ECS 174 Project 2

Jiayi Lei
SID: 915016668
Email: cjlei@ucdavis.edu

Zhengfeng Lai
SID :916678034
Email: lzhengfeng@ucdavis.edu

May 20, 2019

1 Short Answer Questions

1.1 1

For clustering points in a continuous vote space, mean-shift would be appropriate. Since it is less time consuming compare to graph-cuts algorithm and easier to run without human supervising compare to k-mean algorithm.
1. K-means algorithm is an algorithm that tries to cluster points into k groups. The goal of this algorithm is to minimize the sum of distance from points to the center of the group where the points are belonged to. Noticed that we need to put in parameter k into the algorithm, which is sometimes hard to pick. Also, the output will be affected by the initial group centers and the outliers. It needs evaluation of the distance between the hypothesized center to all bins, which is impossible to do with a continuous vote space.
Mean-shift is an algorithm that moves toward the attraction basin, i.e. the center of the dynamic kernel region. Compare to k-means algorithm, mean-shift only needs one parameter and can find multiple modes. However, sometimes it is hard to pick the appropriate window size.
Graph-cuts is an algorithm that tries to minimize the intra distance between points from the same group and maximize the cut value between clusters. The downside of this algorithm is that it is time consuming to run it. It is applied to the situation where the space has been discretized into single units like pixels in an image rather than the case mentioned above.

1.2 2

K-means algorithm, utilizing Euclidian distance as distinguishing criteria, it may have unexpected clustering. If the given dataset is two circles, the results would be two half circles for each cluster rather than two circles by human clustering. And k-means plays poor in clusters within clusters. It attempts to find the center of clusters and then try to minimize the sum of squared differences among all the points near the cluster centers. Therefore, it cannot make the image given, it can only split the circles in half cleanly.

1.3 3

```


3. First, find the center of mass point for each blob.
Then, extract radius invariant circularity feature.
Finally, cluster the blobs according to their circularity feature.
```

```
1) Find the center of mass point for each blob:
for each pixel in blob:
 sum_of_pos += pos(pixel)
 center_of_mass = sum_of_pos/size(blob)
 return center_of_mass
2) Extract radius invariant circularity feature:
radius_mean = size(blob)
for each pixel in boundary(blob):
 sum_square_distance += (center_of_mass−position(pixel))^2
 circularity = sum_square_distance/size(boundry(blob))
 return circularity
3) Clusters k−means(circularity[],k)
 return k−means = circularity[]
```

2 Short Programming

2.1 (a)

Matlab Code

```
function [points] = GetPoints(img)
flag = false;
while ~flag
    imshow(img);
    title('Press Enter When Done');
    hold on;
    [x,y] = ginput();
    points = [x,y]';
    plot(x,y,'rx','linewidth',2);
    for i = 1:size(x,1)
        text(x(i),y(i)−10,num2str(i));
    end
    flag = true;
    hold off;
end
end
```

2.2 (b)

computedH.m is attached in the folder we submitted and also the code is attatched in the appendix section for your convenience to read.

2.3 (c)

warpImage.m is attached in the folder we submitted and also the code is attatched in the appendix section for your convenience to read.

2.4 (d)

