Assignment 04: Image Registration

1 Part 01: Registration of two images (books) using 08 hand selected matched points

- Main Octave Script: (`main_script1.m`)
  First run this script.

```
clear
pkg load image

% Initialization of the mosaic area
I_mos = zeros(400,400);

img1 = imread('books0.pgm');
img2 = imread('books1.pgm');

% 8 Hand selected points from the image books0.pgm
p1 = [40,7,87,81,147,244,271,169;
30,171,230,147,20,36,164,90;
1,1,1,1,1,1,1,1];

% 8 Hand selected points from the image books1.pgm
p2 = [63,13,83,87,170,266,272,183;
15,146,215,131,15,41,173,85;
1,1,1,1,1,1,1,1];

% Calcalate Homography
H = my_homography(p1,p2);

% Initial transformation matrix (simple translation)
M = [ 1 0 -50; 0 1 -75; 0 0 1];

% Warp the image 'img1' into the mosaic 'I_mos' using ...
    transformation matrix M
Iw = warp(img1,M,size(I_mos));
pix_to_update = warp(ones(size(img1)),M,size(I_mos)) > 0;
I_mos(pix_to_update) = Iw(pix_to_update);

% Warp the image 'img2' into the mosaic 'I_mos' using ...
    transformation matrix H*M
Iw = warp(img2,H*M,size(I_mos));
pix_to_update = warp(ones(size(img2)),H*M,size(I_mos)) > 0;
I_mos(pix_to_update) = Iw(pix_to_update);

imwrite(uint8(I_mos),'out.jpg');
```

- My homography function (Using least square method): (`my_homography.m`)

```
    function H = my_homography(p1,p2)
        % This function uses Least Sqaure method to find the ...
           homographic matrix.
        % Here p1 and p2 is the set of matched points.

        %Initialization
        number_of_points = size(p1,2);
        A = [];
        b = [];

        % Constructing A and b for the least square approach ...
           considering P_33 = 1.
        for i = 1:number_of_points
            A = [A; p1(1,i), p1(2,i), p1(3,i), 0, 0, 0, -p1(1,i) * ...
               p2(1,i), -p1(2,i) * p2(1,i)];
            A = [A; 0, 0, 0, p1(1,i), p1(2,i), p1(3,i), -p1(1,i) * ...
               p2(2,i), -p1(2,i) * p2(2,i)];
            b = [b; p2(1,i); p2(2,i)];
        end

        % Solution using least square equation
        x = inv(A' * A) * A' * b;

        % Rearranging the solution into target 3x3 matrix
        H = [x(1),x(2),x(3); x(4),x(5),x(6); x(7),x(8),1];
    end
```

- Other Octave Functions:
  - `resample.m` (Given with the assignment)
  - `warp.m` (Given with the assignment)
- Input Images:

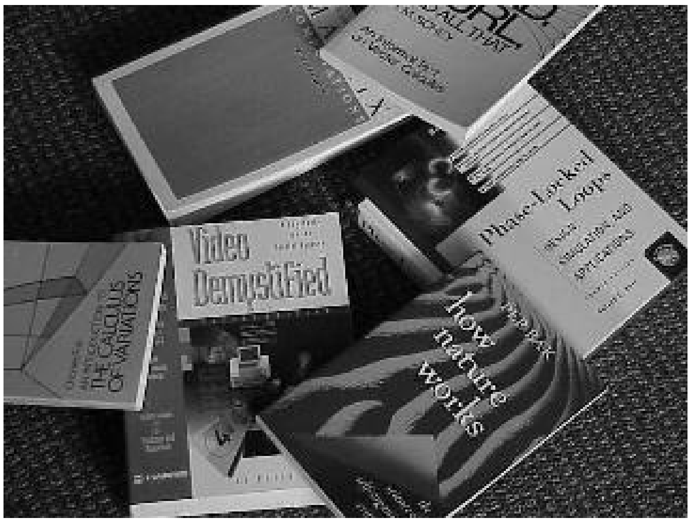

Figure 1: Input image 1(books0.pgm).

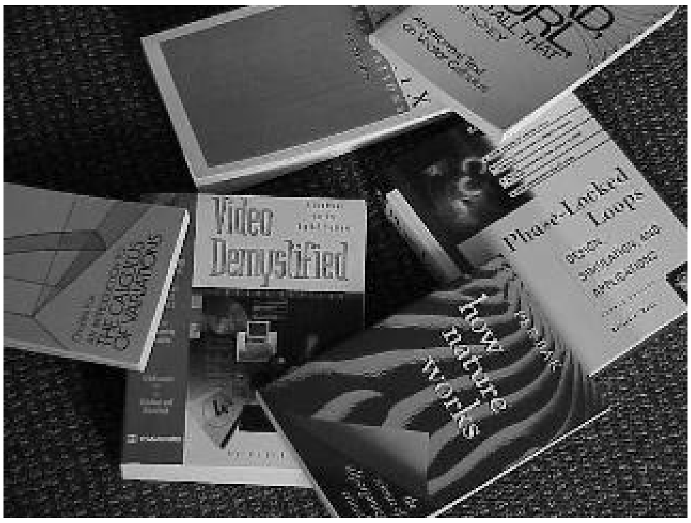
Figure 2: Input image 2(books1.pgm).

