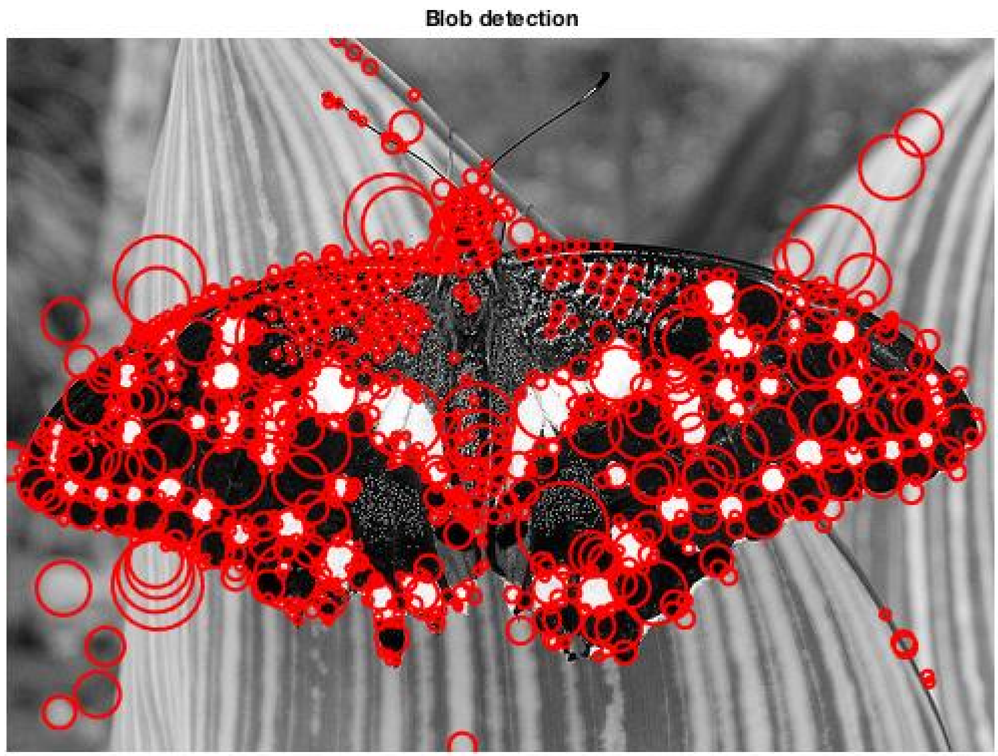
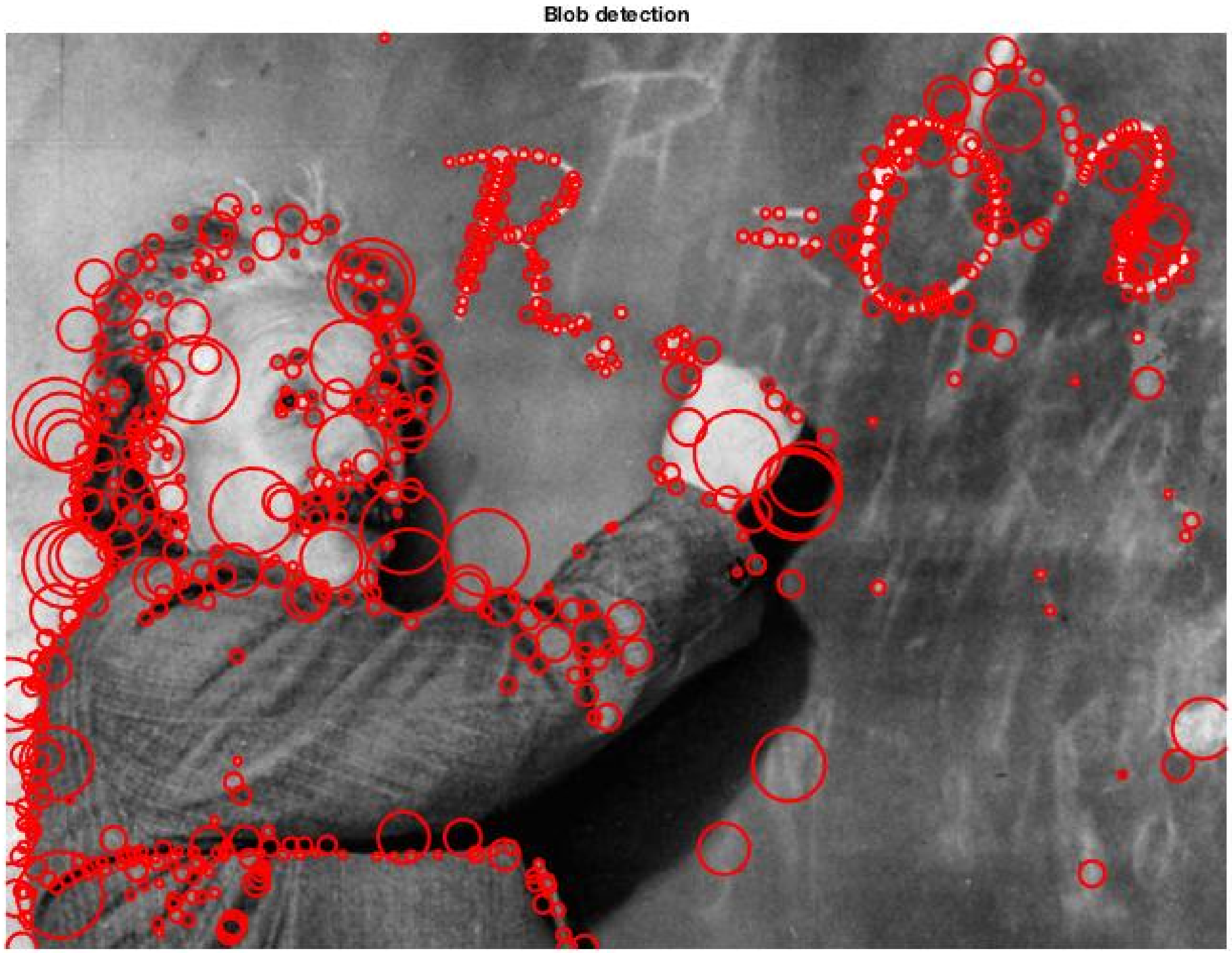
Computer Vision Name:Abhay Doke UID:29552668

