Computer Vision
Assignment 2

Abhineet Jain
201301168

Discrete Linear Transform (DLT):

- Determine the correspondences of a set of points, X_i in Image 1.
- On Image 2, mark the points that are projections of X_i.
- $\forall i$, $x_i = \mathbf{H}X_i$, where H is the homography matrix.
- Consider 9 unknowns in a 3x3, H matrix. Given a single point correspondence, we have 2 equations, and 9 unknowns.
- Hence, we use 4 correspondences to estimate all the camera properties.
- The points should be scattered preferably.
- We use SVD to solve the 8 equations, 9 unknowns matrix equation.

Input Images:

Image 1.

Image 2.

Observations:

The green points marked were selected for DLT, and reprojected points after calculating the H matrix are shown in blue. Figure 1 shows results for automatically detected point matches, and Figure 2 shows results for manually marked points.

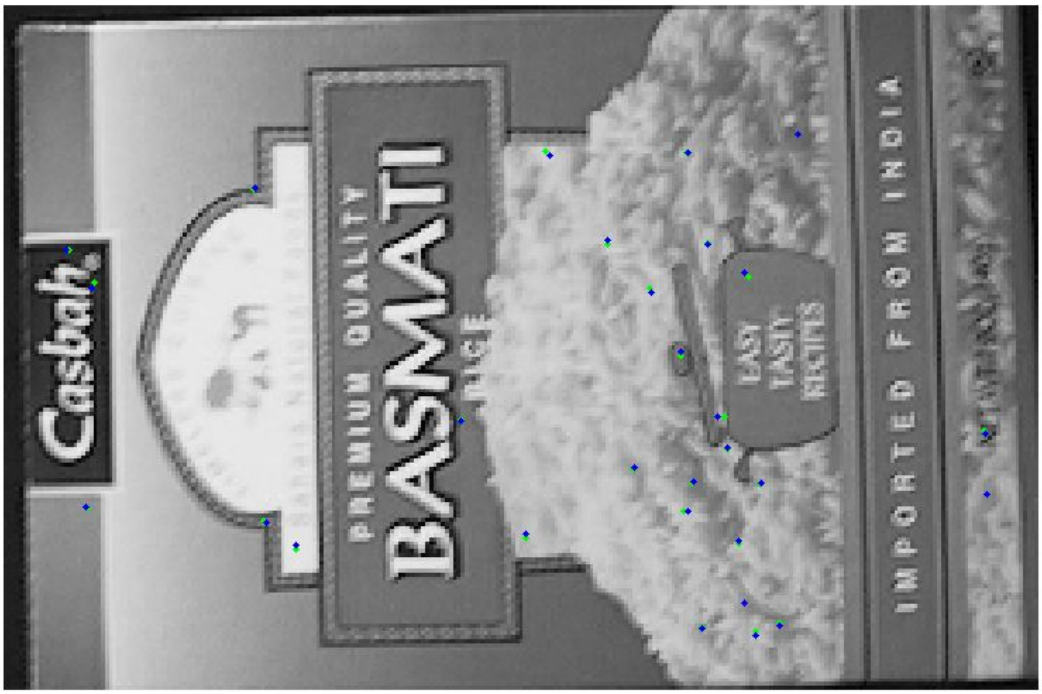


Figure 1. SIFT matched points

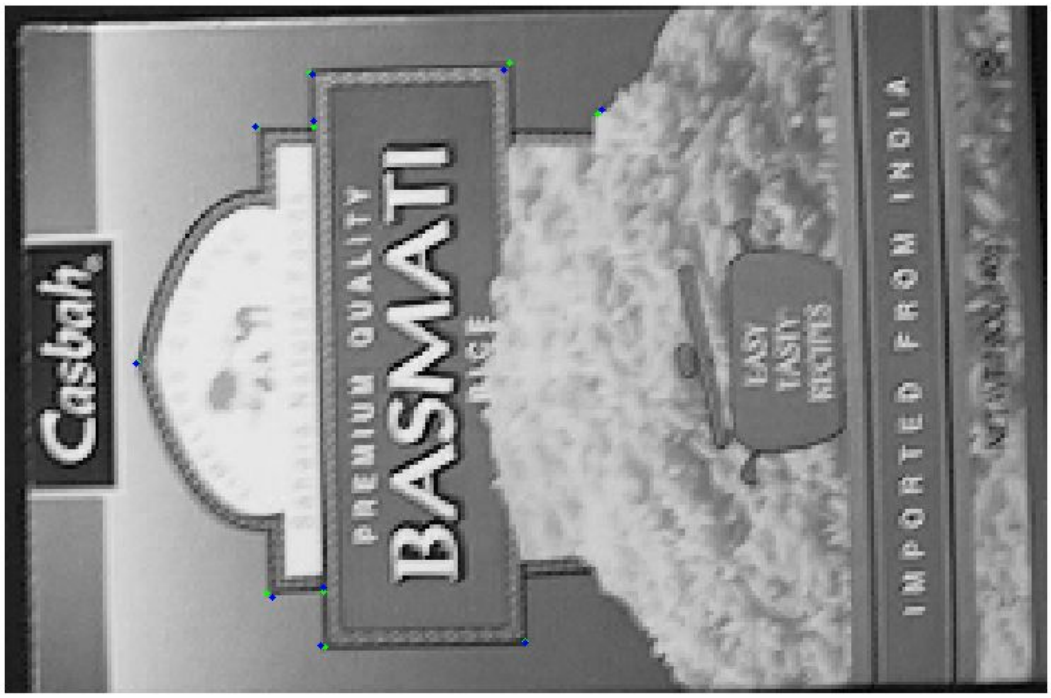


Figure 2. Manually marked points

Results:

Automatically Matched

H:

```
    0.0009    0.0021   -0.6197
   -0.0030    0.0012    0.7848
   -0.0000    0.0000    0.0020
```

Reprojection Error:

```
    0.7306
```

Manually Marked

H:

```
    0.0009    0.0021   -0.6160
   -0.0030    0.0013    0.7877
   -0.0000    0.0000    0.0021
```

Reprojection Error:

```
    1.0569
```

Code:

```
load img1keypts_manual.mat
load img2keypts_manual.mat

% For manually marking points
% image_pts = zeros(10,2);
% imshow('basmati.pgm');
% for i=1:10
%     image_pts(i,:) = ginput(1);
% end
% disp(image_pts);

