ASSIGNMENT 2 :

AFFINE RECTIFICATION :

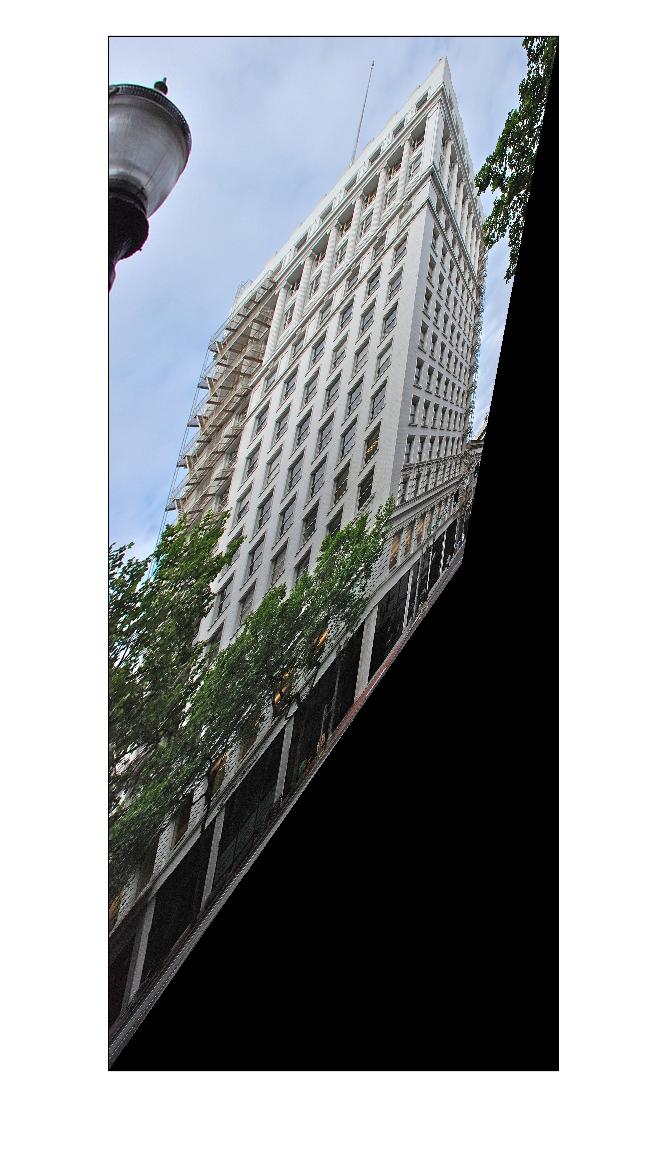
PROCEDURE :

1. To find the line at infinity, we first try to discover a pair of parallel lines on the imaged plane.
2. Because the line at infinity has been relocated from its canonical position to a finite location on the imaged plane, the first step would be to restore it to its canonical position.
3. Because the location of l8 has been retained, the affine qualities of the first plane displayed in the image above can be assessed on the third plane.
4. The projective transformation matrices that translate l1 to its canonical location [0 0 1]T are now found.
5. Once discovered, we can use this transformation to affinely rectify the entire image by applying it to every point of the image.

INPUT TEST IMAGES :



