CSC 47100 Computer Vision

Fall 2017

Professor Zhigang Zhu

Topic: NYC Then And Now

Group Members: Shirong Zheng & Jin Hui Chen

December 15, 2017

Table of Contents

1. Introduction

Why this need to be done?

The first problem with two cameras shooting the same object in different positions. If there are overlapping parts of the two pictures, we have reason to believe there is a correspondence between the two pictures. The task in this section is to describe the correspondence between them, the tool we used is the epipolar geometry, which is an important mathematical method to study stereo vision.

To find the correspondence between two images, the most direct way is to match point by point. If you apply some constraints to the epipolar constraint, the scope of the search can be greatly reduced.

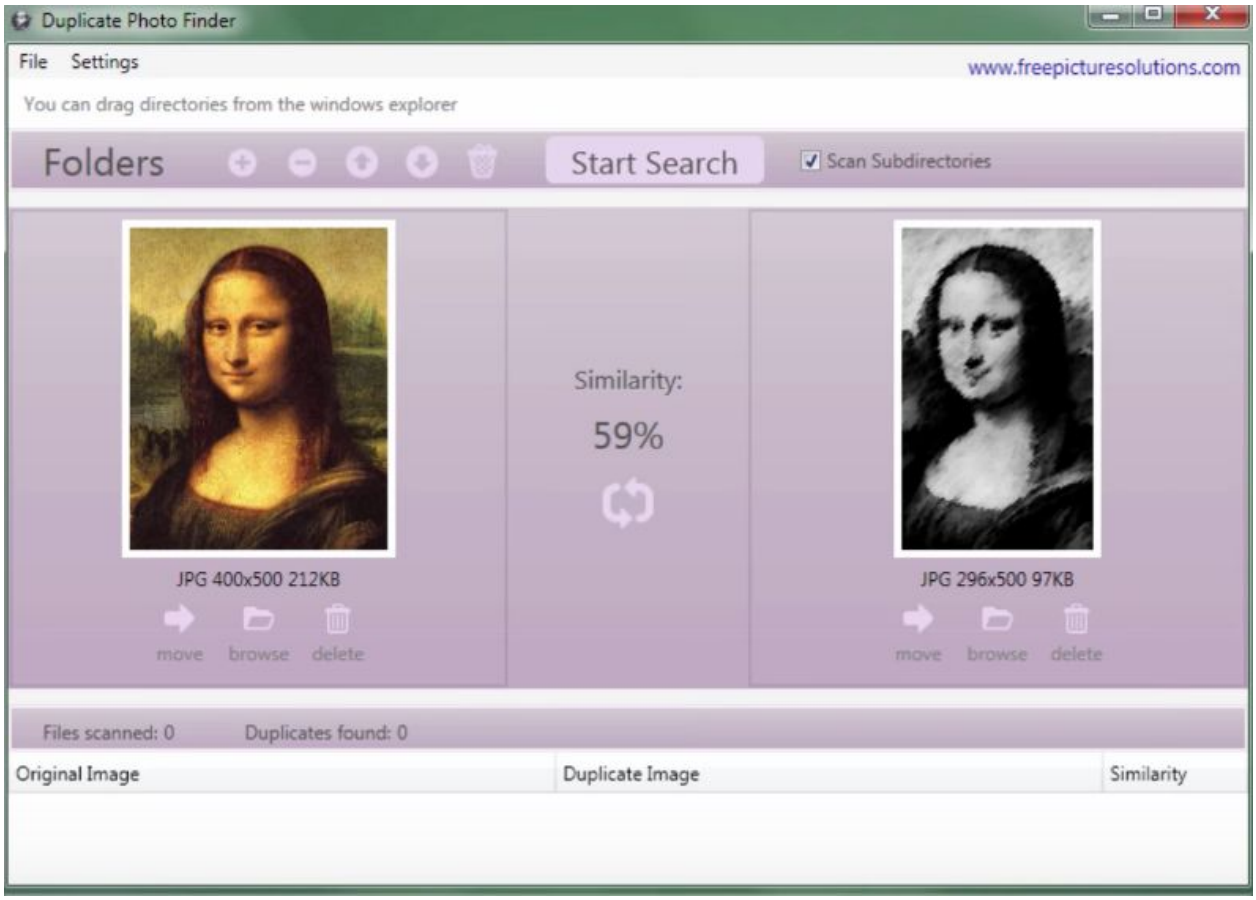
Our job is going to compare two images that were taken in two different time period. Then we will use harris corner to find out corresponding points between two images. After that, they will be tested by epipolar geometry.

