \\ 2t - 3 \\ t/f' \end{bmatrix}, \text{ where } -\infty \leq t \leq -1.$$

This 2D homogenous coordinates can be converted to 2D Euclidean coordinates by dividing the x and y coordinates by z coordinate:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 2f'/t \\ (-3 + 2t)f'/t \end{bmatrix}.$$

Therefore, when $t = -\infty$ the coordinates of the projection of the ray onto the image plane are

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 2f' \end{bmatrix}.$$

And when $t = -1$, the coordinates are

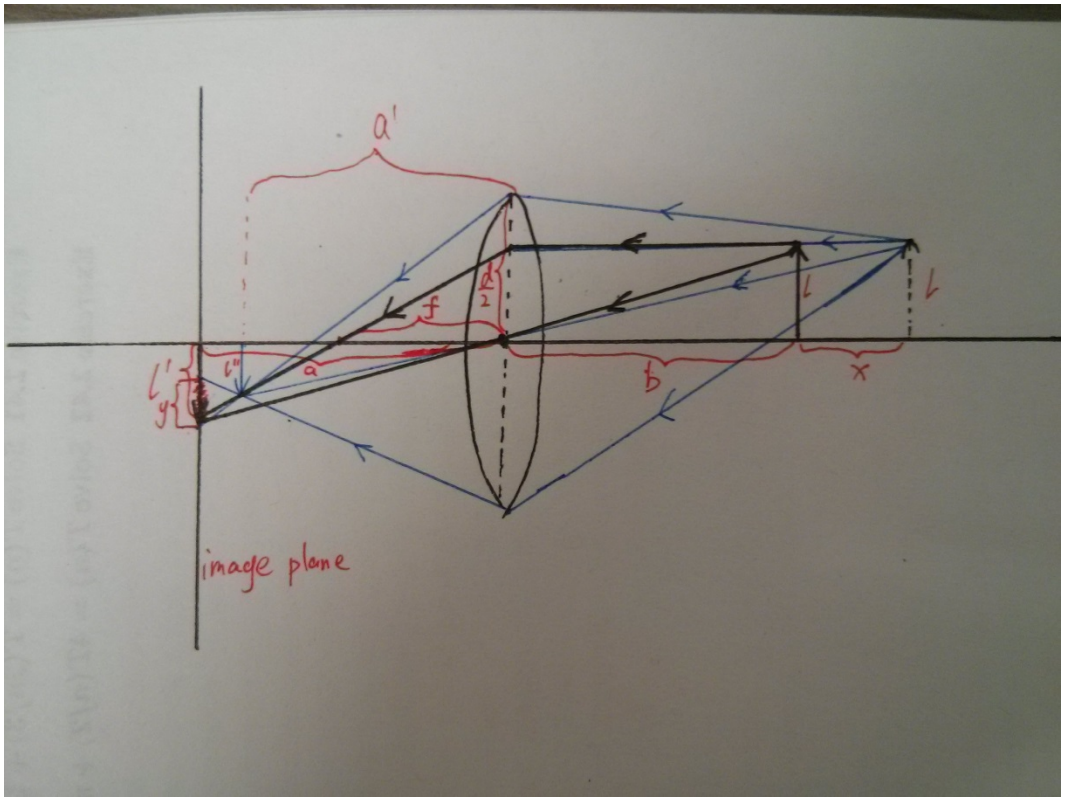
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -2f' \\ 5f' \end{bmatrix}.$$

Problem 2. Thin Lens Equation

Solution:

Assuming the object size is l and image size is l', then from the given information we know that $l = 2l'$.

Here we set the image distance as *a*, object distance as *b* and focal length as *f*.



From thin lens equation and triangle similarity theorem we know that

$\frac{1}{a} + \frac{1}{b} = \frac{1}{f}$ and $\frac{b}{a} = 2$.

Combining this with the given information that $a + b$ = 90cm, we can solve the equations for a, b, and f and get a = 30cm, b = 60cm, and f = 20cm.

Therefore, the answer to question 1 is that the lens must be placed 60cm away from the object.

The answer to question 2 is that the focal length of the lens is 20cm.

Question 2.3

When arrow moves x cm to the right while the lens and image plane remain fixed, the area of blur circle formed from the tip of the arrow is shown as in the image above.

Let's set the new image size as l'', the new image distance as a' and the radius of the blur circle as y.

