Roy Dey 914191568
Patrick Liao 913731408

PS2

Part 1

1. Having a continuous vote space implies that each m or θ in the Hough space is guaranteed to have some value greater than 0 for a varying b or d. Furthermore, there will be a lot more noisy data in the space as well, assuming that no extension methods are used to filter out votes.

   In this case, graph-cuts would be most computationally expensive. The graph cuts algorithm requires each weighted edge for every possible pair of points in the vote space to be calculated. This is unrealistic due to the large number of data points that is to be expected from continuous vote space. Also, this algorithm finds segmentations by eliminating edges with the lowest weight, which does not directly achieve our goal of finding points with the most intersections.

   The mean-shift algorithm would be the most optimal algorithm. Points in the vote space with a large number of intersections will naturally become points of center of mass, so the search window would quickly converge to these points. The runtime of this algorithm is affected by the search window size, so this can also be adjusted accordingly. This algorithm would work better than k-means since k is unknown. Convergence would take longer as well, since it needs to take the additional step of re-defining the group of points to calculate for each iteration.

2. A likely clustering would result in half the image being clustered separately from the other half. The orientation of this clustering is completely random, since the initial center points are also initialized at random. Since k-means clustering is assigned according to the shortest distance to the center point, it is safe to assume that the two center points of each cluster will converge to two opposite halves of the circle.

3. group_blobs(blobs, k): // blobs is an array of values for area of each blob
   k_vals = set k random area values in the range of min and max blob areas
   k_groups = matrix with size of k_vals rows, each row contains blobs that have area closest to corresponding k_val
   while true:
   for each row in k_groups, k_val = avg(k_groups)
   k_groups = re-calculate each row to contain blobs that have area closest to corresponding k_val
   if k_groups doesn't change: break
   return k_groups

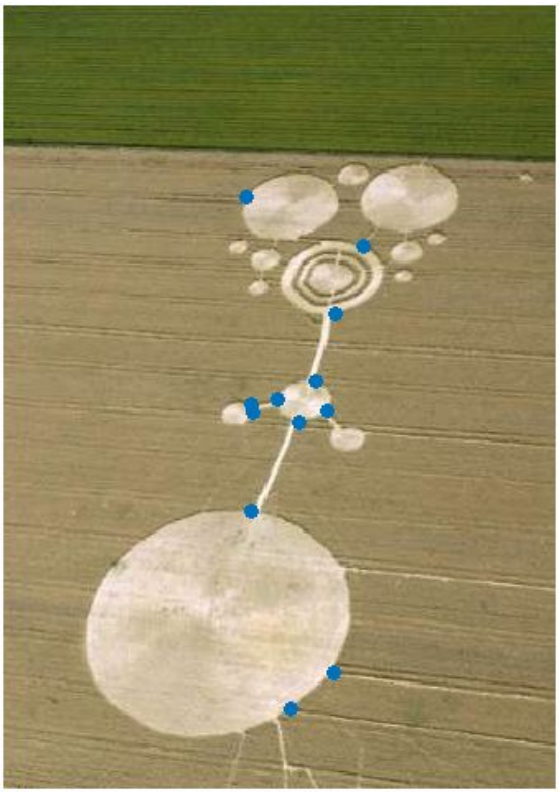
Part 2

1. script

2.

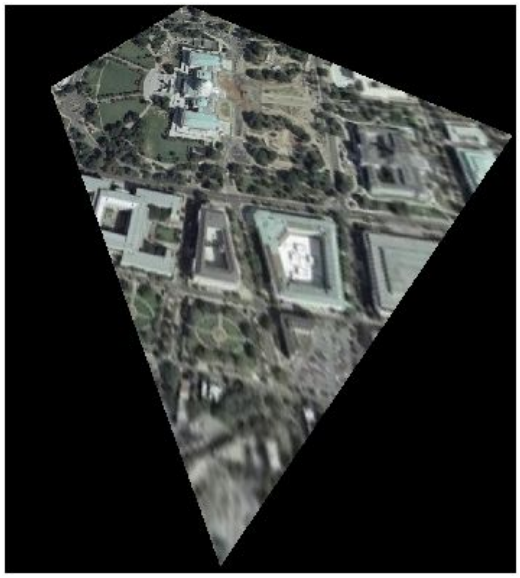
Crop 1 Points

Crop 2 Points (Calculated Using H Matrix)