points.mat is attatched in the folder.
Output for crop1.jpg and crop2.jpg

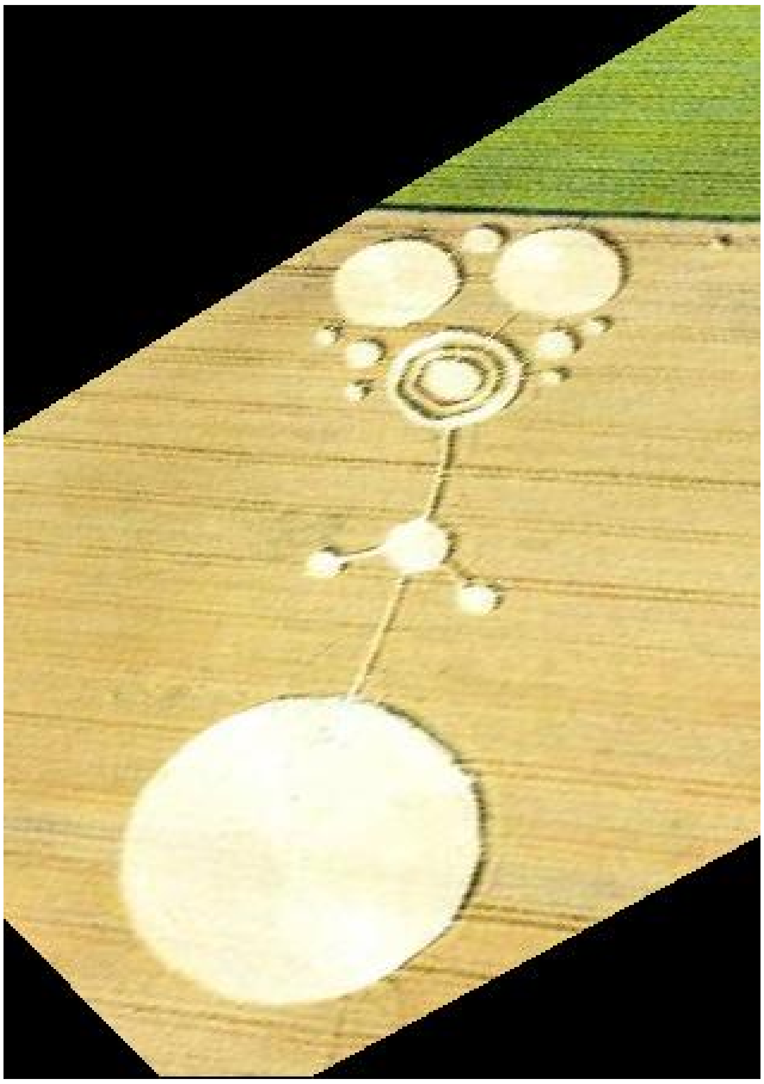


Figure 1: Warped Image of crop 1

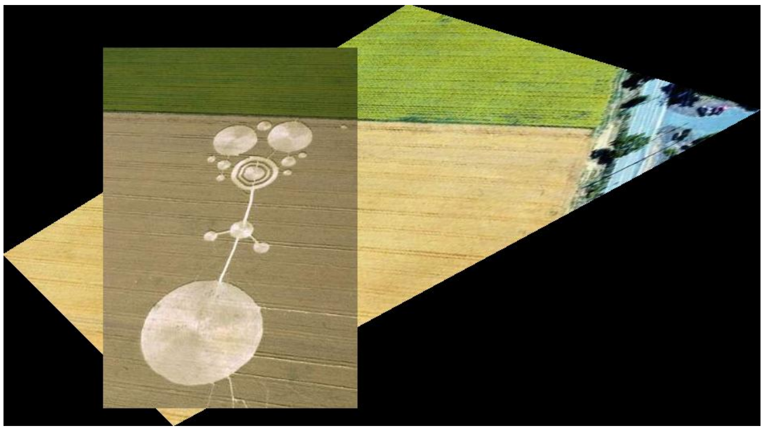


Figure 2: Merge Image of crop1 and crop2