From the thin lens equation $\frac{1}{f} = \frac{1}{a'} + \frac{1}{b+x}$, together with $f = 20\text{cm}$ and $b = 60\text{cm}$,

we get $a' = \frac{20\times(60+x)}{40+x}$, and $a - a' = \frac{10x}{40+x}$. Applying triangle similarity theorem, we know $\frac{2y}{d} = \frac{a-a'}{a'}$, and then the blur radiu is y $= \frac{a-a'}{2a'} \times d = \frac{xd}{2(120+x)}$.

Problem 3. Affine Projection

Solution:

First, because A, B and C are on the same 3D line and the line equation can be expressed as $P = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + t \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix}$, where x_d, y_d, and z_d are the coordinates of a point on the line, $<a_0, b_0, c_0>$ is a vector that's parallel to the line, and t can be any number, then the coordinates of A, B, and C can be expressed as $\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + t_A \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix}$, $\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + t_B \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix}$, and $\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + t_C \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix}$, where t_A, t_B, and t_C are distinct numbers.

The ratio of the distances between the three points is thus the ratio of two vector subtractions:

$$\frac{\overrightarrow{AB}}{\overrightarrow{BC}} = \frac{\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} - \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix}}{\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} - \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix}} = \frac{t_A - t_B}{t_A - t_C} .$$

Let H be the affine transformation matrix,

then $H = \begin{bmatrix} \frac{f}{z_0} & 0 & -\frac{fx_0}{{z_0}^2} & \frac{fx_0}{{z_0}^2} \\ 0 & \frac{f}{z_0} & -\frac{fy_0}{{z_0}^2} & \frac{fy_0}{{z_0}^2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

If we let a, b, and c be the camera image points of A, B, C after affine projection respectively, then the coordinates of a, b, and c are:

$$\begin{bmatrix} x'_a \\ y'_a \\ z'_a \end{bmatrix} = H * \left(\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + t_A \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} \right), \qquad \begin{bmatrix} x'_b \\ y'_b \\ z'_b \end{bmatrix} = H * \left(\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + t_B \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} \right),$$

and $\begin{bmatrix} x'_c \\ y'_c \\ z'_c \end{bmatrix} = \mathrm{H} * \left(\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + t_C \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} \right)$.

Since matrix multiplications are distributive over matrix additions, the above expressions become

$\begin{bmatrix} x'_a \\ y'_a \\ z'_a \end{bmatrix} = \mathrm{H} * \left(\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} \right) + \mathrm{H} * (\mathrm{t_A} \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix})$, $\begin{bmatrix} x'_b \\ y'_b \\ z'_b \end{bmatrix} = \mathrm{H} * \left(\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} \right) + \mathrm{H} * \left(\mathrm{t_B} \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} \right)$, and $\begin{bmatrix} x'_c \\ y'_c \\ z'_c \end{bmatrix} = \mathrm{H} * \left(\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} \right) + \mathrm{H} * (\mathrm{t_C} \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix})$,

So the ratio of two image-point distances is $\frac{\overrightarrow{ab}}{\overrightarrow{ac}} = \frac{\begin{bmatrix} x'_a \\ y'_a \\ z'_a \end{bmatrix} - \begin{bmatrix} x'_b \\ y'_b \\ z'_b \end{bmatrix}}{\begin{bmatrix} x'_b \\ y'_b \\ z'_b \end{bmatrix} - \begin{bmatrix} x'_c \\ y'_c \\ z'_c \end{bmatrix}} = \frac{t_A - t_B}{t_A - t_C}$, which is the same as the ratio of distances between the original points A, B, and C.

Problem 4. Image Formation and Rigid Body Transformations

Solution:

1. [No rigid body transformation]
   1) The extrinsic transformation matrix.

   $$P_{e1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

   2) Intrinsic camera matrix under the perspective camera assumption.

   $$P_{i11} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

   3) intrinsic camera matrix under the affine camera assumption.

   $$P_{i21} = \begin{bmatrix} 0.5000 & 0 & 0 & 0 \\ 0 & 0.5000 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2. [Translation]
   1) The extrinsic transformation matrix.

$$P_{e2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2) Intrinsic camera matrix under the perspective camera assumption.

$$P_{i12} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

3) intrinsic camera matrix under the affine camera assumption.

$$P_{i22} = \begin{bmatrix} 0.3333 & 0 & 0 & 0 \\ 0 & 0.3333 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3. [Translation and rotation]

1) The extrinsic transformation matrix.

$$P_{e3} = \begin{bmatrix} 0.4330 & -0.2500 & 0.8660 & 0 \\ 0.5000 & 0.8660 & 0 & 0 \\ -0.7500 & 0.4330 & 0.5000 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2) Intrinsic camera matrix under the perspective camera assumption.