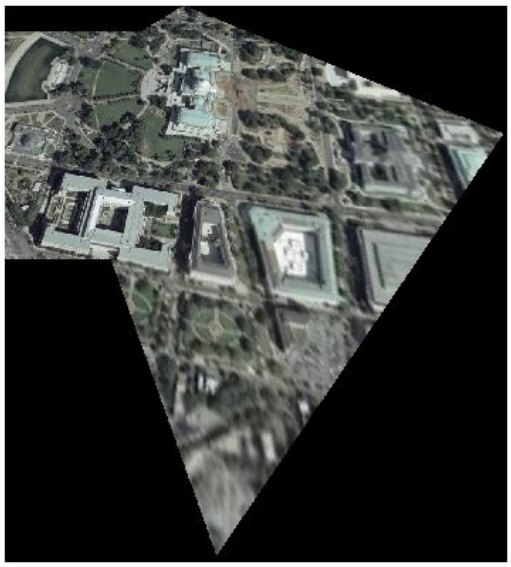


3. script

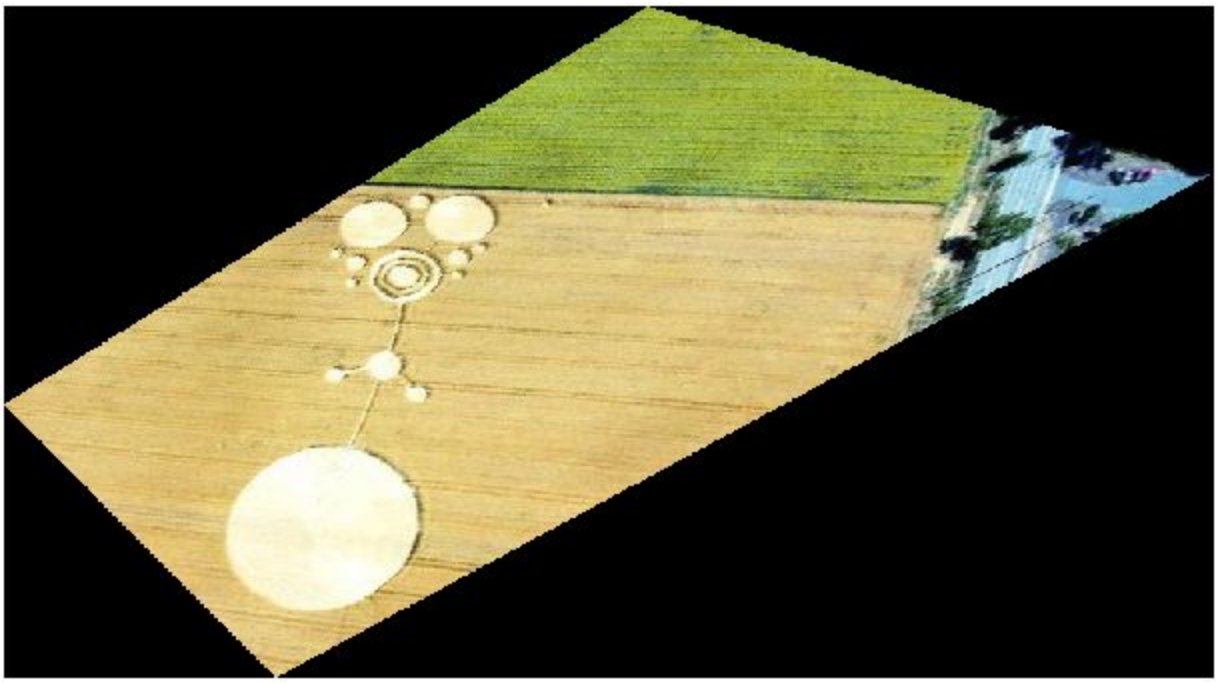
4. **Warp image**