Output for wdc1.jpg and wdc2.jpg

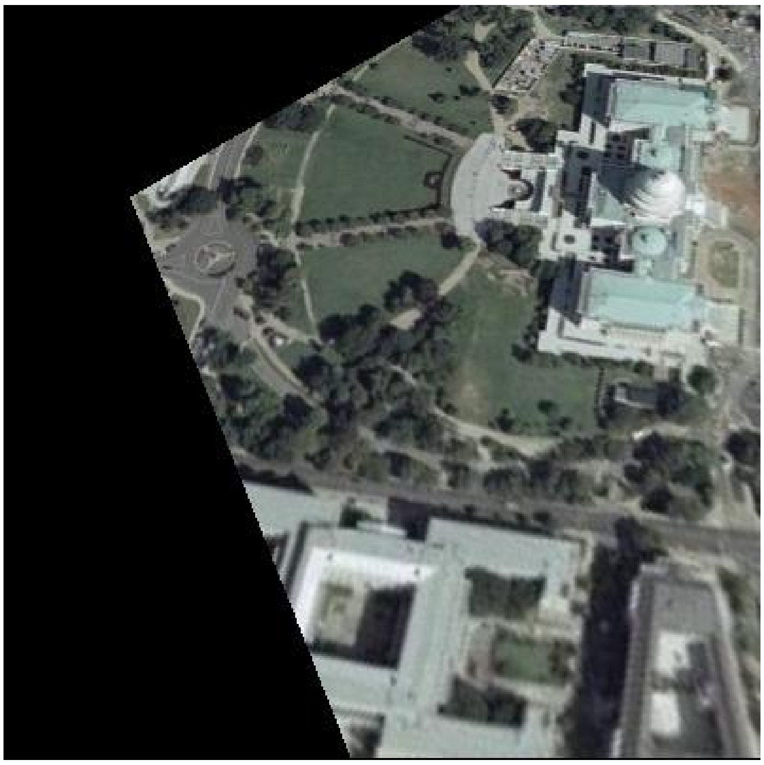


Figure 3: Warped Image of wdc1

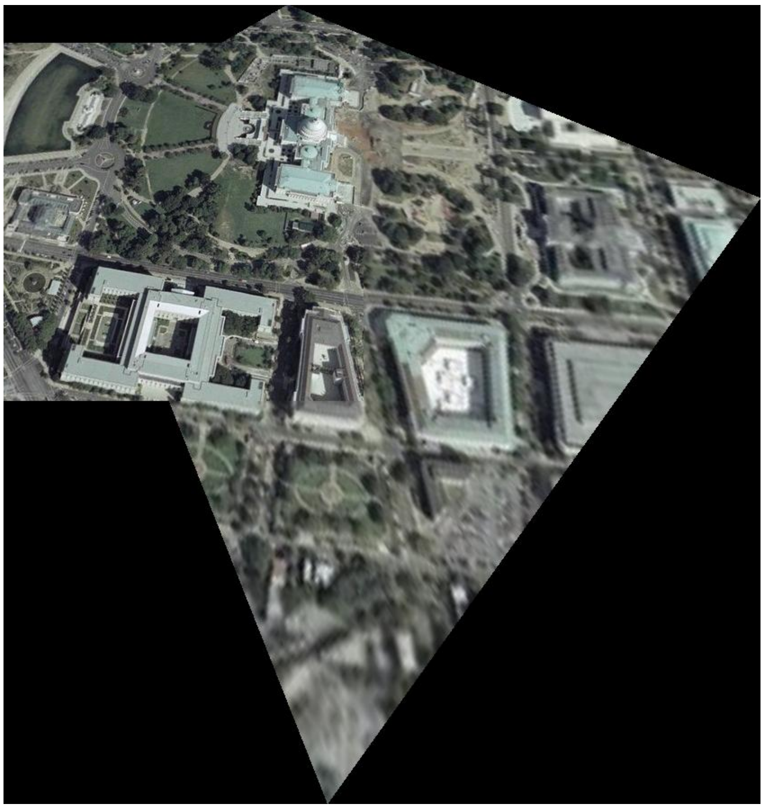


Figure 4: Merge Image of wdc1 and wdc2

2.5 (e)