- Output image (after registration):

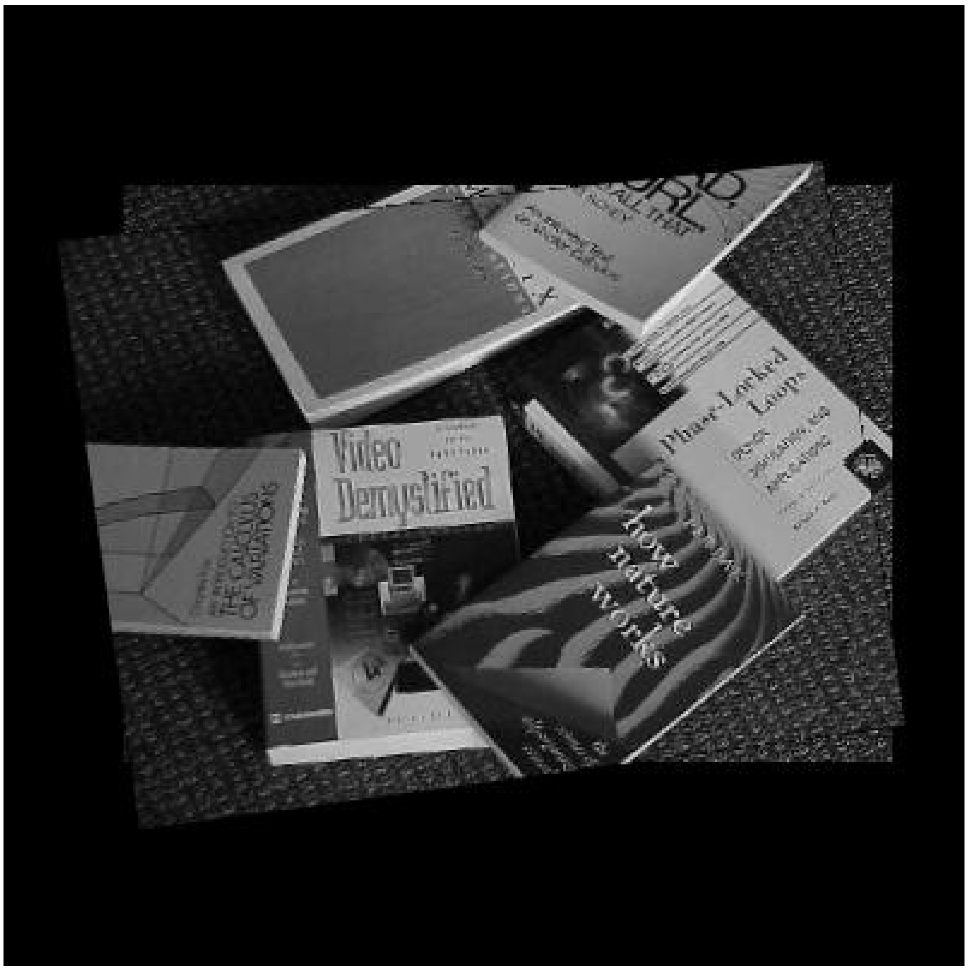
Figure 3: Output image (out.jpg).

2 Part 02: Registration of 07 images (Spruce Goose images) using SIFT and RANSAC

- Main Octave Script: (`main_script2.m`)
  First run this script.

```
clear
pkg load image
tic()

% Initialization of panaromic mosaic
I_mos = zeros(1100,2500);

% Initial transformation matrix (simple translation)
M1 = [ 1 0 -1000;0 1 -300;0 0 1];

% Place the first image at the center of the mosaic. I have started ...
    with goose.3 image.
% Then register goose.2, goose.1 and goose.0 (at the left of ...
    goose.3 image) and
% goose.4, goose.5 and goose.6  images (at the right of goose.3 image).
read_str3 = strcat('goose/goose.',int2str(3),'.pgm');
img3 = imread(read_str3);
Iw = warp(img3,M1,size(I_mos));
pix_to_update = warp(ones(size(img3)),M1,size(I_mos)) > 0;
I_mos(pix_to_update) = Iw(pix_to_update);

% registering goose.2 image in the current mosaic image 'I_mos'
read_str2 = strcat('goose/goose.',int2str(2),'.pgm');
[M,I_mos] = stitch_new_image(read_str3,read_str2,M1,I_mos);

% registering goose.1 image in the current mosaic image 'I_mos'
read_str1 = strcat('goose/goose.',int2str(1),'.pgm');
[M,I_mos] = stitch_new_image(read_str2,read_str1,M,I_mos);

% registering goose.0 image in the current mosaic image 'I_mos'
read_str0 = strcat('goose/goose.',int2str(0),'.pgm');
[M,I_mos] = stitch_new_image(read_str1,read_str0,M,I_mos);

% registering goose.4 image in the current mosaic image 'I_mos'
read_str4 = strcat('goose/goose.',int2str(4),'.pgm');
[M,I_mos] = stitch_new_image(read_str3,read_str4,M1,I_mos);

% registering goose.5 image in the current mosaic image 'I_mos'
read_str5 = strcat('goose/goose.',int2str(5),'.pgm');
[M,I_mos] = stitch_new_image(read_str4,read_str5,M,I_mos);

% registering goose.6 image in the current mosaic image 'I_mos'
read_str6 = strcat('goose/goose.',int2str(6),'.pgm');
[M,I_mos] = stitch_new_image(read_str5,read_str6,M,I_mos);

imwrite(uint8(I_mos),'mosaic1.jpg');
toc()
```