$$P_{i13} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

3) intrinsic camera matrix under the affine camera assumption.

$$P_{i23} = \begin{bmatrix} 0.5000 & 0 & -0.4330 & 0.8660 \\ 0 & 0.5000 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

4. [Translation and rotation, long distance]

1) The extrinsic transformation matrix.

$$P_{e4} = \begin{bmatrix} 0.4330 & -0.2500 & 0.8660 & 0 \\ 0.5000 & 0.8660 & 0 & 0 \\ -0.7500 & 0.4330 & 0.5000 & 13 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

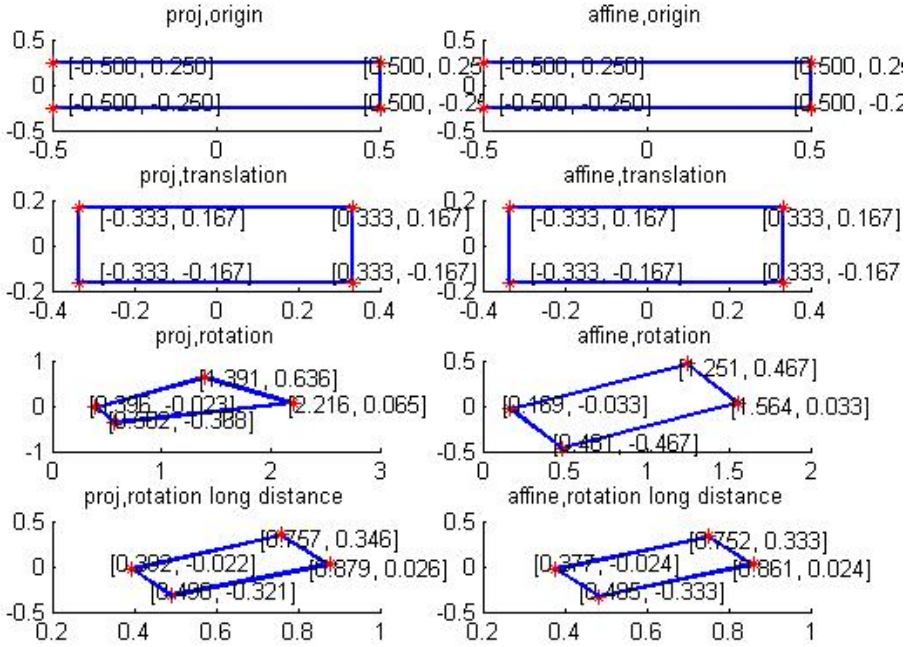
2) Intrinsic camera matrix under the perspective camera assumption.

$$P_{i14} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.2000 & 0 \end{bmatrix}.$$

3) intrinsic camera matrix under the affine camera assumption.

$$P_{i24} = \begin{bmatrix} 0.3571 & 0 & -0.0442 & 0.6186 \\ 0 & 0.3571 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

4) Calculate the image of the four vertices and plot it.



Others:

1. The actual points around which you did the Taylor series expansion for the affine camera models.
   We choose the center of the four points after rotation and translation as the point around which we did the Taylor series expansion. The points are $\mathrm{PO1} = (0,0,2.0000)^T$, $\mathrm{PO2} = (0,0,3.0000)^T$, $\mathrm{PO3} = (1.7321,0,2.0000)^T$, and $\mathrm{PO4} = (1.7321,0,14.0000)^T$.

2. How did you arrive at these points?
   This choice was made based on the reasoning that the center of the points after rotation and translation is the closest point to each of the four points. This will produce the best approximation result.

3. How do the projective and affine camera models differ? Why is this difference smaller for the last image compared to the second last?
   The projective camera model involves no approximation while the affine camera model uses Taylor series expansion to estimate image coordinates through a linear transformation, and as a result the latter model only maps parallel lines to parallel lines. The affine transformation uses the same focal length f for all points around a certain point on the object, an assumption that is not true in reality: points at different positions do have different f values. When f is large, however, we may neglect the error introduced by the approximation since the relative magnitude of error is smaller. Therefore, the images produced by the two models are much more similar in the last case than in the second last case.