Another Example with MU1 and MU2

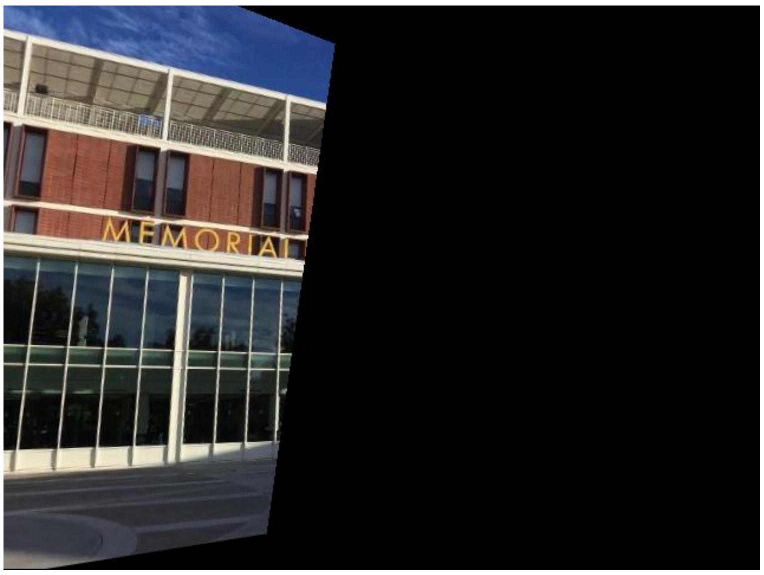


Figure 5: Warped Image of MU1

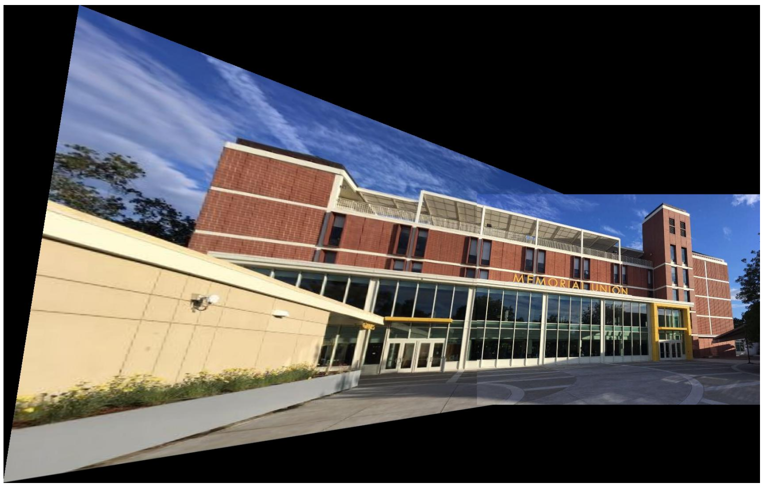
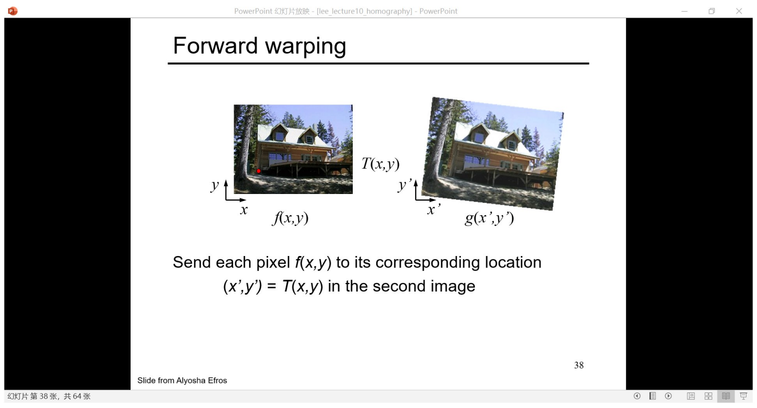


Figure 6: Merge Image of MU1 and MU2

2.6 (f)

Input image and billboard image:

In this problem, we are trying to merge the image of the slide into the empty rectangle in the board.