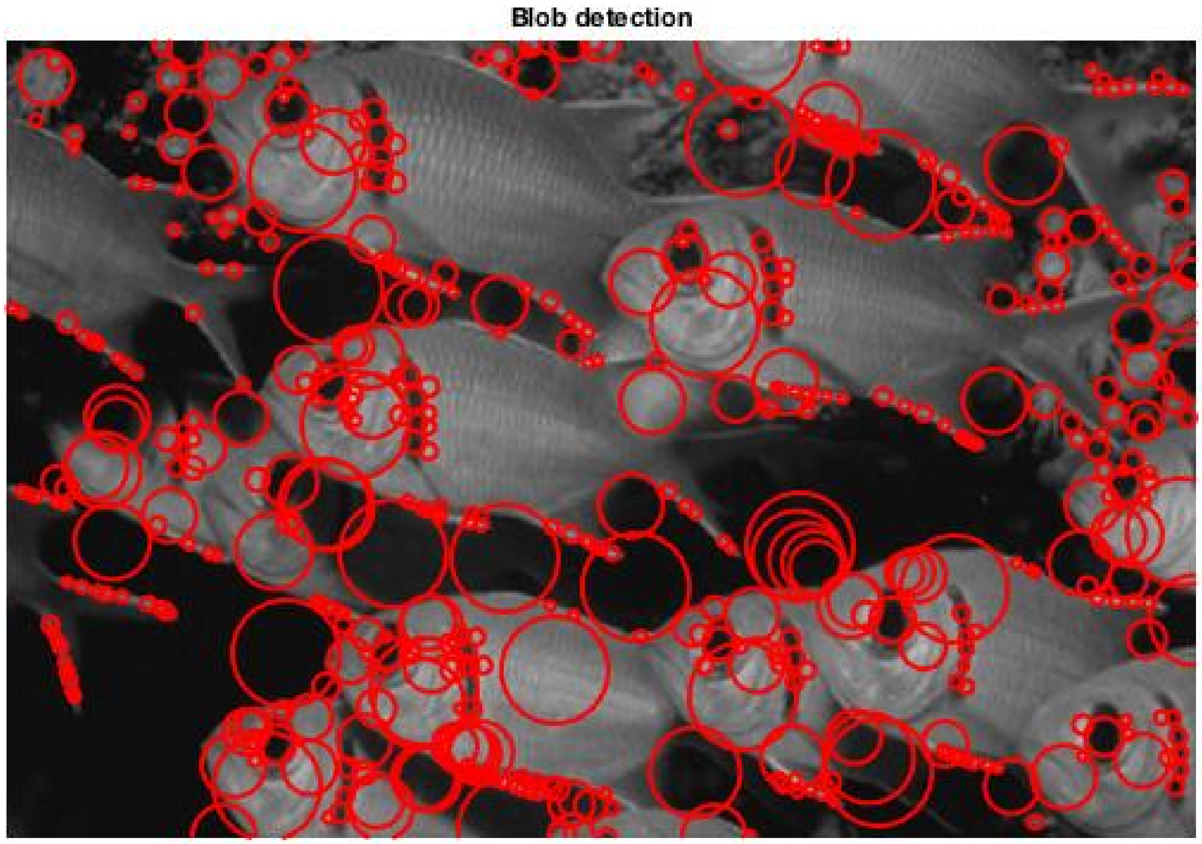
Mini Project 03

1.Scale-space blobs detection

Images Used for hybrid images:


Blob detection


Blob detection



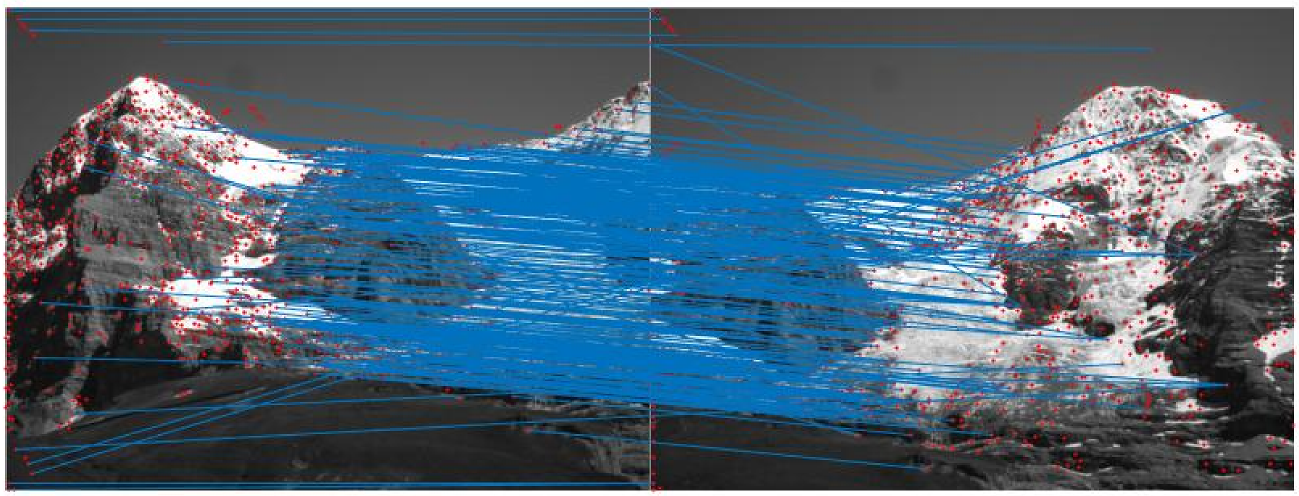
Code for **detectBlobs.m**

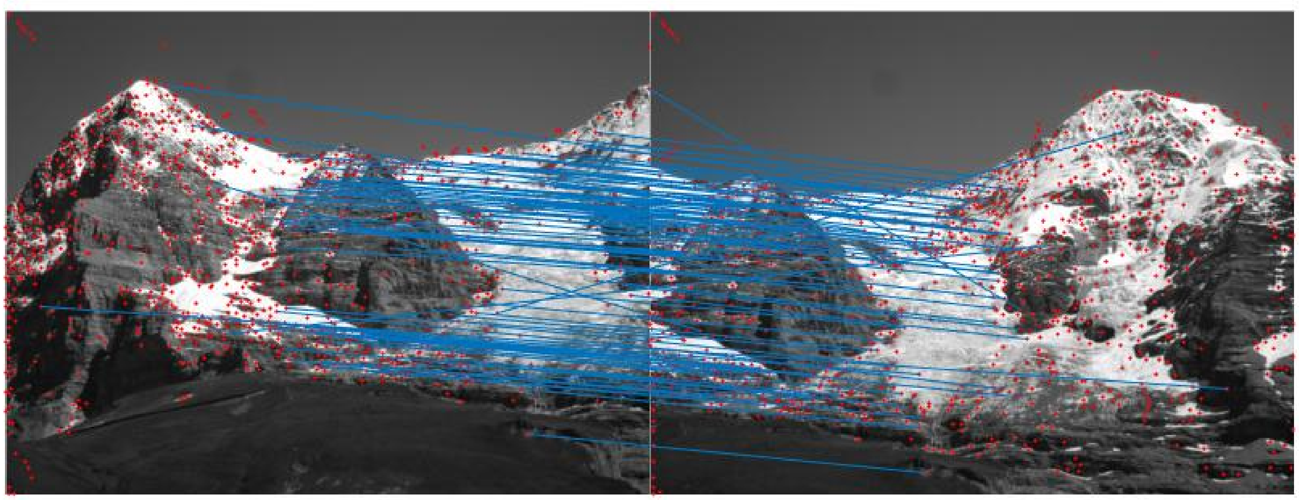
```
function blobs = detectBlobs(im, param)
% This code is part of:
%
%   CMPSCI 670: Computer Vision, Fall 2016
%   University of Massachusetts, Amherst
%   Instructor: Subhransu Maji
%
%   Mini project 3

% Input:
%   IM - input image
%
% Ouput:
%   BLOBS - n x 4 array with blob in each row in (x, y, radius, score)
%
% Dummy - returns a blob at the center of the image
im = im2double(rgb2gray(im));
% sig = 2;
% k = 1.44;
sig = 1.414;
k = sqrt(1.414);
n = 15;
[h,w] = size(im);
pyramid = zeros(h,w,n);


for i = 1:n
    sigi = sig * k^(i-1);
    filt_size = 2*ceil(3*sigi)+1;
    LoG =  sigi^2 * fspecial('log', filt_size, sigi);
    laplacianImage = imfilter(im, LoG, 'same', 'replicate');
    laplacianImage = laplacianImage .^ 2;
    pyramid(:,:,i) = laplacianImage.^2;
end

scale_pyramid = zeros(h, w, n);
for i = 1:n
    scale_pyramid(:,:,i) = ordfilt2(pyramid(:,:,i), 3^2, ones(3));
end
for i = 1:n
    scale_pyramid(:,:,i) = max(scale_pyramid(:,:,max(i-1,1):min(i+1,n)),[],3);
end
scale_pyramid = scale_pyramid .* (scale_pyramid == pyramid);

blobs = zeros(20000,4);
point_counter = 1;
for i = 1:n
    [row,cols] = find(scale_pyramid(:,:,i)>0.0001);
    number_of_points = length(row);
    for j = 1:number_of_points
        blobs(point_counter,1) = cols(j);
        blobs(point_counter,2) = row(j);
```