- Octave Function: (`stitch_new_image.m`)

```
function [M1,I_mos] = stitch_new_image(str1,str2,M,I_mos)
    % This function will stitch the image 'str2' into the paranomic ...
        mosaic
    % 'I_mos' based on the image 'str1'. M is the previous ...
        transformation matrix.

    % Find matching pixels using SIFT. I have slightly modified at ...
        the end of
    % the match function so that rather than displaying matched ...
        pixels (using line connections),
    % i have returned the matched pixels.
    [x1,x2] = match(str1,str2);

    % Find the projective homography
    % Use Ransac to remove outlier mathed pixels
    [H, inliers] = ransacfithomography(x1,x2, 0.005);

    % New transformation matrix = Current projective homography * ...
        previous transformation matrix
    M1 = H*M;

    % Warp the image 'str2' into the paranomic mosaic 'I_mos' using ...
        transformation matrix M1
    img2 = imread(str2);
    Iw = warp(img2,M1,size(I_mos));
    pix_to_update = warp(ones(size(img2)),M1,size(I_mos)) > 0;
    I_mos(pix_to_update) = Iw(pix_to_update);
end
```

- Other Octave Functions and one binary file (exe):
  - `homography2d.m` [1]
  - `normalise2dpts.m` [1]
  - `hnormalise.m` [1]
  - `iscolinear.m` [1]
  - `ransacfithomography.m` [1]
  - `randomsample.m` [1]
  - `ransac.m` [1]
  - `sift.m` [2]
  - `match.m` [2]
  - `siftWin32.exe` [2]
  - `resample.m` (Given with the assignment)
  - `warp.m` (Given with the assignment)

- Input Images:

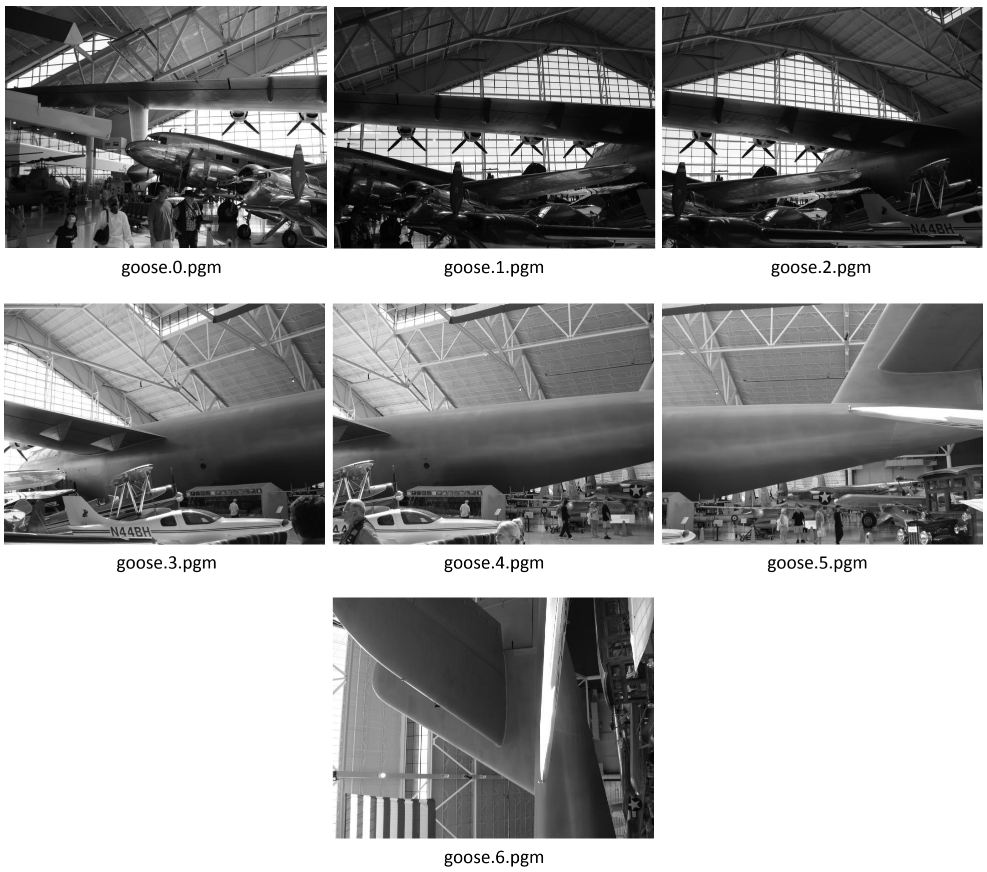

Figure 4: Input images (Spruce Goose).

- Output image (after registration):

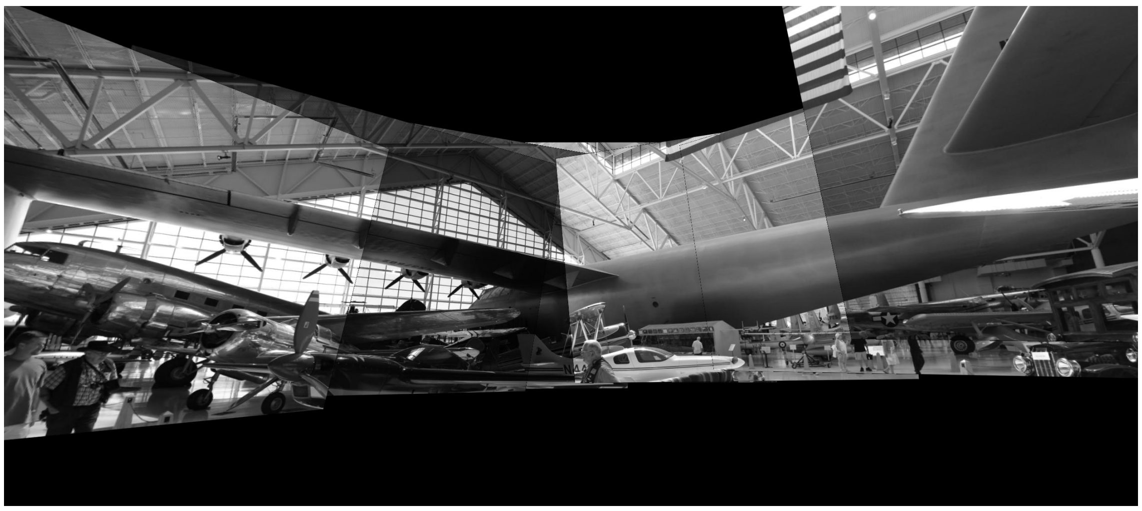

Figure 5: Output image (mosaic1.jpg).

3 Part 03(BONUS): Registration of 07 images (Spruce Goose images) using Cylindrical projection, SIFT and RANSAC

- I got the implementation idea (on cylindrical projection) from reference [3] and [4].
- Main Octave Script: (`main_script3.m`)
  First run this script.

```
clear
pkg load image
tic()

% Create cylinder images of all the given input images.
% I have got the idea from the following online pdf files (class ...
    lectures):
% [3] http://www.csie.ntu.edu.tw/¬cyy/courses/vfx/05spring/ ...
    lectures/handouts/lec06_stitching_4up.pdf
% [4] http://groups.csail.mit.edu/graphics/classes/CompPhoto06/ ...
    html/lecturenotes/15_pano_6.pdf

cylinder_image_create();

% Initialization of panaromic mosaic
I_mos = zeros(750,2300);

% Initial transformation matrix (simple translation)
M1 = [ 1 0 −1000;0 1 −200;0 0 1];

% Now do the same registration process (Part 2) on these ...
    cylindrical images.

% Place the first image at the center of the mosaic. I have started ...
    with goose.3 image.
% Then register goose.2, goose.1 and goose.0 (at the left of ...
    goose.3 image) and
% goose.4, goose.5 and goose.6  images (at the right of goose.3 image).
read_str3 = strcat('cylin_goose/goose.',int2str(3),'.pgm');
img3 = imread(read_str3);
Iw = warp(img3,M1,size(I_mos));
pix_to_update = warp(ones(size(img3)),M1,size(I_mos)) > 0;
I_mos(pix_to_update) = Iw(pix_to_update);

% registering goose.2 image in the current mosaic image 'I_mos'
read_str2 = strcat('cylin_goose/goose.',int2str(2),'.pgm');
[M,I_mos] = stitch_new_image(read_str3,read_str2,M1,I_mos);

% registering goose.1 image in the current mosaic image 'I_mos'
read_str1 = strcat('cylin_goose/goose.',int2str(1),'.pgm');
[M,I_mos] = stitch_new_image(read_str2,read_str1,M,I_mos);

% registering goose.0 image in the current mosaic image 'I_mos'
read_str0 = strcat('cylin_goose/goose.',int2str(0),'.pgm');
[M,I_mos] = stitch_new_image(read_str1,read_str0,M,I_mos);

% registering goose.4 image in the current mosaic image 'I_mos'
read_str4 = strcat('cylin_goose/goose.',int2str(4),'.pgm');
[M,I_mos] = stitch_new_image(read_str3,read_str4,M1,I_mos);

% registering goose.5 image in the current mosaic image 'I_mos'
```

```
    read_str5 = strcat('cylin_goose/goose.',int2str(5),'.pgm');
    [M,I_mos] = stitch_new_image(read_str4,read_str5,M,I_mos);

    % registering goose.6 image in the current mosaic image 'I_mos'
    read_str6 = strcat('cylin_goose/goose.',int2str(6),'.pgm');
    [M,I_mos] = stitch_new_image(read_str5,read_str6,M,I_mos);

    imwrite(uint8(I_mos),'mosaic2.jpg');
    toc()
```