1942

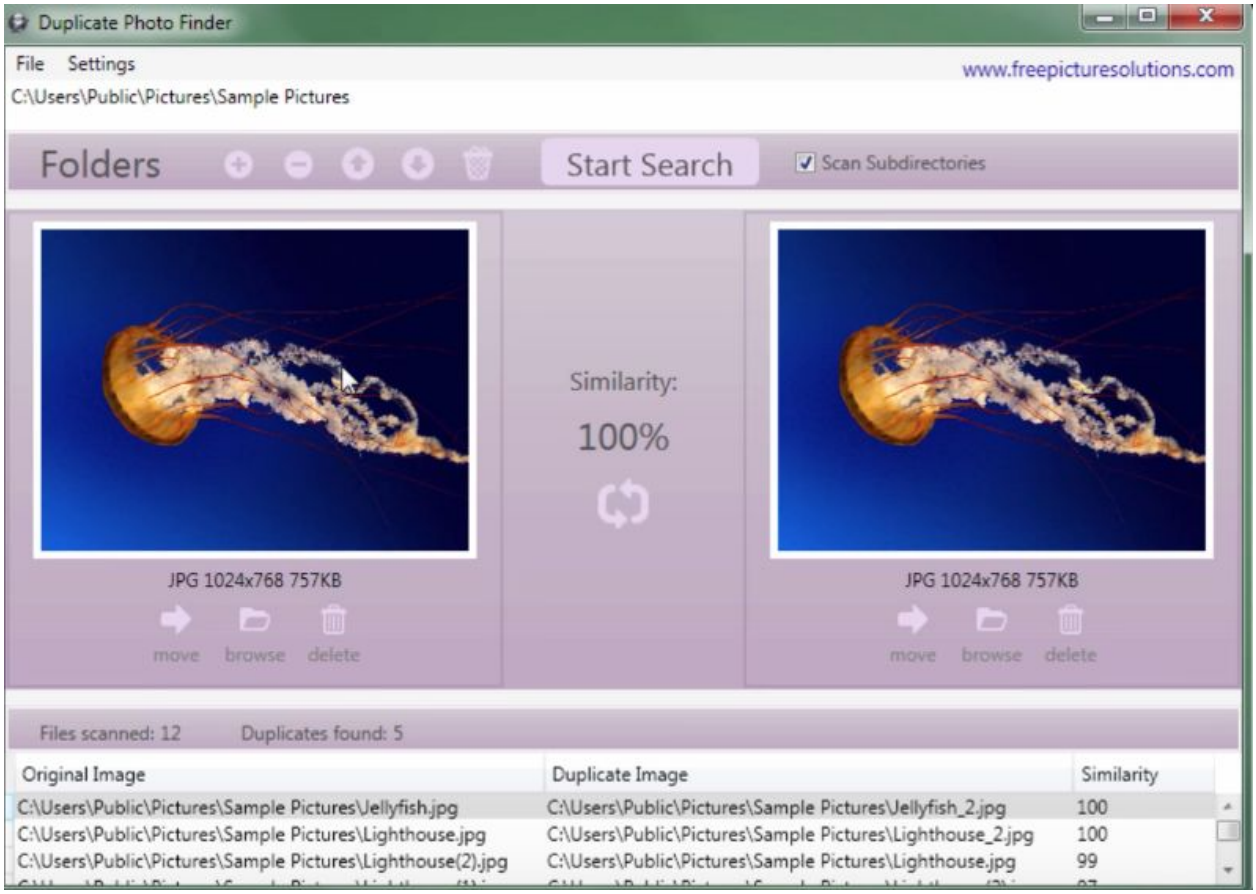
2010

Real World Application

There are some software applications appear in the market. They used
Software: **Duplicate Photo Finder**



The right side image is blurred and indistinct, then this software is hard to find out the the corresponding points.



Both of these two image are clearly to see. Every point is exactly same and match to each other.

2. Related Work

What has been done?

- Work that proposes a different method to solve the same problem.

We thought to use the SIFT detector to customizable and features a decomposition of the algorithm in several reusable M and MEX files. This implementation produces interest points and descriptors which are very similar to David Lowe's implementation.

- Work that uses the same proposed method to solve a different problem.

We had used the same proposed method to solve a different problem in general homework question (HW4).

- A method that is similar to your method that solves a relatively similar problem.

The similar method is SIFT which find out the matched points. The test method is affine transformation which is a linear mapping method that preserves points, straight lines, and planes.

- A discussion of a set of related problems that covers your problem domain.

The related problems could help us deepth and easy understand the topic, then we know which method is best to find the solution of this problem domain.