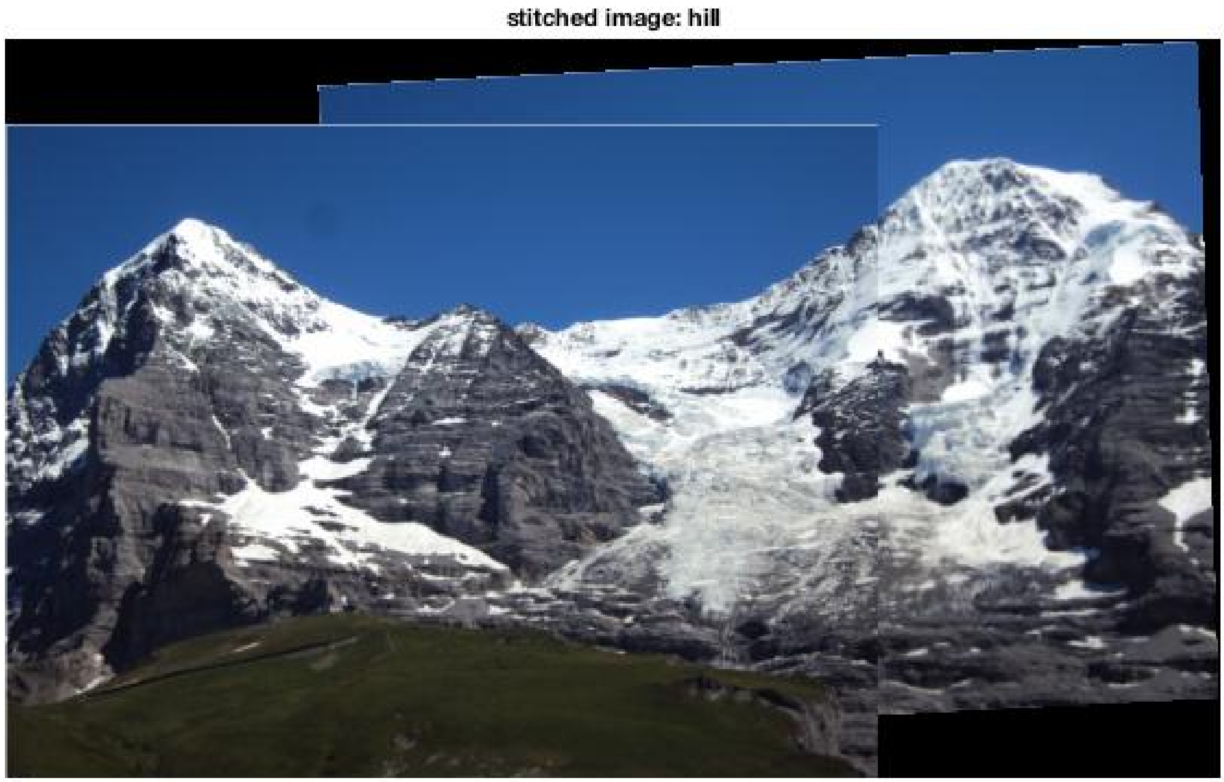
```
        blobs(point_counter,3) = sig * k^(i-1)*sqrt(2);
        blobs(point_counter,4) = scale_pyramid(row(j),cols(j));
        point_counter = point_counter + 1;
    end
end
[values, order] = sort(-blobs(:,3));
blobs = blobs(order,:);
blobs = blobs(1:1000,:);
```

2. Image Stitching

Results for Hill.jpg







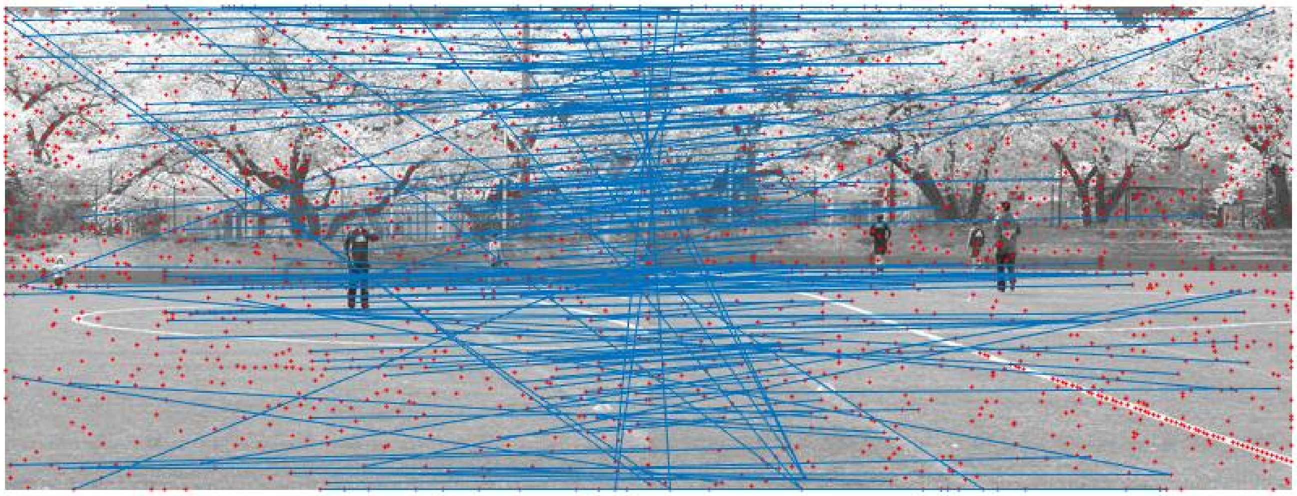
Affine Transformation Matrix:

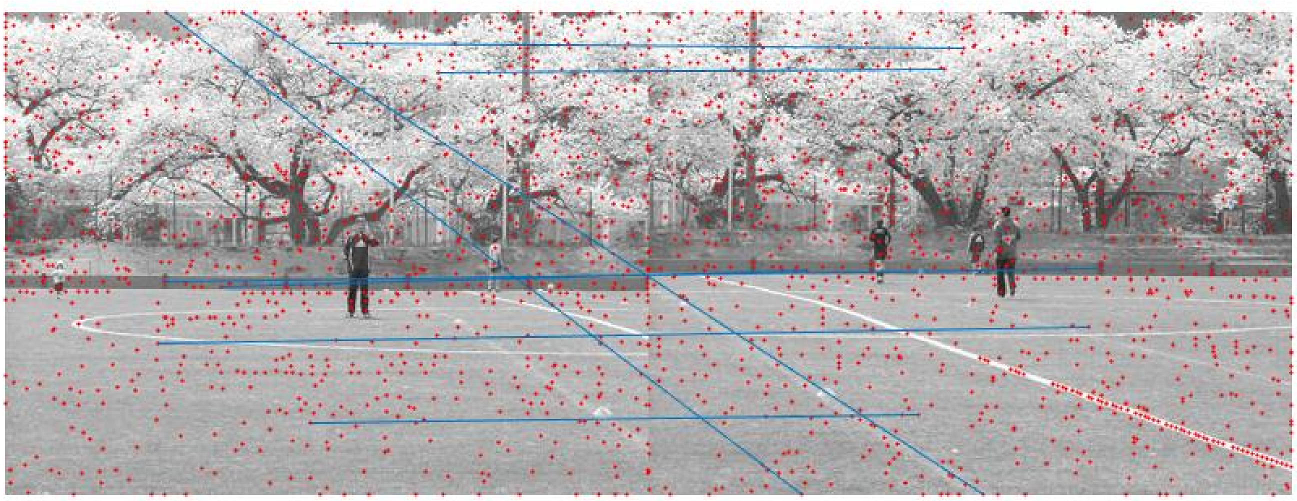
Hill.jpg

1.0102 , 0.0333 , 142.8521;