corr_mat = zeros(size(img1keypts,1)*2, 9);
for i=1:size(img1keypts,1)
    for j=1:size(img1keypts,2)
        corr_mat(2*i-1,j) = img1keypts(i,j);
        corr_mat(2*i-1,j+6) =
-img2keypts(i,1)*img1keypts(i,j);
        corr_mat(2*i,j+3) = img1keypts(i,j);
        corr_mat(2*i,j+6) =
-img2keypts(i,2)*img1keypts(i,j);
    end
end
[u, d, v] = svd(corr_mat);
a = v(:,end);
h = reshape(a,3,3)';

disp('H: ');
disp(h);

reproject = (h*img1keypts')';
reproject_2d = [reproject(:,1)./reproject(:,3),
reproject(:,2)./reproject(:,3)];

err =
mean(sqrt(sum((reproject_2d-img2keypts).^2,2)));
```

```
disp('Reprojection Error: ');
disp(err);

imshow('basmati.pgm');
for i=1:size(reproject,1)
    hold on;
    plot(img2keypts(i,1), img2keypts(i,2), 'g.',
'MarkerSize', 10);
    plot(reproject(i,1)/reproject(i,3),
reproject(i,2)/reproject(i,3), 'b.', 'MarkerSize',
10);
end
```

RANSAC (Random Sample Consensus):

- From a set of n points, randomly select a few points to call the DLT function (in our case, 4 points).
- After 100 iterations of DLT on random points, we pick the points that showed least mean error.
- Mean error was calculated over all reprojected points, as the Euclidean distance of reprojected point with the original point in image coordinate system.
- The best **H** matrix is thus used for final reprojection.

Observations:

The green points marked were selected for DLT, and reprojected points after calculating the H matrix are shown in blue. Figure 1 shows results for automatically detected point matches, and Figure 2 shows results for manually marked points.