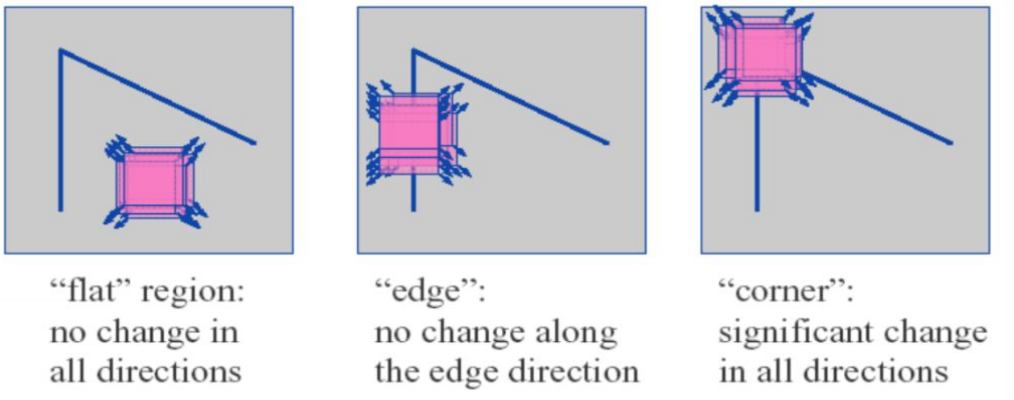
3. Approach

Algorithms

1. Harris Corner approach to detect corner in the image
2. Using Eight-Point Algorithm to solve a system of homogeneous linear equations
3. Normalized 8-Point Algorithm
4. Computer SVD and F matrix
5. Find the epipolar line which equation is F * Homogeneous

Harris Corner

Harris Corner Detector basically finds the difference in intensity for a displacement of (u, v) in all directions.



The figure above clearly reveals that if the pink region moves in a flat space, there are no change in all directions; if the pink region is on the edge, then there is no change along the edge direction; if the pink region is on a corner, then all directions will have huge changes. Let's how we find the corners.

The equation is expressed below:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

- E is the difference between the original and the moved window.
- w(x, y) is the window at position (x, y).
- I is the intensity of the image at a position (x, y)
- u is the window's displacement in the x direction
- v is the window's displacement in the y direction
- I(x+u, y+v) is the intensity of the moved window
- I(x, y) is the intensity of the original

After we took steps to rewrite the equation, we have:

$$\sum [I(x+u,y+v)-I(x,y)]^2$$

$$\approx \sum [I(x,y)+uI_x+vI_y-I(x,y)]^2 \quad \text{First order approx}$$

$$= \sum u^2I_x^2+2uvI_xI_y+v^2I_y^2$$

$$= \sum \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \quad \text{Rewrite as matrix equation}$$

$$= \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Then we can rewrite the original equation to:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Where

$$M = \sum w(x,y) \begin{bmatrix} {I_x}^2 & I_xI_y \\ I_xI_y & {I_y}^2 \end{bmatrix}$$

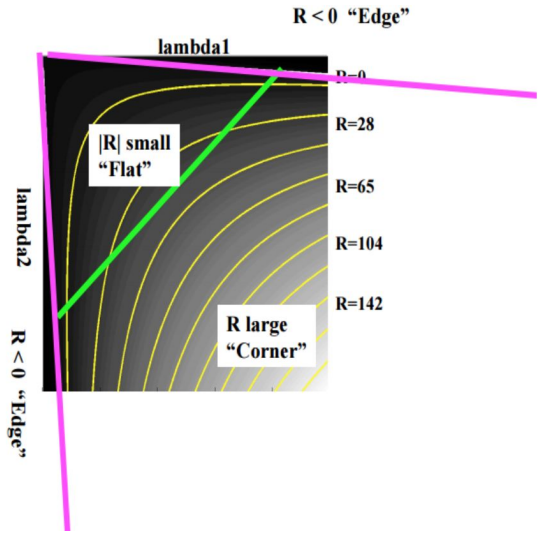
We should have an equation of the Harris Corner Response Measure:

$$R = det(M) - k(trace(M))^2$$

where

- $det(M) = \lambda_1 \lambda_2$
- $trace(M) = \lambda_1 + \lambda_2$
- λ_1 and λ_2 are the eigen values of M

Then we use a standard to measure whether the region is a harris corner or not:



|R| is small, the region is flat
R<0, the region is edge
R is large, the region is corner

Here is the summary of the steps we applied to find the Harris Corner:

1. Compute the horizontal and vertical derivatives of the image. We convolve with derivative of Gaussian
2. Compute the entries of matrix M
3. Compute the Response Measure R
4. Compute local maxima above some threshold

After we found out the harris corners in the images, we extracted the surrounding features of the corners we found. Then we did the matching.

Computation of the Epipolar Lines

A point in the first camera is located on a special line called the epipolar line on the second camera. It works both ways from one camera to the other.