The MATLAB codes for the plotimage function are attached as follows.

```
P1=[-1,-0.5,2,1]';
P2=[1,-0.5,2,1]';
P3=[1,0.5,2,1]';
P4=[-1,0.5,2,1]';
P=[P1, P2, P3, P4];

Pe1=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
Pi11=[1 0 0 0; 0 1 0 0; 0 0 1 0];
Pi21=[0.5 0 0 0; 0 0.5 0 0; 0 0 0 1];
Q11=Pi11*Pe1*P;
Q11=Q11./[Q11(3,:);Q11(3,:);Q11(3,:)];
Q21=Pi21*Pe1*P;
Q21=Q21./[Q21(3,:);Q21(3,:);Q21(3,:)];

Pe2=[1 0 0 0; 0 1 0 0; 0 0 1 1; 0 0 0 1];
Pi12=[1 0 0 0; 0 1 0 0; 0 0 1 0];
PP2=Pe2*P;
x02=sum(PP2(1,:))/4;
y02=sum(PP2(2,:))/4;
z02=sum(PP2(3,:))/4;
Pi22=[1/(z02) 0 -1*x02/(z02*z02) 1*x02/z02; 0 1/(z02) -1*y02/(z02*z02)
1*y02/z02; 0 0 0 1];
Q12=Pi12*Pe2*P;
Q12=Q12./[Q12(3,:);Q12(3,:);Q12(3,:)];
Q22=Pi22*Pe2*P;
Q22=Q22./[Q22(3,:);Q22(3,:);Q22(3,:)];

Pe3=[cos(pi/3) 0 sin(pi/3) 0; 0 1 0 0; -sin(pi/3) 0 cos(pi/3) 1; 0 0 0
1]*[cos(pi/6) -sin(pi/6) 0 0; sin(pi/6) cos(pi/6) 0 0; 0 0 1 0; 0 0 0 1];
Pi13=[1 0 0 0; 0 1 0 0; 0 0 1 0];
PP3=Pe3*P;
x03=sum(PP3(1,:))/4;
y03=sum(PP3(2,:))/4;
z03=sum(PP3(3,:))/4;
Pi23=[1/(z03) 0 -1*x03/(z03*z03) 1*x03/z03; 0 1/(z03) -1*y03/(z03*z03)
1*y03/z03; 0 0 0 1];
Q13=Pi13*Pe3*P;
Q13=Q13./[Q13(3,:);Q13(3,:);Q13(3,:)];
Q23=Pi23*Pe3*P;
Q23=Q23./[Q23(3,:);Q23(3,:);Q23(3,:)];
Pe4=[cos(pi/3) 0 sin(pi/3) 0; 0 1 0 0; -sin(pi/3) 0 cos(pi/3) 13; 0 0 0
```

```
1]*[cos(pi/6) -sin(pi/6) 0 0; sin(pi/6) cos(pi/6) 0 0; 0 0 1 0; 0 0 0 1];
Pi14=[1 0 0 0; 0 1 0 0; 0 0 0.2 0];
PP4=Pe4*P;
x04=sum(PP4(1,:))/4;
y04=sum(PP4(2,:))/4;
z04=sum(PP4(3,:))/4;
Pi24=[5/(z04) 0 -5*x04/(z04*z04) 5*x04/z04; 0 5/(z04) -5*y04/(z04*z04)
1*y04/z04; 0 0 0 1];
Q14=Pi14*Pe4*P;
Q14=Q14./[Q14(3,:);Q14(3,:);Q14(3,:)];
Q24=Pi24*Pe4*P;
Q24=Q24./[Q24(3,:);Q24(3,:);Q24(3,:)];

SUBPLOT(4,2,1), plotsquare(Q11),title('proj,origin');
SUBPLOT(4,2,2), plotsquare(Q21),title('affine,origin');
SUBPLOT(4,2,3), plotsquare(Q12),title('proj,translation');
SUBPLOT(4,2,4), plotsquare(Q22),title('affine,translation');
SUBPLOT(4,2,5), plotsquare(Q13),title('proj,rotation');
SUBPLOT(4,2,6), plotsquare(Q23),title('affine,rotation');
SUBPLOT(4,2,7), plotsquare(Q14),title('proj,rotation long distance');
SUBPLOT(4,2,8), plotsquare(Q24),title('affine,rotation long
distance');
```

Problem 5. Image Warping and Merging

Solution:
The equation used to solve for the homography matrix H is:

$$\begin{pmatrix} a_{x1} \\ a_{y1} \\ a_{x2} \\ a_{y2} \\ a_{x3} \\ a_{y3} \\ a_{x4} \\ a_{y4} \end{pmatrix} \mathrm{h} = 0$$

where $a_x = (-x_1, -y_1, -1, 0, 0, 0, x_2'x_1, x_2'y_1, x_2')^T$, $x_2' = x_2$,
$a_y = (0, 0, 0, -x_1, -y_1, -1, y_2'x_1, y_2'y_1, y_2')^T$, $y_2' = y_2$
and $h = (H_{11}, H_{12}, H_{13}, H_{21}, H_{22}, H_{23}, H_{31}, H_{32}, H_{33})$. The subscripts of x and y represent the four given points (X_1, Y_1), (X_2, Y_2), (X_3, Y_3), and (X_4, Y_4) in the points.mat file.