Figure 5. SIFT matched points

Figure 6. Manually marked points

Results:

Automatically Matched

H:

```
   -0.0009   -0.0021    0.6197
    0.0030   -0.0012   -0.7849
    0.0000   -0.0000   -0.0021
```

Reprojection Error:

```
    0.7678
```

Manually Marked

H:

```
    0.0009    0.0022   -0.6281
   -0.0030    0.0013    0.7781
   -0.0000    0.0000    0.0019
```

Reprojection Error:

```
    0.9853
```

Code:

```
load img1keypts_manual.mat
load img2keypts_manual.mat
best_h = zeros(3,3);
best_err = Inf;
for iter=1:100
    ind1 = randperm(size(img1keypts,1));
    ind1 = ind1(1:6);
    img1keypts_rand = img1keypts(ind1,:);
    img2keypts_rand = img2keypts(ind1,:);

    corr_mat = zeros(size(img1keypts_rand,1)*2, 9);
    for i=1:size(img1keypts_rand,1)
        for j=1:size(img1keypts_rand,2)
            corr_mat(2*i-1,j) =
img1keypts_rand(i,j);
            corr_mat(2*i-1,j+6) =
-img2keypts_rand(i,1)*img1keypts_rand(i,j);
            corr_mat(2*i,j+3) =
img1keypts_rand(i,j);
            corr_mat(2*i,j+6) =
-img2keypts_rand(i,2)*img1keypts_rand(i,j);
        end
    end
    [u, d, v] = svd(corr_mat);
    a = v(:,end);
    h = reshape(a,3,3)';
    reproject = (h*img1keypts')';
    reproject_2d = [reproject(:,1)./reproject(:,3),
reproject(:,2)./reproject(:,3)];

    curr_err =
mean(sqrt(sum((reproject_2d-img2keypts).^2,2)));
    if curr_err < best_err
        best_err = curr_err;
        best_h = h;
```

```
    end

end

disp('H: ');
disp(best_h);

reproject = (best_h*img1keypts')';
reproject_2d = [reproject(:,1)./reproject(:,3),
reproject(:,2)./reproject(:,3)];

disp('Reprojection Error: ');
disp(best_err);

imshow('basmati.pgm');
hold on;

for i=1:size(reproject,1)
    hold on;
    plot(img2keypts(i,1), img2keypts(i,2), 'g.',
'MarkerSize', 12);
    plot(reproject(i,1)/reproject(i,3),
reproject(i,2)/reproject(i,3), 'b.', 'MarkerSize',
10);
end
```

Reprojection Error Comparison:

Method	Automatic Match	Manually Marked
DLT	0.7306	1.0569
RANSAC	0.7678	0.9853

The error related to RANSAC changes every time the code is run, since there are around 33 automatically matched points, and 10 manually marked points, but only 4 are randomly picked in each iteration, for 100 iterations. Many combination of points are not considered because of this.

Mosaicing (Same camera center):

Input Images:

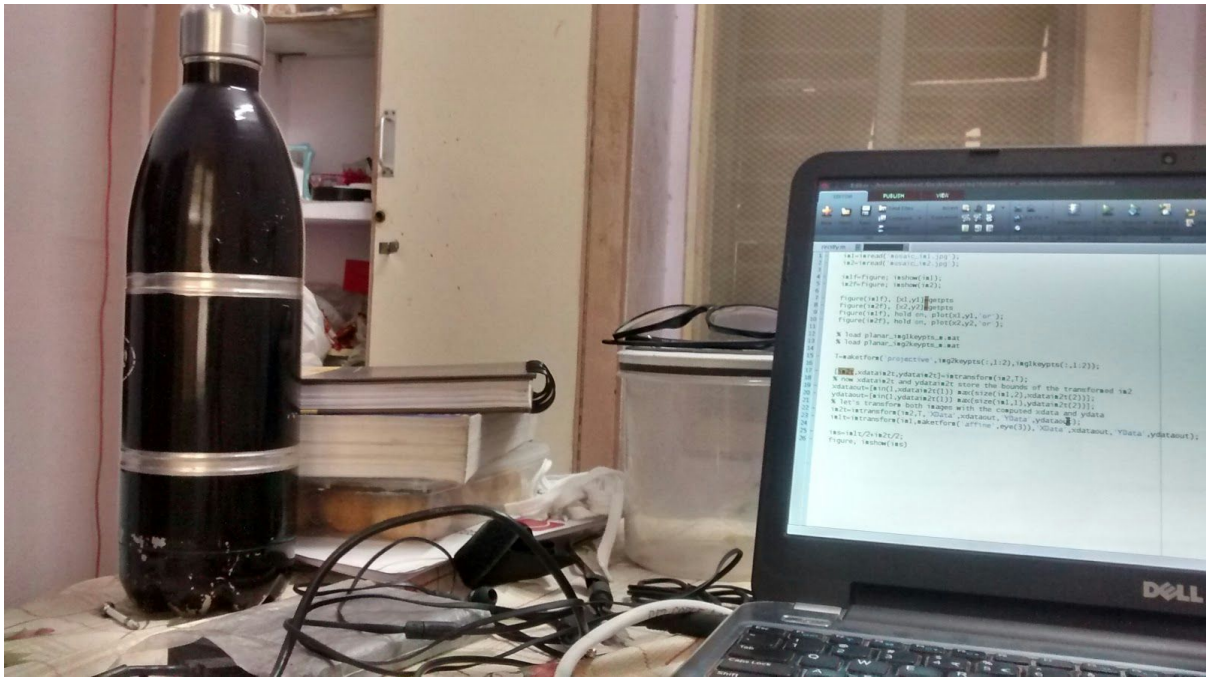


Image 1.

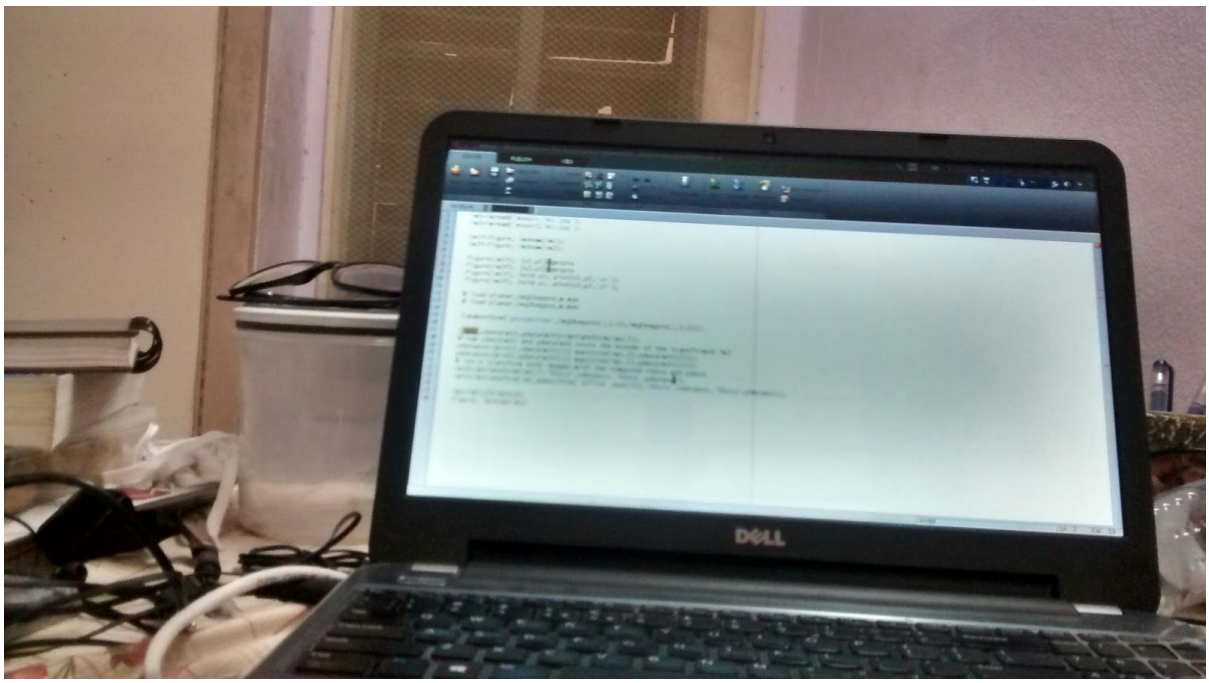
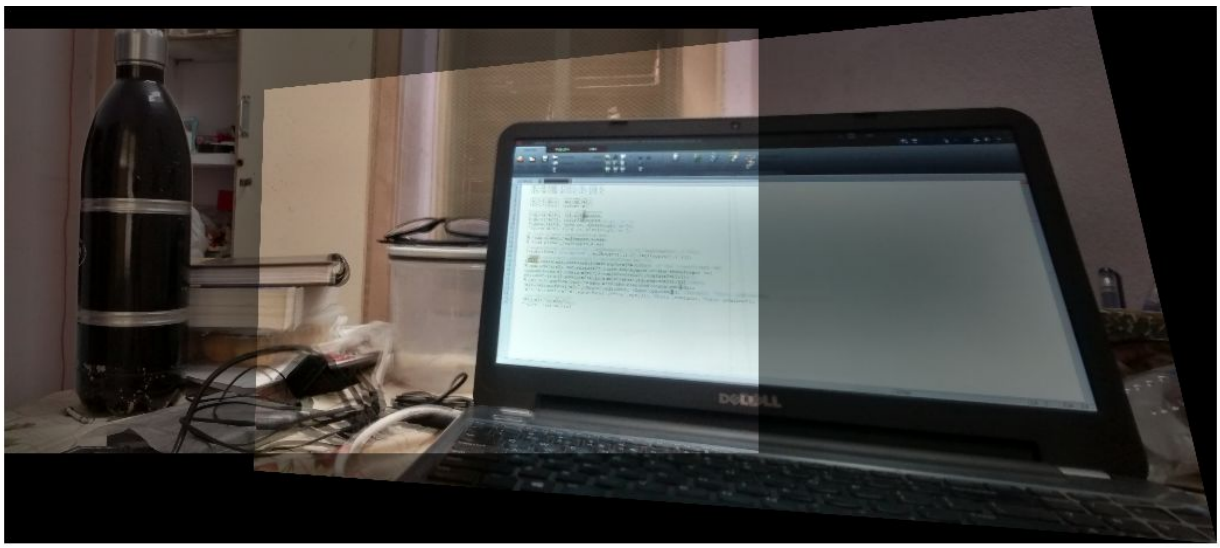