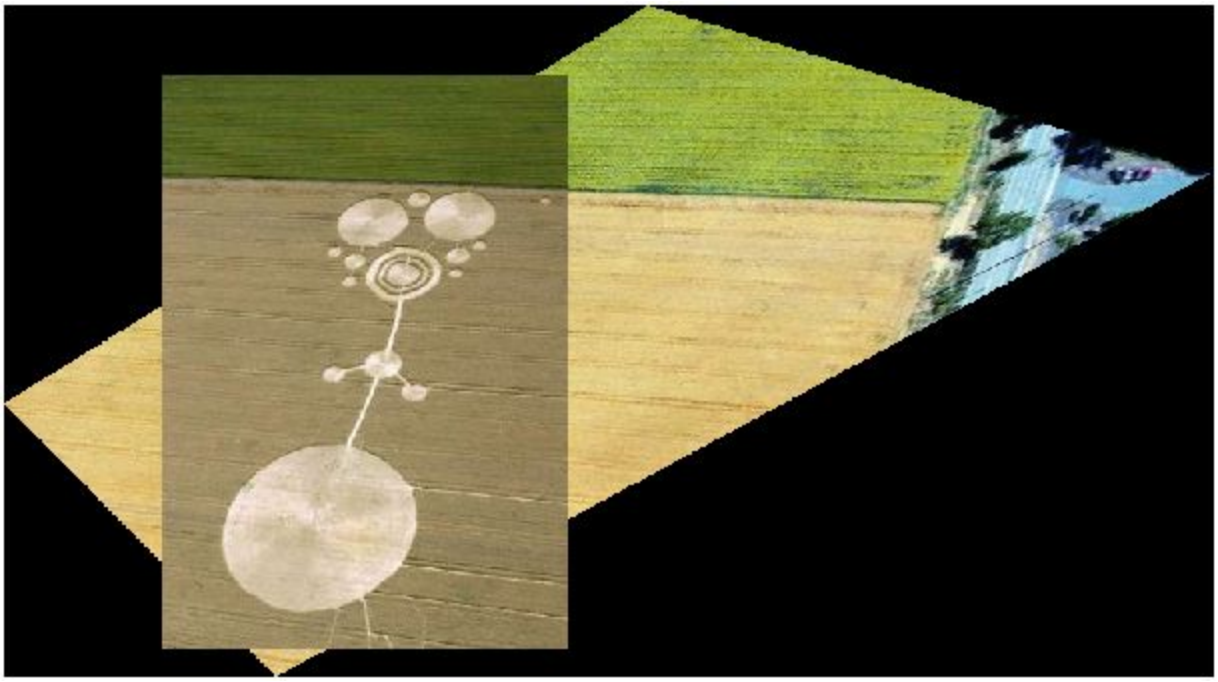
Merge image



Warp Image



Merge Image

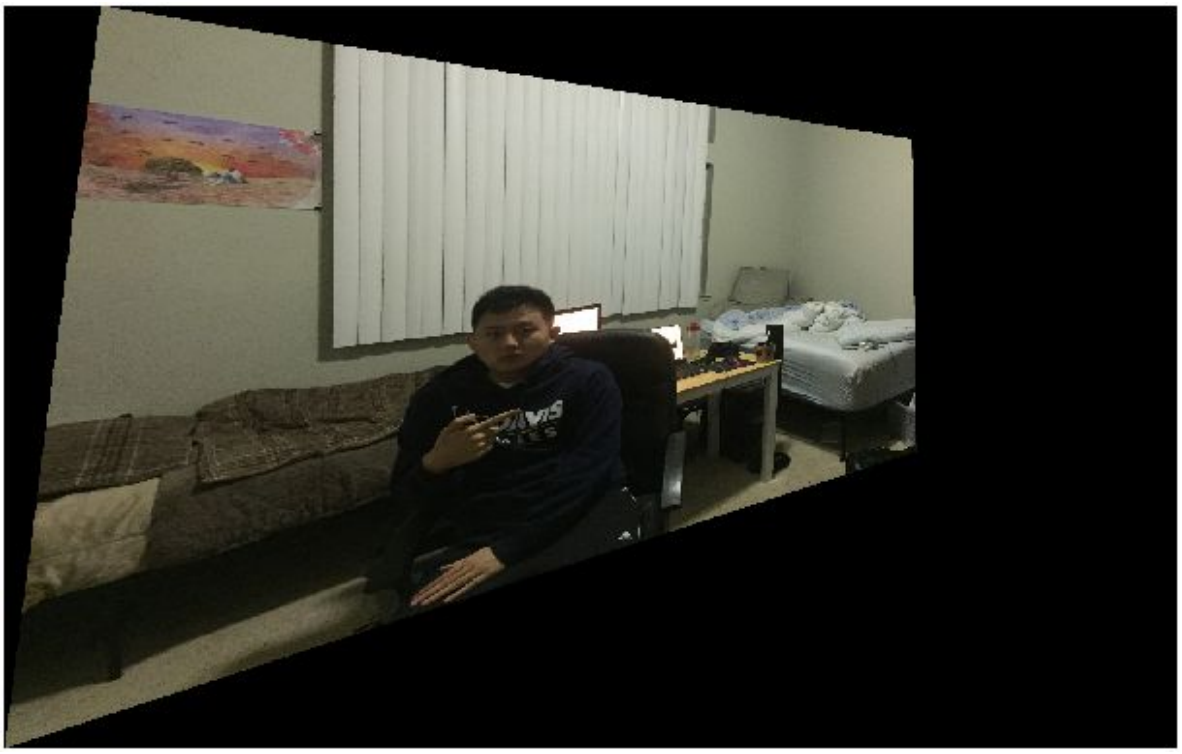


5.

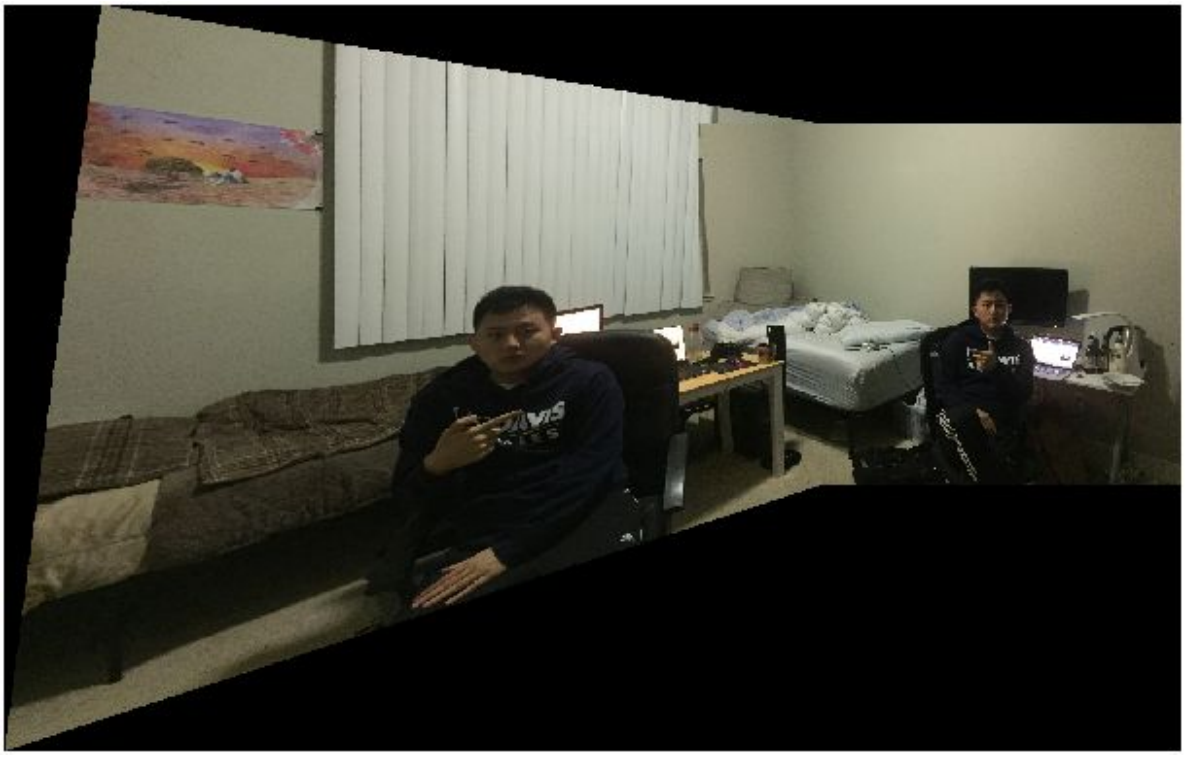
Input Image

Reference Image

Warp Image



Merged Image



6.

Input Image

Reference Image

Warp Image

Merge Image