-0.0513 , 1.0090 , -18.2562;

Results for Field.jpg





stitched image: field

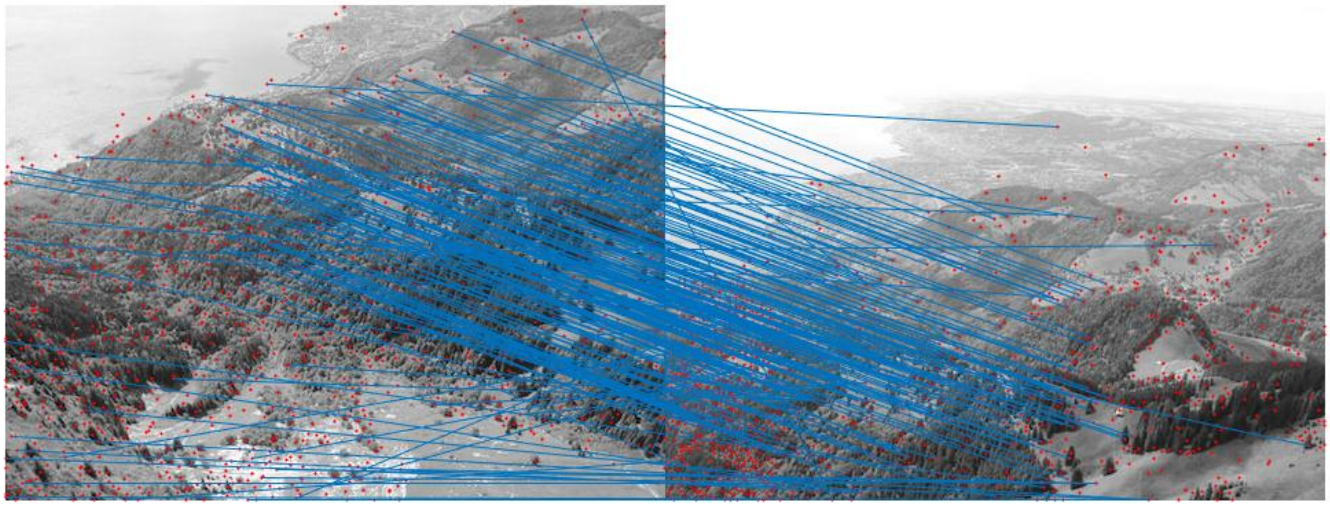
Affine Transformation Matrix:

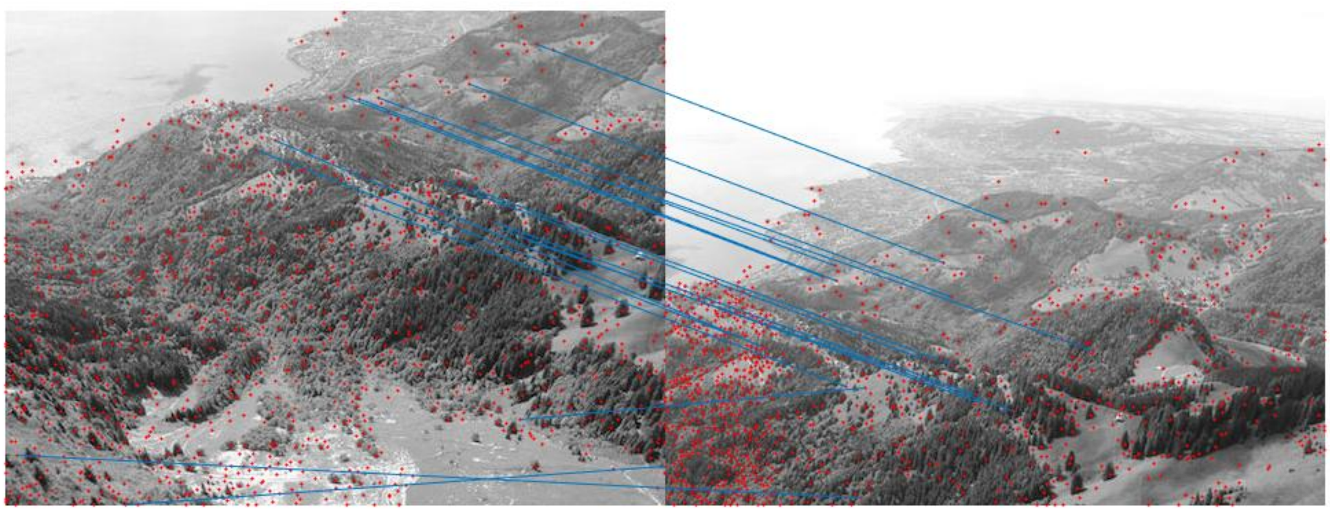
Field.jpg

0.9996 , 0.0112 , 256.6738;

-0.0004 , 1.0112 , 8.6738;

Results for Ledge.jpg





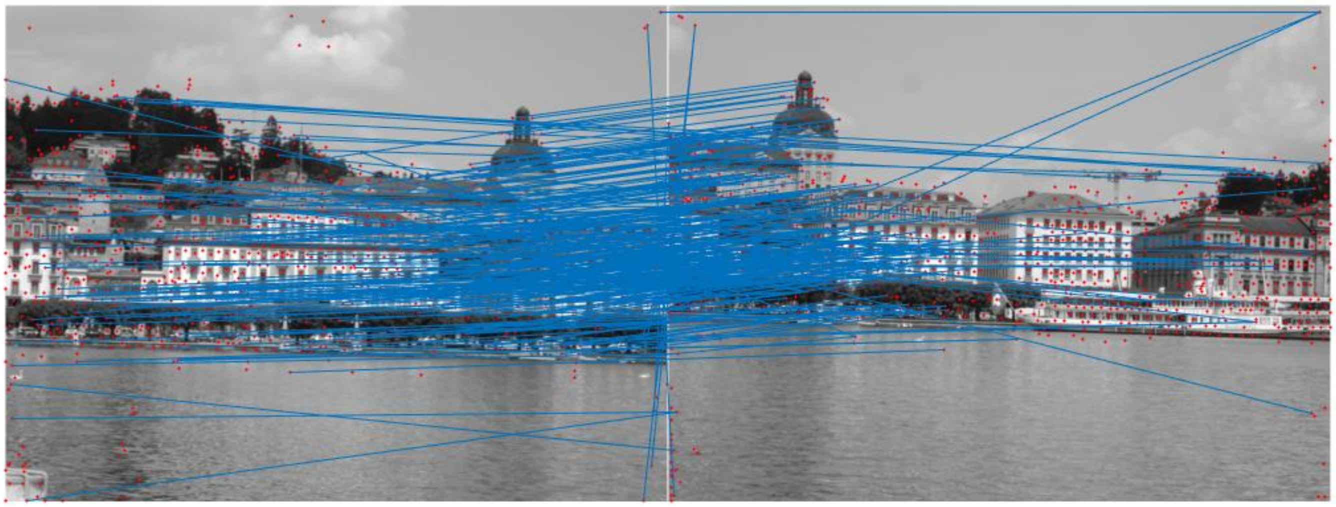
Affine Transformation Matrix:

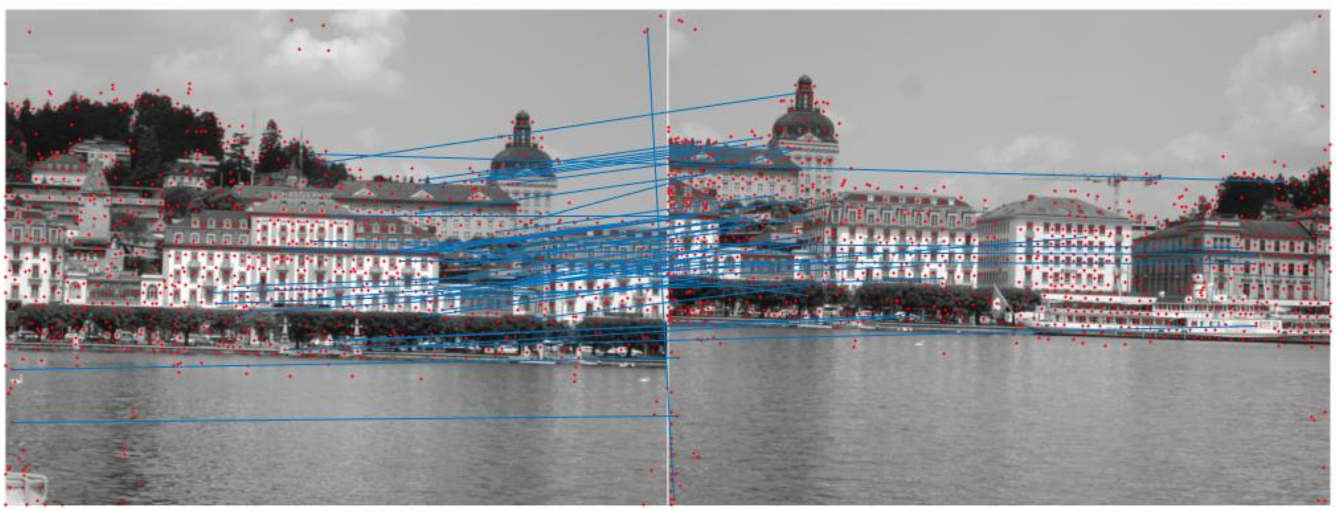
Ledge.jpg

1.0049 , -0.0562 , 143.6702;

0.0679 , 0.9771 , -135.0352;

Results for Pier.jpg





stitched image: pier

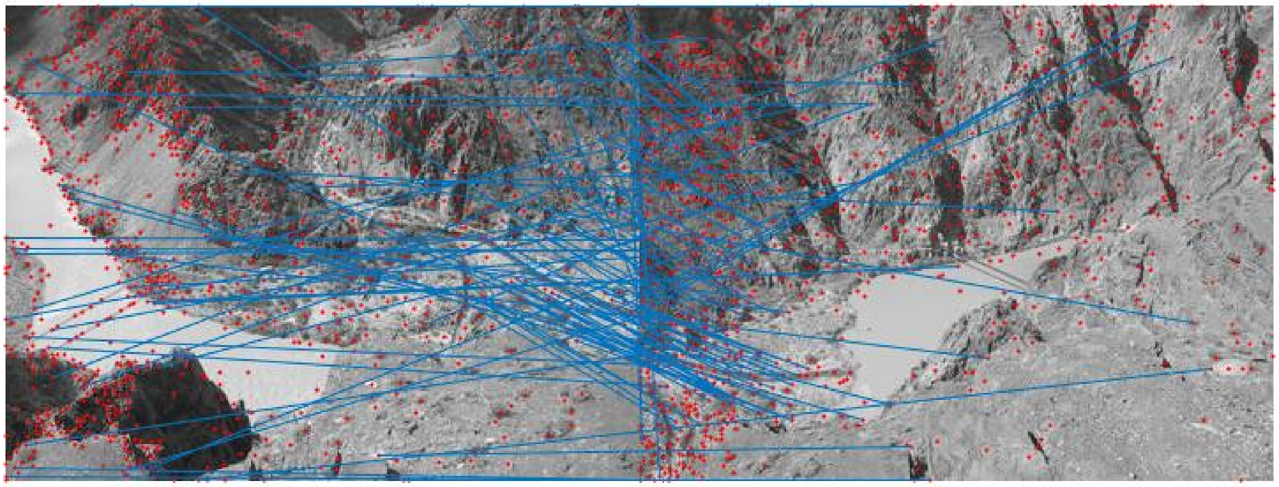
Affine Transformation Matrix:

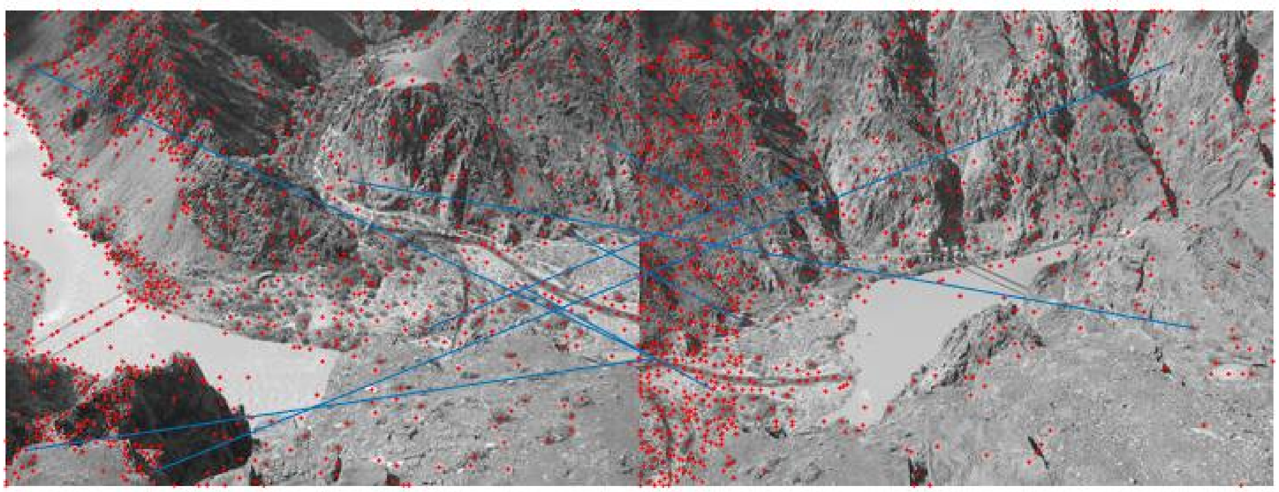
Pier.jpg

1.0078 , 0.0108 , 286.5016;

-0.0078 , 0.9892 , 29.4984

Results for River.jpg





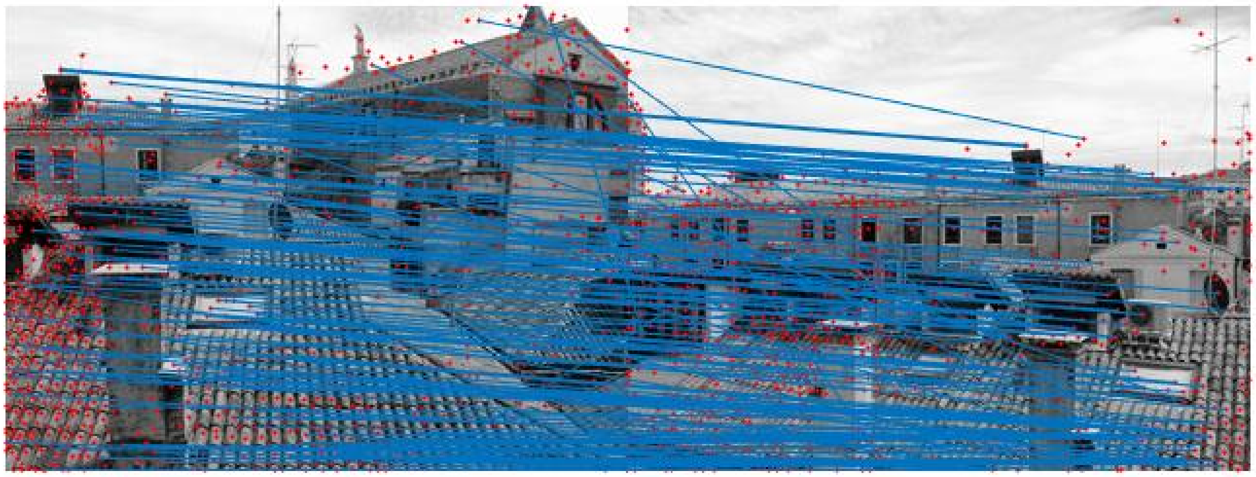
Affine Transformation Matrix:

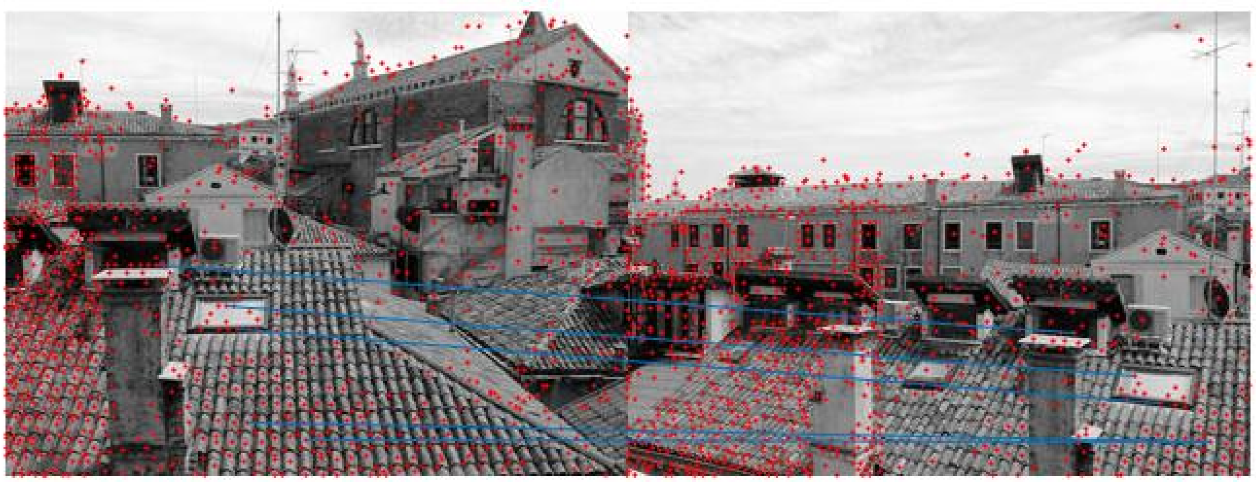
River.jpg

0.9329 , -0.3276 , 321.2789;

0.3539 , 0.9487 , -63.7605;

Results for Roof.jpg





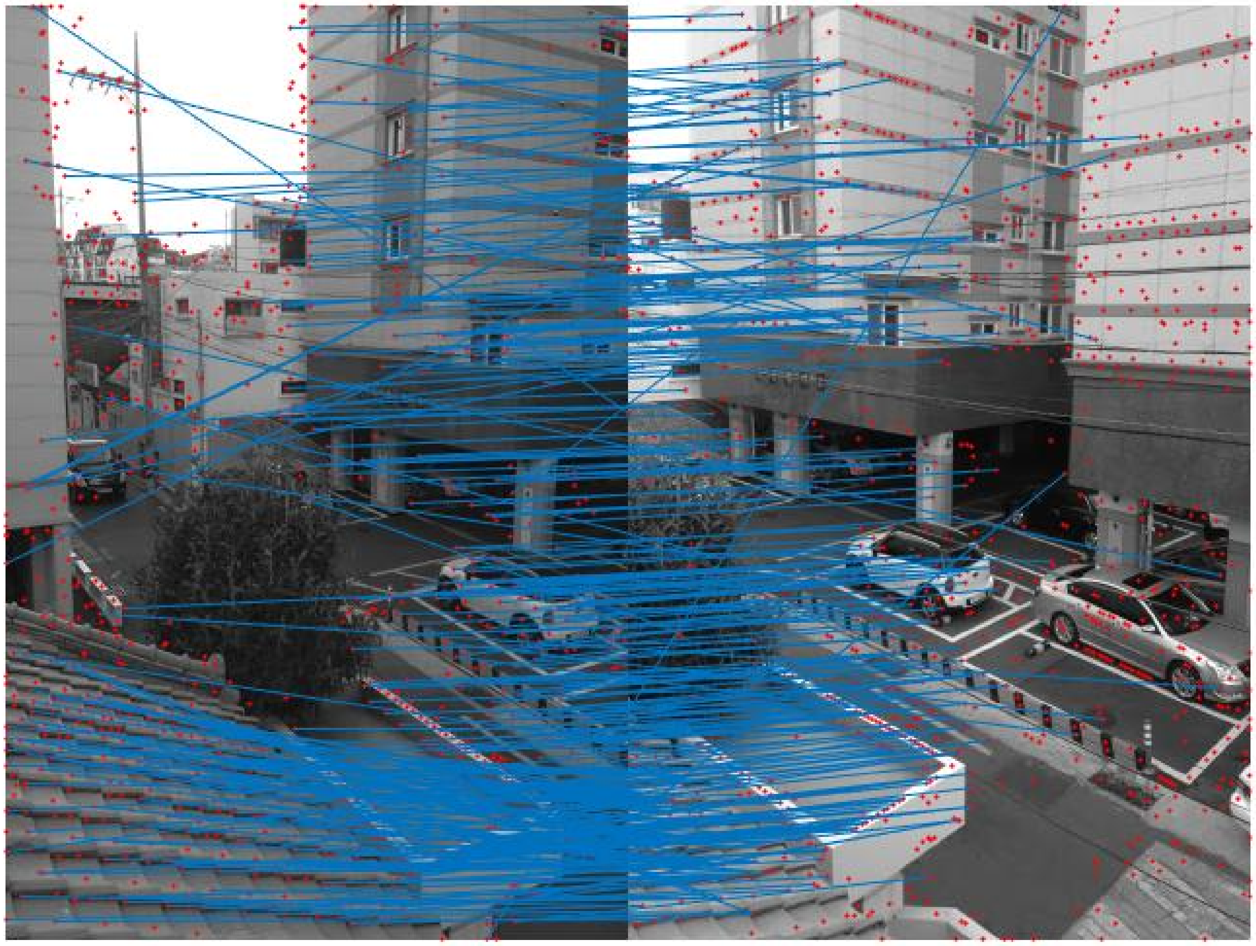
Affine Transformation Matrix:

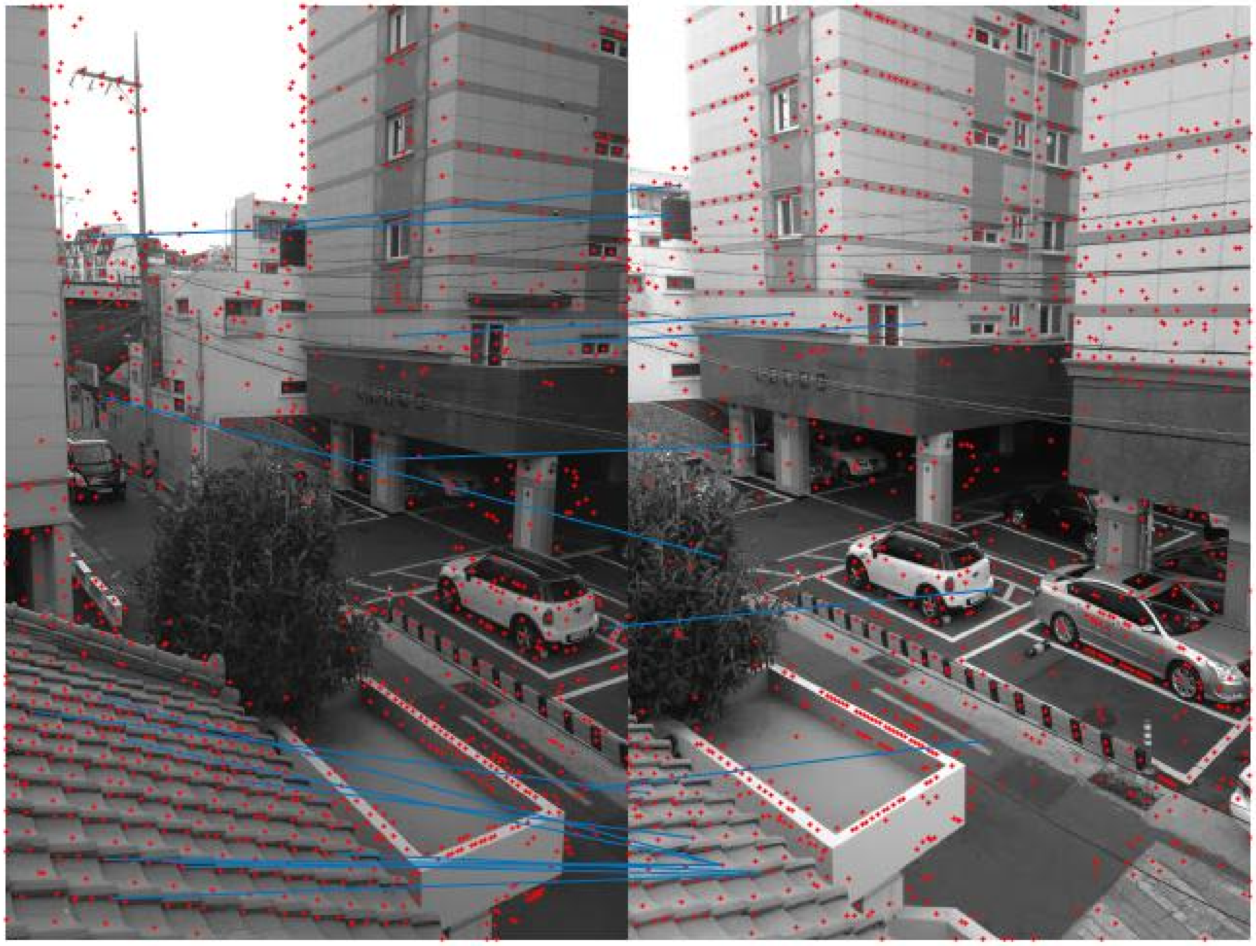
Roof.jpg

0.9693 , 0.1818 , -183.3869;

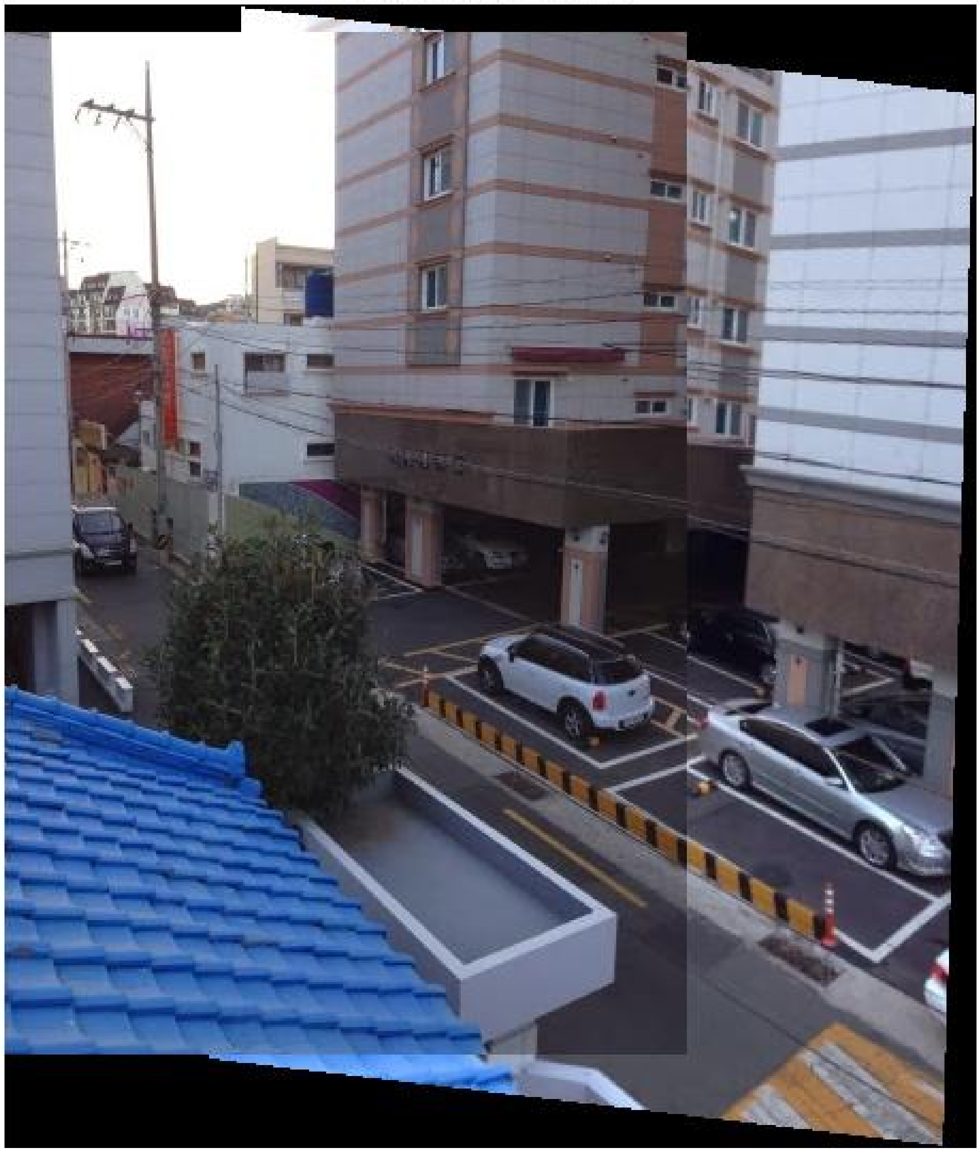
-0.0951 , 1.0119 , -15.1813;