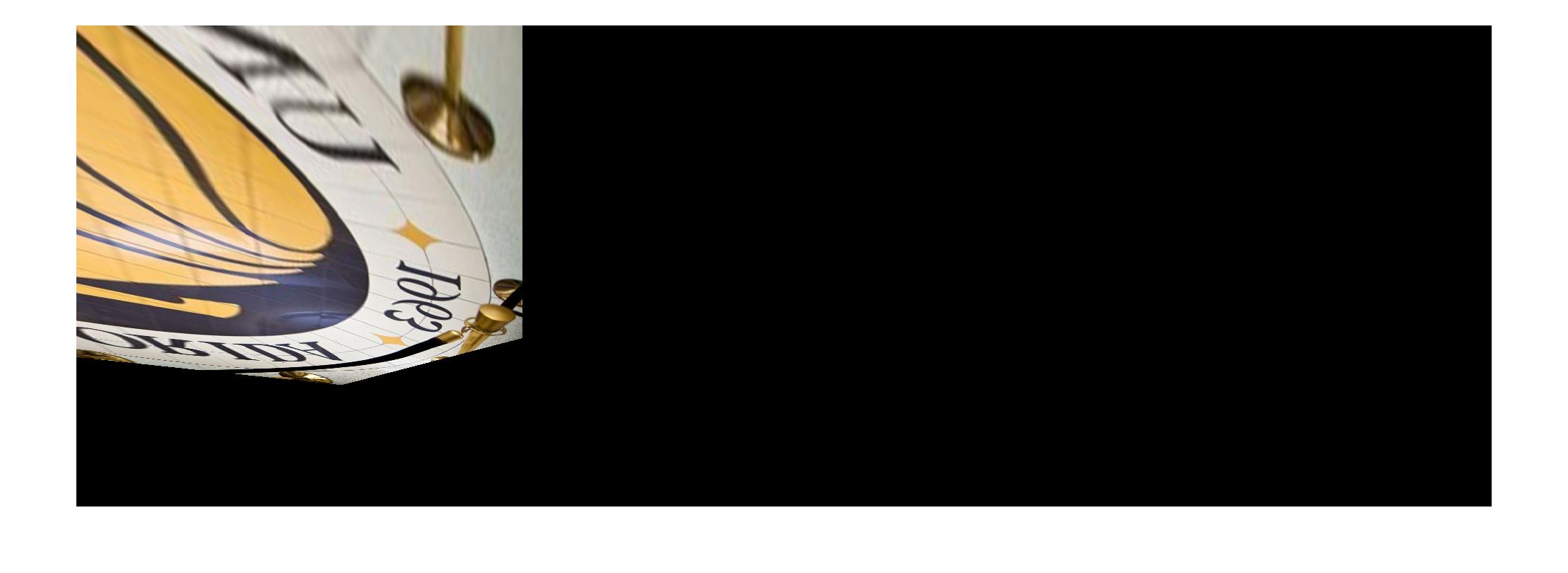
OUTPUT IMAGE



INPUT 2 AND OUTPUT 2:



INPUT 3 :

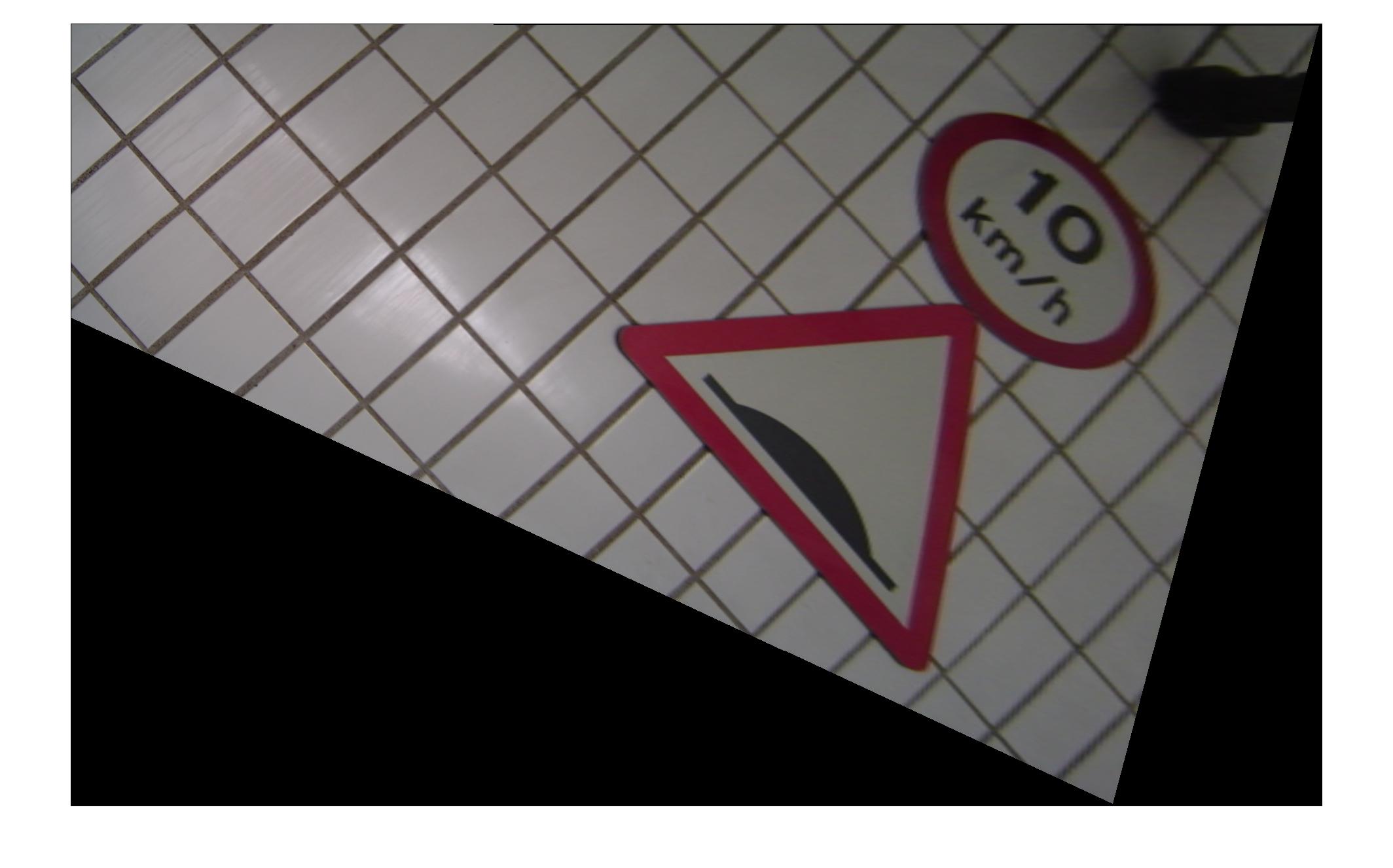
For Pegasus image, I tried clicking different points, here are some good things I got..  