Image 2.

Resulting Image:



Code:

```
im1=imread('mosaic_im1.jpg');
im2=imread('mosaic_im2.jpg');
load mosaic_img1keypts_m.mat
load mosaic_img2keypts_m.mat

T=maketform('projective',img2keypts(:,1:2),img1keypt
s(:,1:2));

[im2t,xdataim2t,ydataim2t]=imtransform(im2,T);
% now xdataim2t and ydataim2t store the bounds of
the transformed im2
xdataout=[min(1,xdataim2t(1))
max(size(im1,2),xdataim2t(2))];
ydataout=[min(1,ydataim2t(1))
max(size(im1,1),ydataim2t(2))];
% let's transform both images with the computed
xdata and ydata
im2t=imtransform(im2,T,'XData',xdataout,'YData',ydat
aout);
im1t=imtransform(im1,maketform('affine',eye(3)),'XDa
ta',xdataout,'YData',ydataout);
```

```
ims=im1t/2+im2t/2;
figure, imshow(ims)
```

Mosaicing (Planar scene):

Input Images:

Resulting Image:

Code:

```
im1=imread('planar_im1.jpg');
im2=imread('planar_im2.jpg');
load planar_img1keypts_m.mat
load planar_img2keypts_m.mat

T=maketform('projective',img2keypts(:,1:2),img1keypt
s(:,1:2));

[im2t,xdataim2t,ydataim2t]=imtransform(im2,T);
% now xdataim2t and ydataim2t store the bounds of
the transformed im2
```

```
xdataout=[min(1,xdataim2t(1))
max(size(im1,2),xdataim2t(2))];
ydataout=[min(1,ydataim2t(1))
max(size(im1,1),ydataim2t(2))];
% let's transform both images with the computed
xdata and ydata
im2t=imtransform(im2,T,'XData',xdataout,'YData',ydat
aout);
im1t=imtransform(im1,maketform('affine',eye(3)),'XDa
ta',xdataout,'YData',ydataout);

ims=im1t/2+im2t/2;
figure, imshow(ims)
```