Results for Building.jpg





stitched image: building



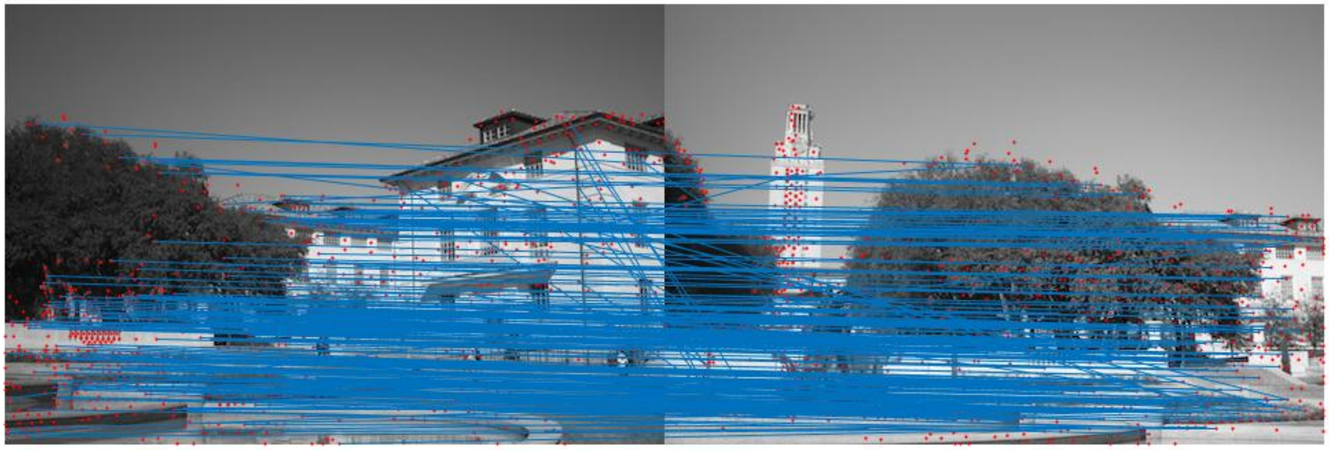
Affine Transformation Matrix:

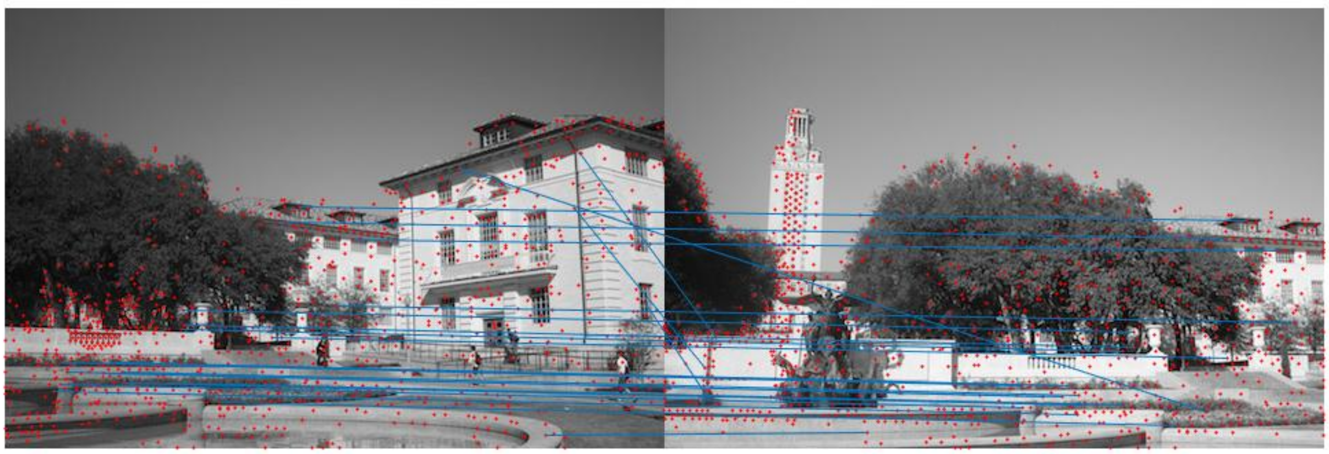
Building.jpg

1.0775 , -0.0308 , 110.6616;

0.1279 , 1.0307 , -12.3888;

Results for Uttower.jpg





Affine Transformation Matrix:

Uttower.jpg

0.9806 , -0.0689 , -190.8962;

0.0202 , 0.9570 , -11.9891;

Code for **computeMatches.m**

```
function m = computeMatches(f1,f2)
% This code is part of:
%
%   CMPSCI 670: Computer Vision, Fall 2016
%   University of Massachusetts, Amherst
%   Instructor: Subhransu Maji
%
%   Mini project 3

[N,d] = size(f1);
[M,d] = size(f2);
m = zeros(N,1);
for i = 1:N
    [min_values,I] = sort(sum((f2-f1(i,:)).^2,2),'ascend');
    ratio = min_values(1)/min_values(2);
    if ratio<0.8
        m(i) = I(1);
    else
        m(i) = 0;
    end
end
```

Code for **ransac.m**

```
function [inliers, transf] = ransac(matches, c1, c2, method)
% This code is part of:
%
%   CMPSCI 670: Computer Vision, Fall 2016
%   University of Massachusetts, Amherst
%   Instructor: Subhransu Maji
%
%   Mini project 3

[I] = find(matches>0);
matches_new = matches(I);
c1_new = c1(I,:);
c2_new = c2(matches_new,:);

max_inliers_count = 0;
best_transformation = zeros(2,3);
for iterations = 1:1000
    n = 3;
    msize = numel(matches_new);
    random_indices = randperm(msize, n);

% ******************************************************** %
    X = ones(3,n);
    col_counter = 1;
    for i = 1:n
        X(1,col_counter) = c2(matches_new(random_indices(col_counter)),1);
        X(2,col_counter) = c2(matches_new(random_indices(col_counter)),2);
        col_counter = col_counter + 1;
    end


    x_prime = zeros(2,n);
    col_counter = 1;
    for i = 1:n
        x_prime(1,col_counter) = c1_new(random_indices(col_counter),1);
        x_prime(2,col_counter) = c1_new(random_indices(col_counter),2);
        col_counter = col_counter + 1;
    end


    affine_transformation = x_prime / X;

    affine_transformation = affine_transformation;

    m = affine_transformation(1:2,1:2);
    t = affine_transformation(1:2,3);



    transformed_c2 = (m*c2(matches_new,1:2)'+t)';
    original_c2 = c1_new(:,1:2);
```

```
    points_count = size(original_c2,1);
    inliers_count = 0;
    for i = 1:points_count
        distance = sqrt((transformed_c2(i,:) - original_c2(i,:)).^2);
        if distance < 2
            inliers_count = inliers_count+1;
        end
    end
    if inliers_count > max_inliers_count
        max_inliers_count = inliers_count;
        best_transformation = affine_transformation;
    end
end

disp(max_inliers_count);
disp(best_transformation);
transf = best_transformation;
n = size(matches,1);
inliers = zeros(max_inliers_count,1);
inlier_count = 1;


m = best_transformation(1:2,1:2);
t = best_transformation(1:2,3);
for i = 1:n
    if matches(i)>0
        %transformed_c2 = (m * c1(i,1:2)' + t)';
        transformed_c2 = (m * c2(matches(i),1:2)' + t)';
        original_c2 = c1(i,1:2);
        distance = sqrt((transformed_c2 - original_c2).^2);
        if distance < 2
            inliers(inlier_count,1) = matches(i);
            inlier_count = inlier_count + 1;
        end
    end

end

transf = best_transformation;
```

All Afine Transformation Matrices Hill.jpg

1.0102 , 0.0333 , 142.8521;

-0.0513 , 1.0090 , -18.2562

Field.jpg

0.9996 , 0.0112 , 256.6738;

-0.0004 , 1.0112 , 8.6738;

Ledge.jpg

1.0049 , -0.0562 , 143.6702;

0.0679 , 0.9771 , -135.0352;

Pier.jpg

1.0078 , 0.0108 , 286.5016;

-0.0078 , 0.9892 , 29.4984

River.jpg

0.9329 , -0.3276 , 321.2789;

0.3539 , 0.9487 , -63.7605;

Roof.jpg

0.9693 , 0.1818 , -183.3869;

-0.0951 , 1.0119 , -15.1813;

Building.jpg

1.0775 , -0.0308 , 110.6616;

0.1279 , 1.0307 , -12.3888;

Uttower.jpg

0.9806 , -0.0689 , -190.8962;

0.0202 , 0.9570 , -11.9891;