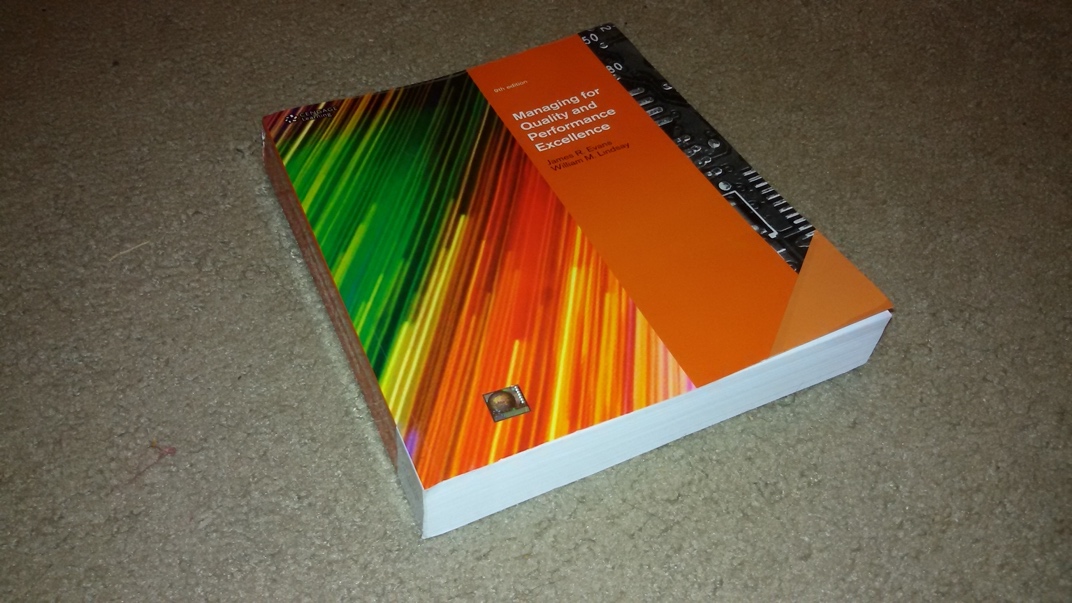
INPUT 4 : CROP\_CIRCLE



INPUT AND OUTPUT 5 :



INPUT 6 :



CONCLUSIONS AND REFLECTIONS ON AFFINE RECTIFICATION:

1. Clicking Points is the Most important thing in the affine rectification.
2. Slightest deviation in clicking points would result in bad output.
3. Affine Rectification makes the lines intersecting in the image to lines parallel as in the world.
4. Under this transformation affine properties are preserved.
5. Although affine rectification does not allow us to perform some basic viewing tasks, the technique is extremely useful in correcting geometric distortions and deformations produced by poor camera quality.
6. This approach can also be useful for precise measurements in satellite imaging and other applications.
7. One disadvantage of my technique is that it is semi-automatic, which means that it requires user input to discover the line of infinity, and the accuracy is dependent on the user's ability to locate parallel lines on the image. Computer vision methods, on the other hand, can be used to detect the pair of parallel lines automatically and then feed them into the above code.

CODE :

function H = affine\_rectification(in\_img,filename)

img = imread(in\_img);

figure(1)

imshow(img)

title("Click 4 Points on the Image")

[x,y] = ginput(4);

close Figure 1

A = [x(1) y(1) 1];

B = [x(2) y(2) 1];

C = [x(3) y(3) 1];

D = [x(4) y(4) 1];

% A B

% C D

line1 = cross(A, B);

line2 = cross(C, D);

% l1 || l2

line3 = cross(A, C);

line4 = cross(B, D);

% l3 || l4

point\_infy\_1 = cross(line1, line2);

point\_infy\_1 = point\_infy\_1/(point\_infy\_1(1,3));

point\_infy\_2 = cross(line3, line4);

point\_infy\_2 = point\_infy\_2/(point\_infy\_2(1,3));

%line at infy

line\_infy = cross(point\_infy\_1, point\_infy\_2);

line\_infy\_x = line\_infy(1,1)/line\_infy(1,3);

line\_infy\_y = line\_infy(1,2)/line\_infy(1,3);

% Computing Homography H

H = [1 0 0; 0 1 0; line\_infy\_x line\_infy\_y 1]

%H = [1 -2 3; 3 5 2; -1 3 4]

tform = projective2d(H')

Iout = imwarp(img,tform);

Ishow = imshow(Iout)

end

METRIC RECTIFICATION :

PROCEDURE :