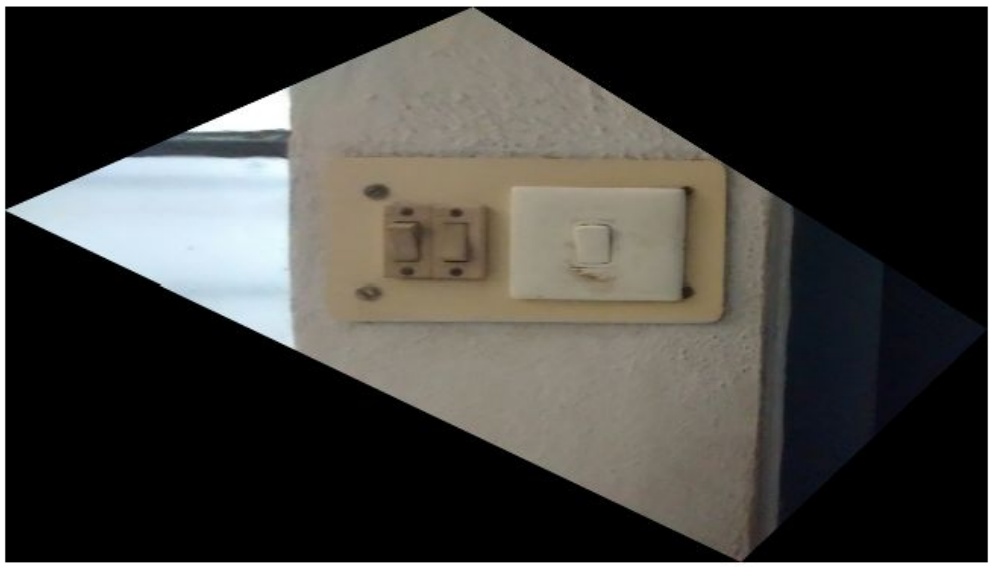
Rectification of Perspective Distortion:

Input Images:

Image 1. Fronto-parallel view

Image 2, 3, 4. Input Distorted images

Resulting Images:



Rectified Images

Code:

```
load plane_1.mat
load plane_3.mat

T=maketform('projective',image_pts(:,1:2),front_pts(
:,1:2));

im = imread('plane_image_3.png');

rect_img = imtransform(im, T, 'Size', size(im));
% figure, imshow(im);
figure, imshow(rect_img);
```

Image Overlay:

Input Images:

Image 1. Overlay image

Image 2. Sports image, Field

Image 3. Sports image, Table

Image 4. Sports image, Pool

Resulting images:

Code:

```
close all;

load pool.mat

flag_keypts = [0 0 1; 0 450 1; 300 0 1; 300 450 1];

corr_mat = zeros(size(flag_keypts,1)*2, 9);
for i=1:size(flag_keypts,1)
    for j=1:size(flag_keypts,2)
        corr_mat(2*i-1,j) = flag_keypts(i,j);
        corr_mat(2*i-1,j+6) =
-image_pts(i,1)*flag_keypts(i,j);
        corr_mat(2*i,j+3) = flag_keypts(i,j);
        corr_mat(2*i,j+6) =
-image_pts(i,2)*flag_keypts(i,j);
    end
end
[u, d, v] = svd(corr_mat);
a = v(:,end);
h = reshape(a,3,3)';

disp('H: ');
disp(h);

imshow('flag.png');
for i=1:size(flag_keypts,1)
    hold on;
    plot(flag_keypts(i,2)+1, flag_keypts(i,1)+1,
'b.', 'MarkerSize', 10);
end
figure;

reproject = (h*flag_keypts')';
imshow('pool.jpg');
for i=1:size(reproject,1)
```

```
    hold on;
    plot(image_pts(i,1), image_pts(i,2), 'g.',
'MarkerSize', 10);
    plot(reproject(i,1)/reproject(i,3),
reproject(i,2)/reproject(i,3), 'b.', 'MarkerSize',
10);
end
figure;

flag_image = imread('flag.png');
overlay_image = imread('pool.jpg');
for i=1:size(flag_image,1)
    for j=1:size(flag_image,2)
        projection = (h*[i j 1]')';

overlay_image(round(projection(2)/projection(3)),
round(projection(1)/projection(3)),:) =
flag_image(i, j, :);
    end
end
imshow(overlay_image);
```