PowerPoint 幻灯片放映 - [lee_lecture10_homography] - PowerPoint

Forward warping

Send each pixel $f(x,y)$ to its corresponding location $(x',y') = T(x,y)$ in the second image

38

Slide from Alyosha Efros

幻灯片 第 38 张，共 64 张

Figure 7: slid.png

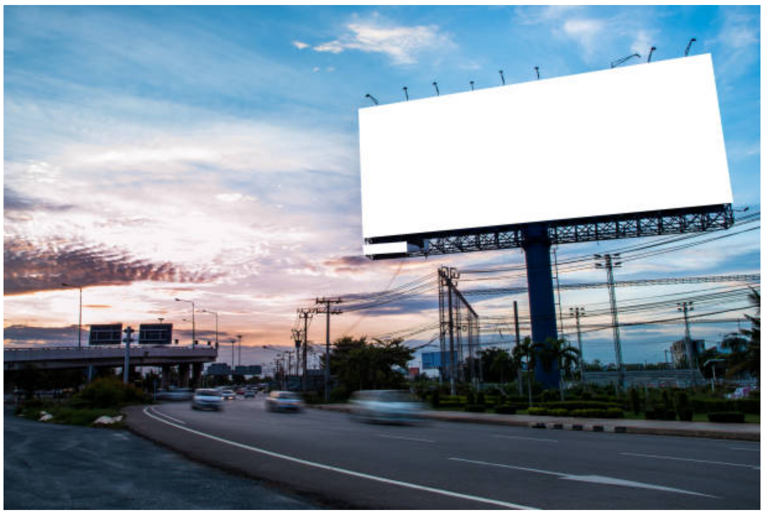
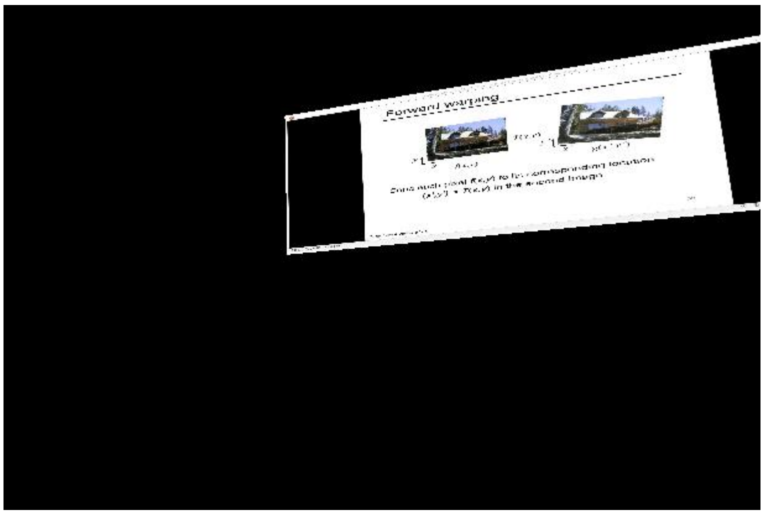


Figure 8: board.png



Forward warping

Send each pixel f(x,y) to its corresponding location (x',y') = T(x,y) in the second image