- Octave Function 1: (`cylinder_image_create.m`)

```
    function cylinder_image_create()
        % This function will convert each input images into cylinder ...
            images using cylindrical
        % warping. Write images into the 'cylinder_goose' folder.

        for i = 0 : 6
            str = strcat('goose/goose.',int2str(i),'.pgm');
            I = imread(str);
            out_str = strcat('cylin_goose/goose.',int2str(i),'.pgm');
            imwrite(uint8(cylinder(I,1050)),out_str);
        end
    end
```

- Octave Function 2: (`cylinder.m`)

```
    function out = cylinder(I,f)
        % This function will do the cylindrical warping of image I ...
            considering
        % f as focal length. I have trial with different values of f ...
            from 700 to 1200,
        % and use the value 1050 that gives me the desire output.

        % Here, the equations are taken from reference [3]

        % Initializations
        [ydim xdim] = size(I);
        yc = ydim/2;
        xc = xdim/2;
        out = zeros(size(I));
        xp = yp = xo = yo = [];

        % For x = 1 to xdim find transformed x coordinates ximg
        x = 1:xdim;
        theta = (x - xc)./f;
        X = sin(theta);
        Z = cos(theta);
        xn = X ./ Z;
        ximg = floor(f .* xn + xc);

        % For y = 1 to ydim find transformed y coordinates yimg
        for y = 1 : ydim
            h = (y - yc)./f;
            yn = h ./ Z;
            yimg = floor(f .* yn + yc);

            % Accumulating 1D points to make 2D grid of
            % Original points [xo yo] and transformed points [xp yp]
            xp = [xp,ximg];
            xo = [xo,x];
            yp = [yp,yimg];
            yo = [yo,ones(1,xdim).*y];
        end

        % Removing some transformed points that are out of boundary of ...
            original image

        pos = find(xp < 1 | xp > xdim | yp < 1 | yp > ydim);
        xp(pos) =  yp(pos) = xo(pos) = yo(pos) = [];

        % Placing the pixel values using the mapping ([xp yp] <-> [xo yo])
        out(sub2ind([ydim, xdim],yo,xo)) = I(sub2ind([ydim, xdim],yp,xp));
    end
```

- Other Octave Functions and one binary file (exe):
  - `stitch_new_image.m` (My own function which is also used in Part 2)
  - `homography2d.m` [1]
  - `normalise2dpts.m` [1]
  - `hnormalise.m` [1]
  - `iscolinear.m` [1]
  - `ransacfithomography.m` [1]
  - `randomsample.m` [1]
  - `ransac.m` [1]
  - `sift.m` [2]
  - `match.m` [2]
  - `siftWin32.exe` [2]
  - `resample.m` (Given with the assignment)
  - `warp.m` (Given with the assignment)
- Output image (after registration using cylindrical projection):

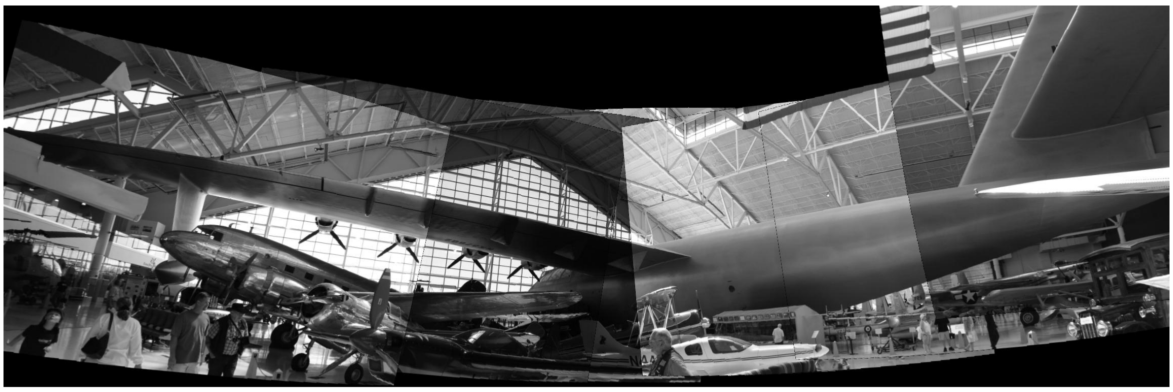

Figure 6: Output image (mosaic2.jpg).