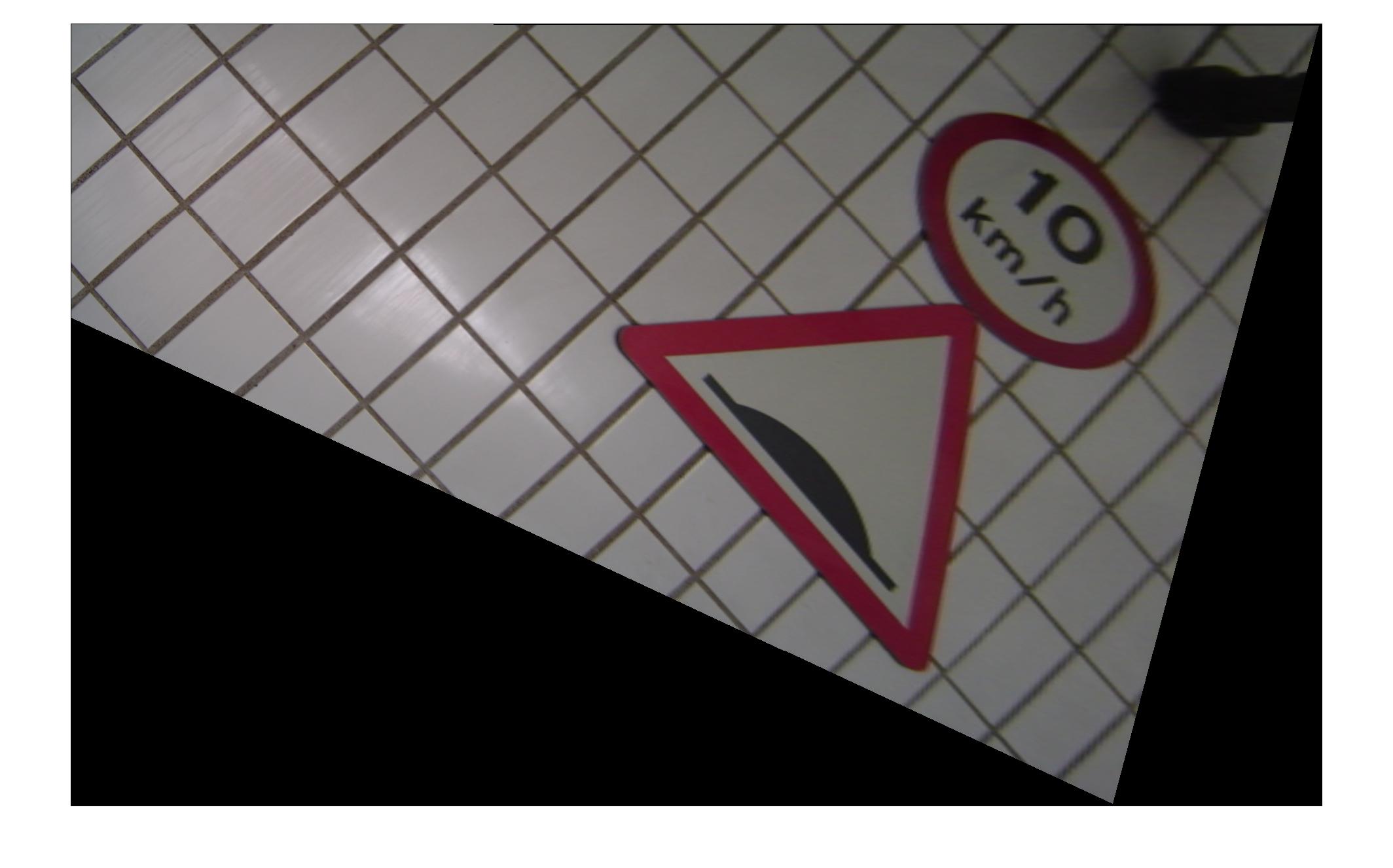
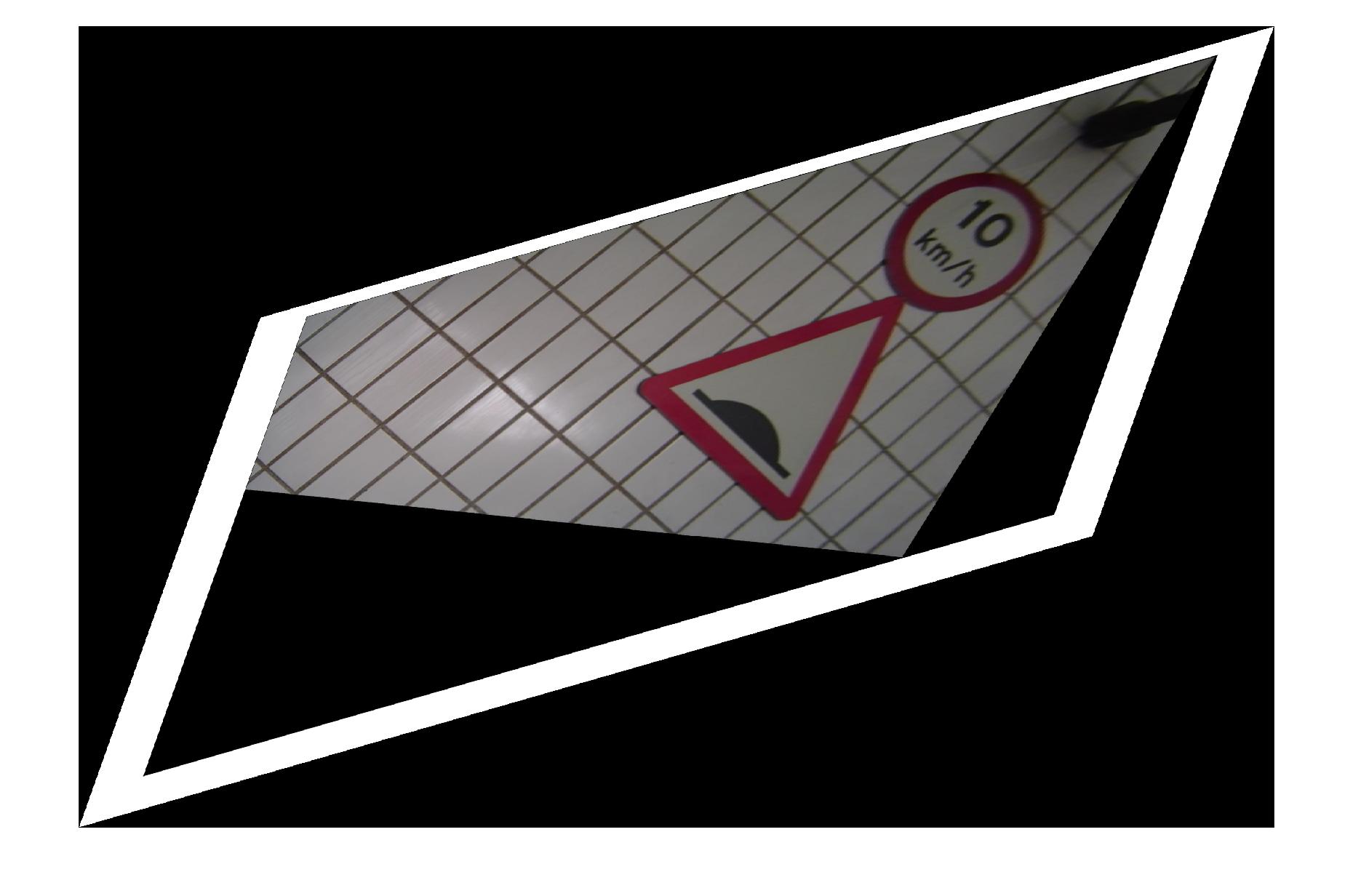
1. From the result of the following equation, we try to discover the conic. lT C\*8m = 0 lT C\*8m = 0 lT C\*8m
2. However, because C\*8 is a six-vector, we must first discover five pairs of orthogonal lines on the imaged plane that can be stacked together. [l1 l2 l3 l4 l5] L = [l1 l2 l3 l4 l5] [m1 m2 m3 m4 m5] M = [m1 m2 m3 m4 m5]
3. Then, on the elements (l1m1, (l1m2 + l2m1)/2, l2m2, (l1m3 + l3m1)/2, (l2m3 + l3m2)/2, l3m3), we discover the linear constraint. c equals 0
4. Once C\*8 is discovered, an appropriate holography is created that allows the canonical position of the circular points to be retrieved.
5. Once we've located it, we can apply this transformation to every point in the image to metrically correct it.