Figure 9: slide_warp.png

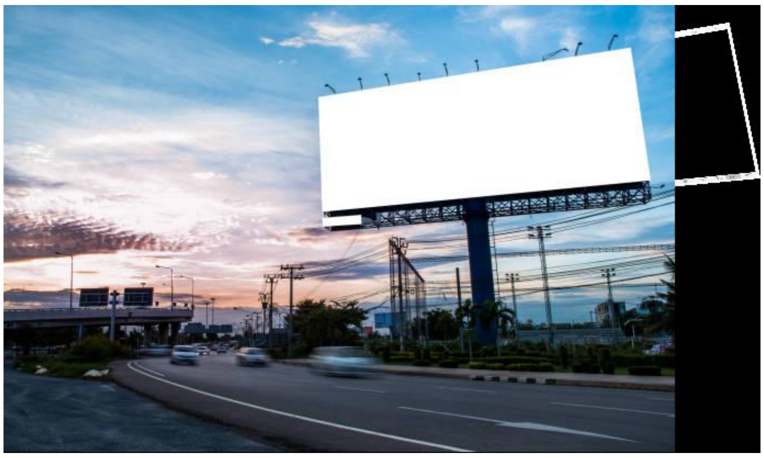


Figure 10: slide_merge.png

3 Extra Credit

```
inputIm = imread('crop1.jpg');
refIm = imread('crop2.jpg');
t1 = importdata('cc1.mat');
t2 = importdata('cc2.mat');
H = RANSAC(t1,t2);
disp(H);
[warpIm, mergeIm] = warpImage(inputIm, refIm, H);
imwrite(warpIm,'crop_warp_RANSAC.jpg');
imwrite(mergeIm,'crop_merge_RANSAC.jpg');
```

OUTPUTS:

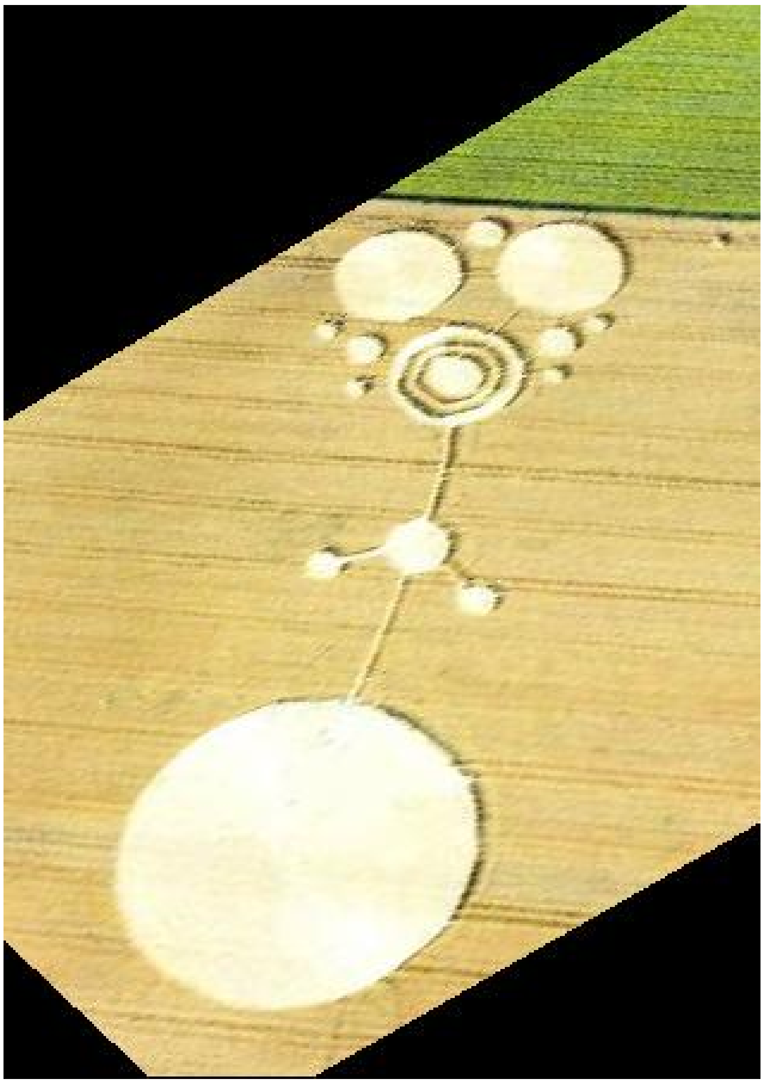


Figure 11: Warped Image by using RANSAC

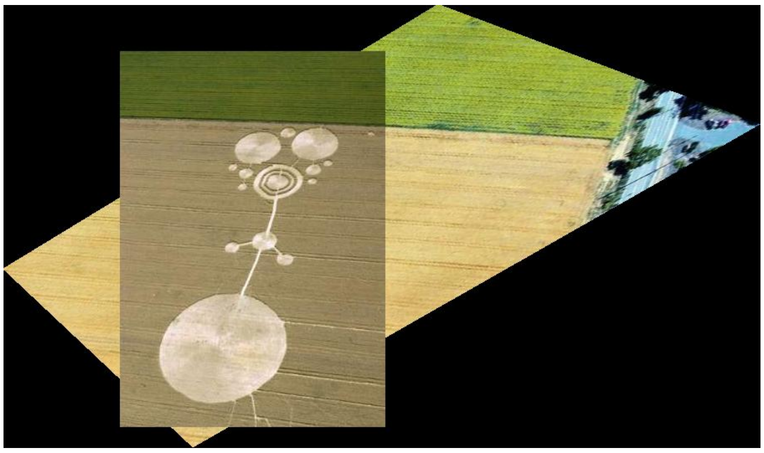


Figure 12: Merge Image oby using RANSAC

In Comparision, outputs wiht non-RANSAC

We can not see too much difference between outputs given by RANSAC and outputs given by Non-RANSAC.