4 My Salient Observations:

- Observation 1: My observation is that one out-lier point produce heavy impact on the output. For example in the part 01 (registration of two book images using hand selected points) if i put one out-lier point (out of 8 points), it will change the output completely. Figure 7 shows an example.

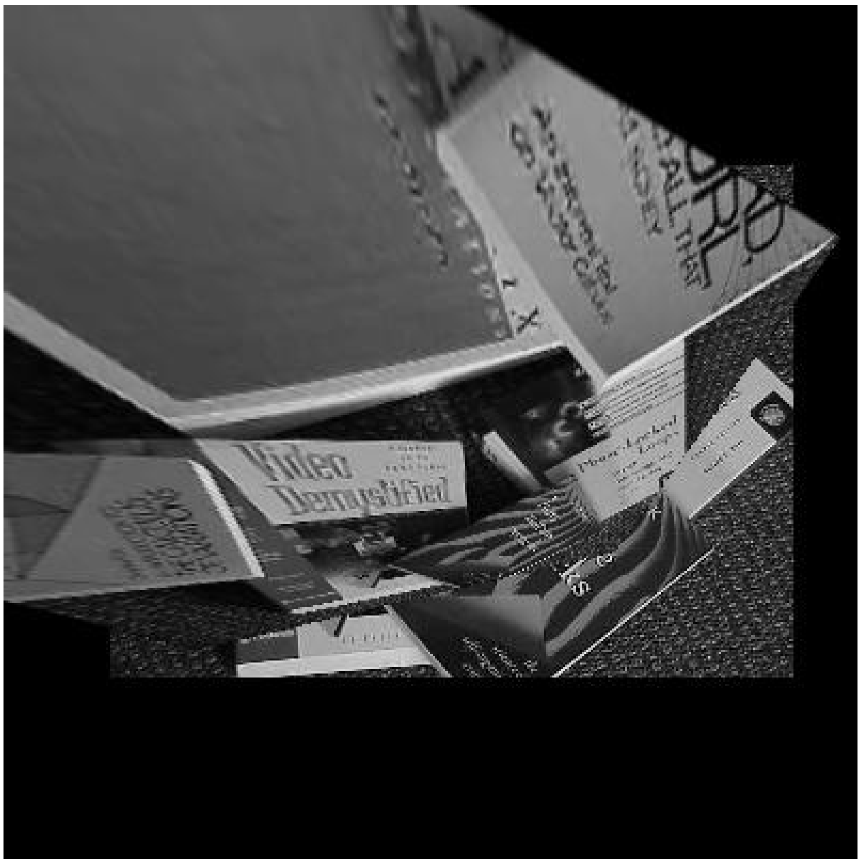

Figure 7: Output image showing the effect of one out-lier point.

- Observation 2: Minimum 04 non-colinear matched points are needed to make the registration of two images. But more points will improve the chance of perfect registration. Figure 8 shows four corner points that produce a good output which is shown in the figure 9. Figure 10 shows four bad corner points (colinear) that produce a bad output which is shown in the figure 11.

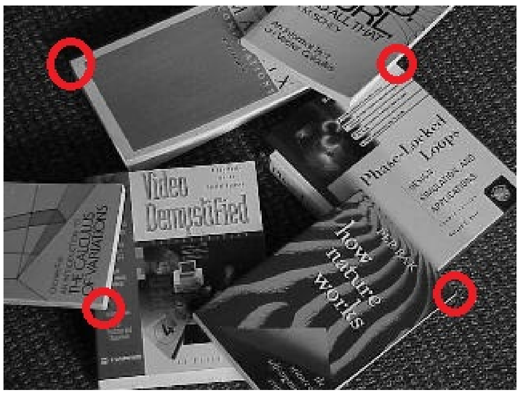

Figure 8: Four hand selected non-colinear points (red circled corner points). These points produce a good output(Figure 9).