INPUTS AND OUTPUTS



I/O 2:



IO 3 :   


IO 4 :



OBSERVATION AND CONCLUSION :

* This algorithm has the same disadvantage. It's semi-automatic, which means it needs user input. However, the work can be expanded by employing computer vision algorithms to locate parallel lines or circles.
* The advantage of this paradigm is that it is possible to measure length and shape on the world plane from the image plane without having to explicitly recover the homography parameters.
* Not all of the circles and squares in the real world display accurately in the rectified images in the images above.
* Identifying the gaps, cues, circles, lines, and other elements that create the means to locate the line of infinity (in the case of affine r) is one of the most difficult tasks in both circumstances.

Code :

function metric\_rectification(input\_img)

in\_img = imread(input\_img);

figure(1)

imshow(in\_img)

[x,y]= getpts;

l1 = [x(1) y(1)]; l2 = [x(3) y(3)];

m1 = [x(2) y(2)]; m2 = [x(4) y(4)];

M = [l1(1) \* m1(1) (l1(1) \* m1(2) + l1(2) \* m1(1)) ; l2(1) \* m2(1) (l2(1) \* m2(2) + l2(2) \* m2(1))];

b = [-l1(2) \* m1(2); -l2(2) \* m2(2)];

sol = linsolve(M, b);

mat = [sol(1) sol(2); sol(2) 1];

[U, S, V] = svd(mat);

A = transpose(U) \* sqrt(S) \* U;

H = eye(3);

H(1:2, 1:2) = A;

H(1,1) = abs(H(1,1)); H(2,2) = abs(H(2,2));

projective\_transform = projective2d(H);

out\_img = imwarp(in\_img, projective\_transform);

figure(2);

imshow(out\_img)

end