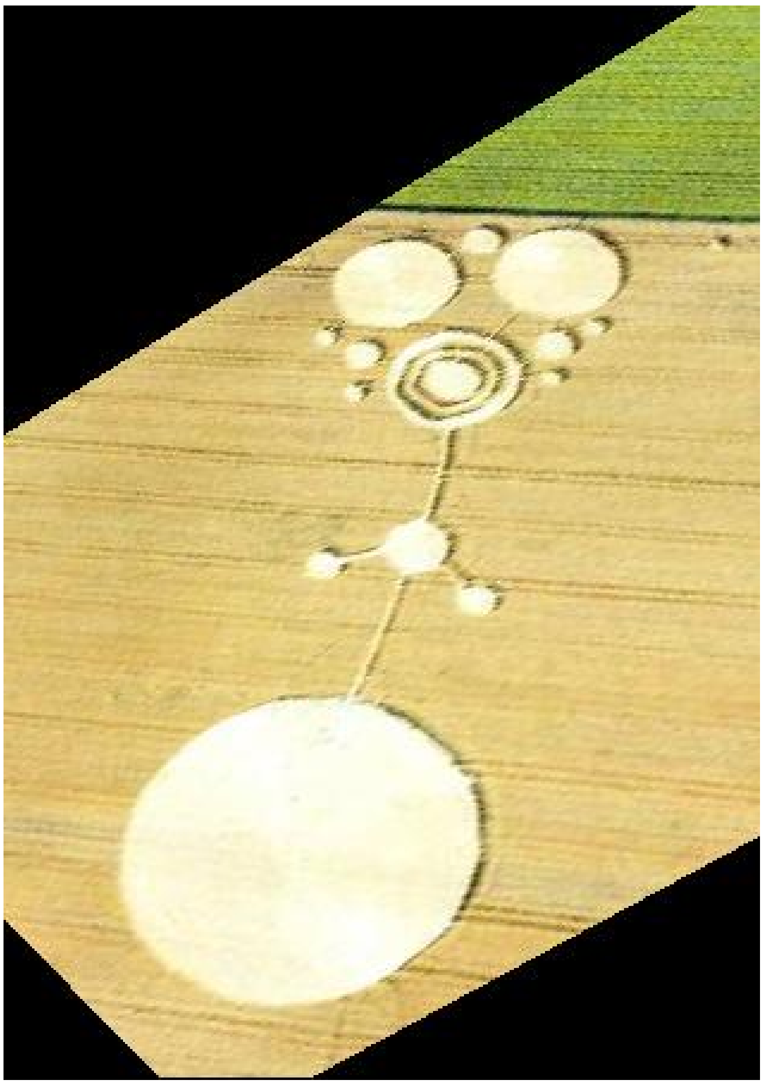


Figure 13: Warped Image of crop 1

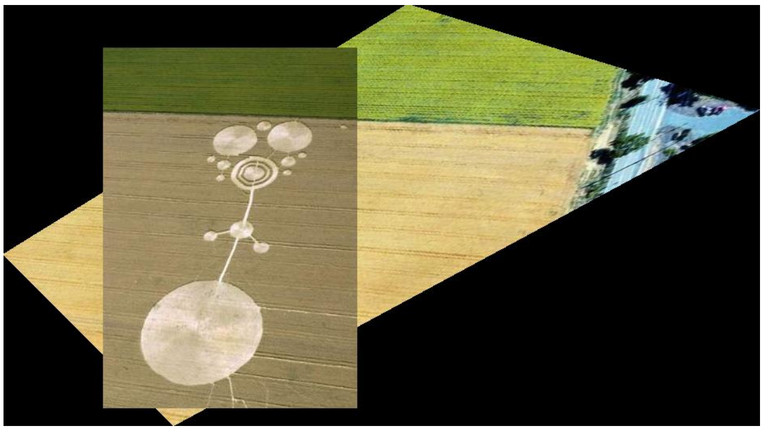


Figure 14: Merge Image of crop1 and crop2

4 Appendix

4.1 computeH.m

Matlab Code

```
function H = computeH(t1, t2)
% compute the associated 3x3 homography matrix H
% input: t1,t2
% t1,t2: a set of corresponding image points
% t1,t2: 2xN matrices
% H: associated 3x3 homography matrix
% n >= 4 pairs
t1 = t1';
t2 = t2';
A = [];
for i = 1 : length(t1)
    A_part = [t1(i,1) t1(i,2) 1 0 0 0 (-t1(i,1) * t2(i,1)) (-t1(i,2)*t2(i,1)) -t2(i,1);
              0 0 0 t1(i,1) t1(i,2) 1 (-t1(i,1) * t2(i,2)) (-t1(i,2)*t2(i,2)) -t2(i,2)];
    A = [A;A_part];
end
[V,~] = eig(A'*A);
H = V(:,1);
H = reshape(H,3,3)';
end
```

4.2 warpImage.m

Matlab Code

```matlab
function [warpIm, mergeIm] = warpImage(inputIm, refIm, H)
inputIm = double(inputIm);
refIm = double(refIm);

[height,width,dim] = size(refIm);
[hi,wi,~] = size(inputIm);
warpIm = zeros(height,width,dim);

H_inv = H^-1;
frame = H*[1,1,wi,wi;
           1,hi,1,hi;
           1,1,1,1];

frame = frame./frame(3,:);

max_frame1 = max([frame(1,:),width]);
min_frame1 = min([frame(1,:),0]);
max_frame2 = max([frame(2,:),height]);
min_frame2 = min([frame(2,:),0]);
max_x = fix(max_frame1) - fix(min_frame1);
max_y = fix(max_frame2) - fix(min_frame2);
mergeIm = zeros(max_y,max_x,dim);


for i=1:max_y
    for j=1:max_x
        s = H_inv*[j+fix(min_frame1);
                   i+fix(min_frame2);
                   1];
        x = s(2)/s(3);
        y = s(1)/s(3);
        if x>=1&&y>=1&&x<=hi&&y<=wi
            mergeIm(i,j,:) = (fix(x+1)-x)*(fix(y+1)-y)*inputIm(fix(x),fix(y)
                ,:) ...
                                + (x-fix(x))*(fix(y+1)-y)*inputIm(fix(x+1),fix(y
                                  ),:) ...
                                + (x-fix(x))*(y-fix(y))*inputIm(fix(x+1),fix(y
                                  +1),:) ...
                                + (fix(x+1)-x)*(y-fix(y))*inputIm(fix(x),fix(y
                                  +1),:);
        end
        if j>-fix(min_frame1)&&i>-fix(min_frame2)&&j<width+1-fix(min_frame1)
            &&i<height-fix(min_frame2)
            warpIm(i+fix(min_frame2),j+fix(min_frame1),:) = mergeIm(i,j,:);
            mergeIm(i,j,:) = refIm(i+fix(min_frame2),j+fix(min_frame1),:);
        end
    end
end
warpIm = uint8(warpIm);
mergeIm = uint8(mergeIm);
end
```