Singular Value Decomposition (SVD) method implemented in MATLAB (function

svd) was used to solve this equation. The “right singular vector” (the last column from V) contains the coefficients of the homography matrix that best fits the points. The expression for h were rearranged into matrix H.

The images after warping and merging are shown below.

The MATLAB codes for functions computeH.m and warp.m are attached as follows.

[computeH.m]

```
function H = computeH(targetPoints, sourcePoints)
A(1,:) = [-targetPoints(1,1), -targetPoints(1,2), -1, 0, 0, 0,
sourcePoints(1,1)*targetPoints(1,1), sourcePoints(1,1)*targetPoints(1,2),
sourcePoints(1,1)];
```

```
A(2,:) = [0, 0, 0, -targetPoints(1,1), -targetPoints(1,2), -1,
sourcePoints(1,2)*targetPoints(1,1), sourcePoints(1,2)*targetPoints(1,2),
sourcePoints(1,2)];
A(3,:) = [-targetPoints(2,1), -targetPoints(2,2), -1, 0, 0, 0,
sourcePoints(2,1)*targetPoints(2,1), sourcePoints(2,1)*targetPoints(2,2),
sourcePoints(2,1)];
A(4,:) = [0, 0, 0, -targetPoints(2,1), -targetPoints(2,2), -1,
sourcePoints(2,2)*targetPoints(2,1), sourcePoints(2,2)*targetPoints(2,2),
sourcePoints(2,2)];
A(5,:) = [-targetPoints(3,1), -targetPoints(3,2), -1, 0, 0, 0,
sourcePoints(3,1)*targetPoints(3,1), sourcePoints(3,1)*targetPoints(3,2),
sourcePoints(3,1)];
A(6,:) = [0, 0, 0, -targetPoints(3,1), -targetPoints(3,2), -1,
sourcePoints(3,2)*targetPoints(3,1), sourcePoints(3,2)*targetPoints(3,2),
sourcePoints(3,2)];
A(7,:) = [-targetPoints(4,1), -targetPoints(4,2), -1, 0, 0, 0,
sourcePoints(4,1)*targetPoints(4,1), sourcePoints(4,1)*targetPoints(4,2),
sourcePoints(4,1)];
A(8,:) = [0, 0, 0, -targetPoints(4,1), -targetPoints(4,2), -1,
sourcePoints(4,2)*targetPoints(4,1), sourcePoints(4,2)*targetPoints(4,2),
sourcePoints(4,2)];
[U S V] = svd(A);

H(1,:) = V(1:3,9);
H(2,:) = V(4:6,9);
H(3,:) = V(7:9,9);
```

[warp.m]

```
function targetImage = warp(targetImage, logoImage, targetPoints, H)
logoimgr = logoImage(:,:,1);
logoimgg = logoImage(:,:,2);
logoimgb = logoImage(:,:,3);

sz = size(targetImage);
[X Y] = meshgrid(1:sz(1,2), 1:sz(1,1));

xv = [targetPoints(:,1)',targetPoints(1,1)];
yv = [targetPoints(:,2)', targetPoints(1,2)];
in = inpolygon(X,Y,xv,yv);
XY=[Y(in) X(in)]';
szXY=size(XY);
XYZ=[XY(2,:);XY(1,:);ones(1,szXY(:,2))];
XYZ=H*XYZ;
```

```
XYZ=XYZ./[XYZ(3,:);XYZ(3,:);XYZ(3,:)];
                    ZIr=interp2(double(logoimgr), XYZ(1,:)',XYZ(2,:)');
                    ZIg=interp2(double(logoimgg), XYZ(1,:)',XYZ(2,:)');
                    ZIb=interp2(double(logoimgb), XYZ(1,:)',XYZ(2,:)');

szXYZ=size(XYZ);

for i=1:szXYZ(1,2)
                    targetImage(XY(1,i),XY(2,i),1)=ZIr(i,1);
                    targetImage(XY(1,i),XY(2,i),2)=ZIg(i,1);
                    targetImage(XY(1,i),XY(2,i),3)=ZIb(i,1);
end;
```