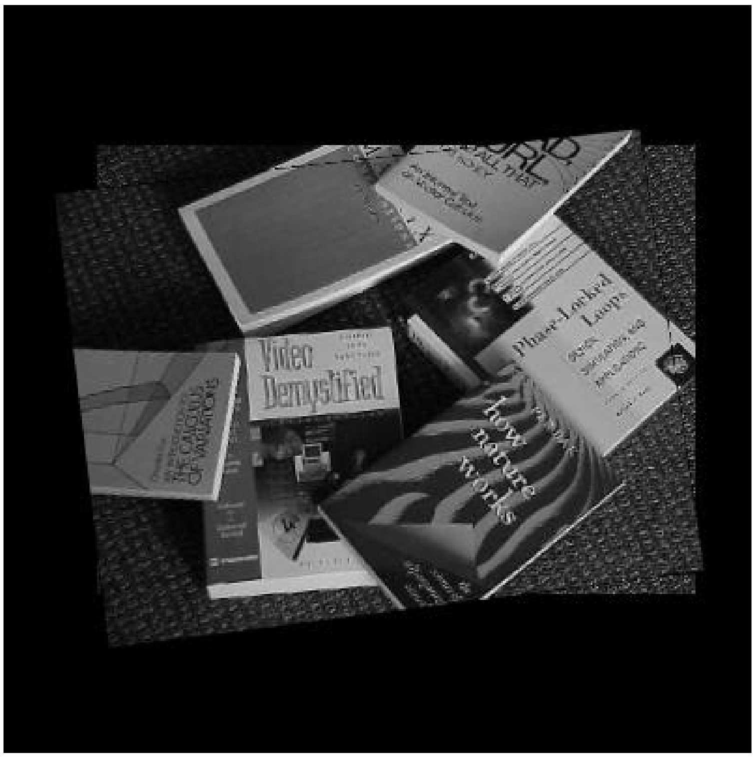

Figure 9: Generated output using only 04 non-colinear points (which are shown in Figure 8).

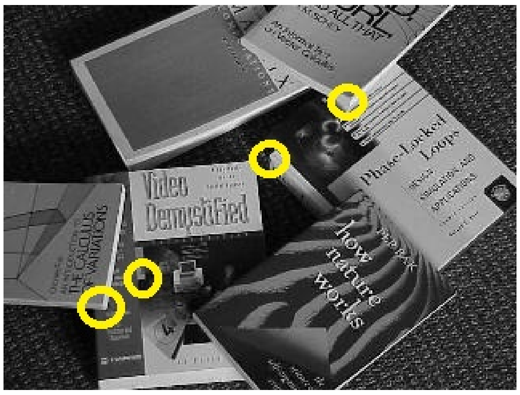

Figure 10: Four hand selected colinear points (yellow circled corner points). These points produce a bad output(Figure 11).

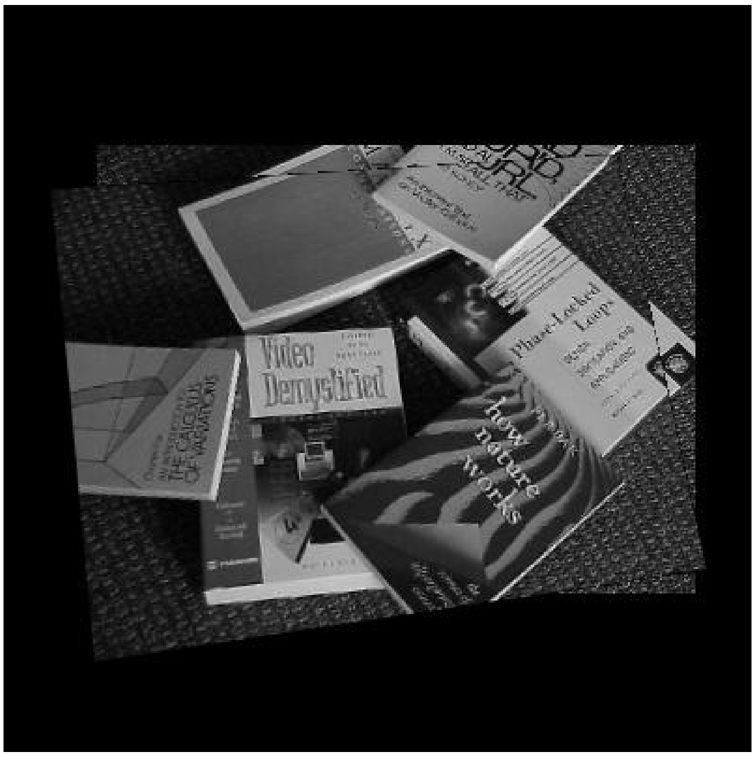

Figure 11: Generated output using only 04 colinear points (which are shown in Figure 10).

- Observation 3: Input images can have different levels of brightness (with each other) which create artifacts (visible image boundary) in the panaroma output image. Figure 5 and 6 have these artifacts clearly.
- Observation 4: The order of image registration have significant impact on the output and also, two images must have some overlapped area with salient points to do the registration. Figure 12 is showing the panaroma output that is created considering Goose images in the sequence goose.0, goose.1, goose.2, goose.3, goose.4, goose.5 and goose.6. In my observation, best order is goose.3, goose.2, goose.1, goose.0, goose.4, goose.5 and goose.6 (Figure 5 is created using this sequence).

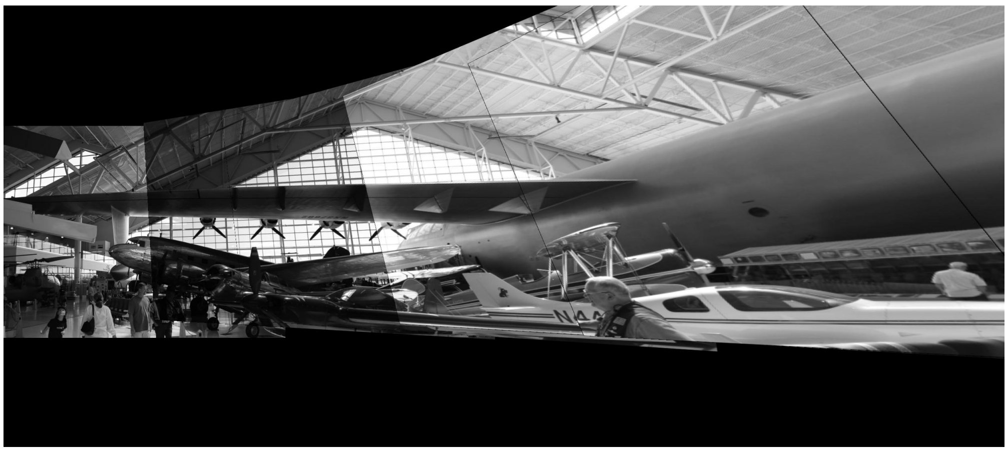

Figure 12: Generated output considering Goose images in the sequence goose.0, goose.1, goose.2, goose.3, goose.4, goose.5 and goose.6.

- Observation 5: Moving objects create artifacts. In the given set (Spruce Goose) of images there are some moving objects (they changes their location over images). Figure 13 showing some of the example artifact regions.

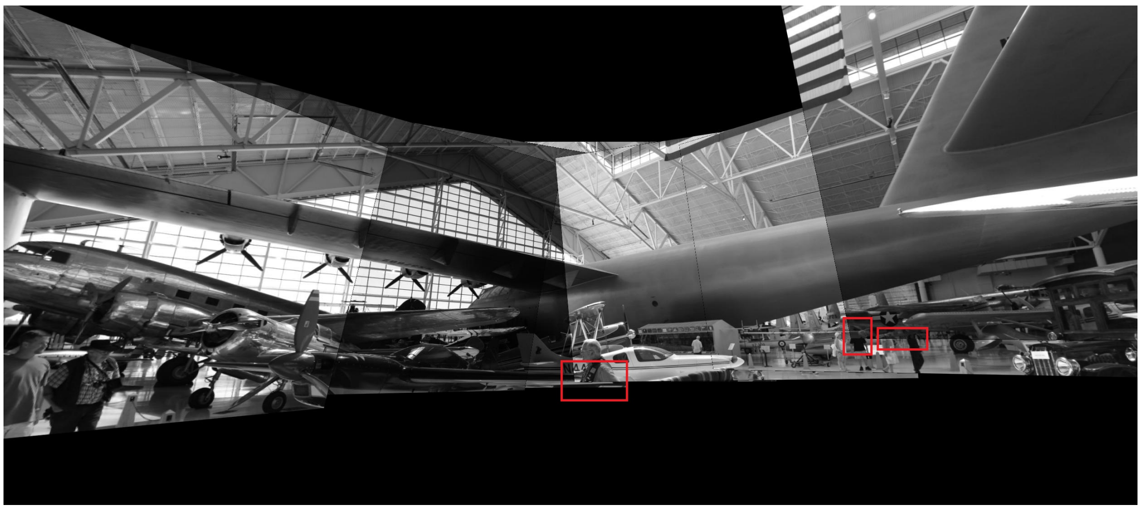

Figure 13: Output panaroma image where the artifacts (due to moving objects) are shown with red rectangle area.

- Observation 6: Due to perspective view (or vanishing points), the left and the right side of the output image are heavily stretched (see figure 5). We can suppress this problem by adding the concept of cylindrical projection (see figure 6).
- Observation 7: Due to transformation and warping into a common coordinate system, the output image contains excess black boundary regions (see figure 5 and 6).

Bibliography

[1] “Matlab and octave functions for cvip.” http://www.peterkovesi.com/matlabfns/index.html. [Online; accessed 10-Nov-2015].

[2] “Demo software: Sift keypoint detector.” http://www.cs.ubc.ca/ lowe/keypoints/. [Online; accessed 12-Nov-2015].

[3] http://www.csie.ntu.edu.tw/c̃yy/courses/vfx/05spring/lectures/handouts/lec06_stitching_4up.pdf. [Online; accessed 14-Nov-2015].

[4] http://groups.csail.mit.edu/graphics/classes/CompPhoto06/html/lecturenotes/15_pano_6.pdf. [Online; accessed 14-Nov-2015].