The equation of epipolar line is given by:

$$l_1 = \mathcal{F}p' \tag{2}$$

The equation of epipolar line in the other image is given by:

$$l_2 = p^T\mathcal{F} \tag{3}$$

We can then using the information contained int the result, we obtain in homogeneous coordinates, we can extract the lines equation as:

$$au + bv + p = 0 \tag{4}$$

and then we get,

$$v = -\frac{a}{b}u - \frac{p}{b} \tag{5}$$

Once we have the epipolar lines associated with both the images, we can plot the results in the images. Same has been shown in the figure 3.

The epipoles are the null spaces of the matrix $\mathcal{F}$, and can be determined using:

$$e_1\mathcal{F} = 0 \tag{6}$$

$$\mathcal{F}e_2 = 0 \tag{7}$$

To do so, we can rely on the computation of the normalized eigenvector associated to the smallest eigenvalue fo the matrix $\mathcal{F}$'$\mathcal{F}$.

Eight-Point Algorithm

Find out the essential matrix or the fundamental matrix related to a stereo camera pair from a set of corresponding image points.

Normalized 8-Point Algorithm

The fundamental matrix general function: $\mathbf{x}'^T F\mathbf{x} = \mathbf{0}$

$\mathbf{x} \leftrightarrow \mathbf{x}'$ is any pair of matching points of two images

Since each set of points match provides a linear equation for calculating the F-factor. That the equation can calculate the unknown F:

The coordinates: $\mathbf{x} = (x, y, 1)^T$, $\mathbf{x}' = (x', y', 1)^T$

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = 0$$

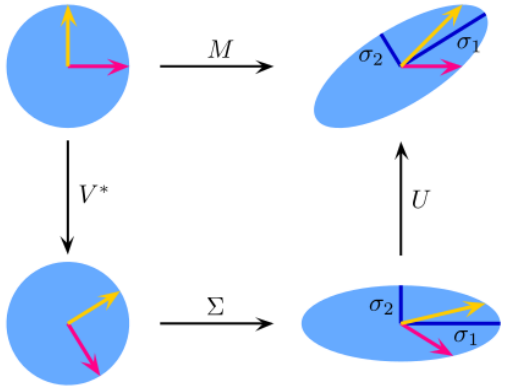
$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

$$\begin{bmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{bmatrix} \mathbf{f} = 0$$

$$A\mathbf{f} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathbf{f} = \mathbf{0}$$

If there is a non-zero solution, the rank of the coefficient matrix A is at most 8.

But the rank of A could be equal to 9, then we use SVD to find the solution $A = UDV^T$



$$M = U \cdot \Sigma \cdot V^*$$

The calculated epipoles for the respective images are:

$$e_1 = \begin{bmatrix} 1845.38 & 1613.01 \end{bmatrix}^T \tag{9}$$

and the epipoles for the respective images are:

$$e_2 = \begin{bmatrix} 2204.82 & 1440.37 \end{bmatrix}^T \tag{10}$$

The two estimated values make sense because we can see these lines intersecting at a point. If we try to read the results from the image intself where are the epipolar lines are intersecting, then we can clearly see that the oberserved pixels have almost the same values as the calculated values.
The observed values are:

$$e_1 = \begin{bmatrix} 1845.5 & 1611.5 \end{bmatrix}^T \tag{11}$$

$$e_2 = \begin{bmatrix} 2205.15 & 1441.41 \end{bmatrix}^T \tag{12}$$

And the corresponding error for e_1 is :

$$error_1 = \begin{bmatrix} -0.12 & 1.51 \end{bmatrix}^T \tag{13}$$

and for e_2 is:

$$error_2 = \begin{bmatrix} -0.33 & -1.03 \end{bmatrix}^T \tag{14}$$

Fundamental Matrix

$$\begin{bmatrix} -6.4189433e\text{-}07 & -9.6923519e\text{-}07 & -6.1682706e\text{-}03 \\ 1.8445219e\text{-}06 & 4.8460931e\text{-}06 & 1.6820901e\text{-}02 \\ -8.3442767e\text{-}04 & -2.1819338e\text{-}02 & 9.9960104e\text{-}01 \end{bmatrix}$$

4. Implementation And Analysis

Harris Corner

A corner in Harris corner detection is defined as "the highest value pixel in a region" (usually `3X3`or `5x5`) so your comment about no point reaching a "threshold" seems strange to me. Just collect all pixels that have a higher value than all other pixels in the `5x5` neighborhood around them.

Apart from that: I'm not 100% sure, but I think you should have:

`K(j,i) = det(H) - lambda*(trace(H)^2)` Where lambda is a positive constant that works in your case (and Harris suggested value is 0.04).

In general the only sensible moment to filter your input is before this point:

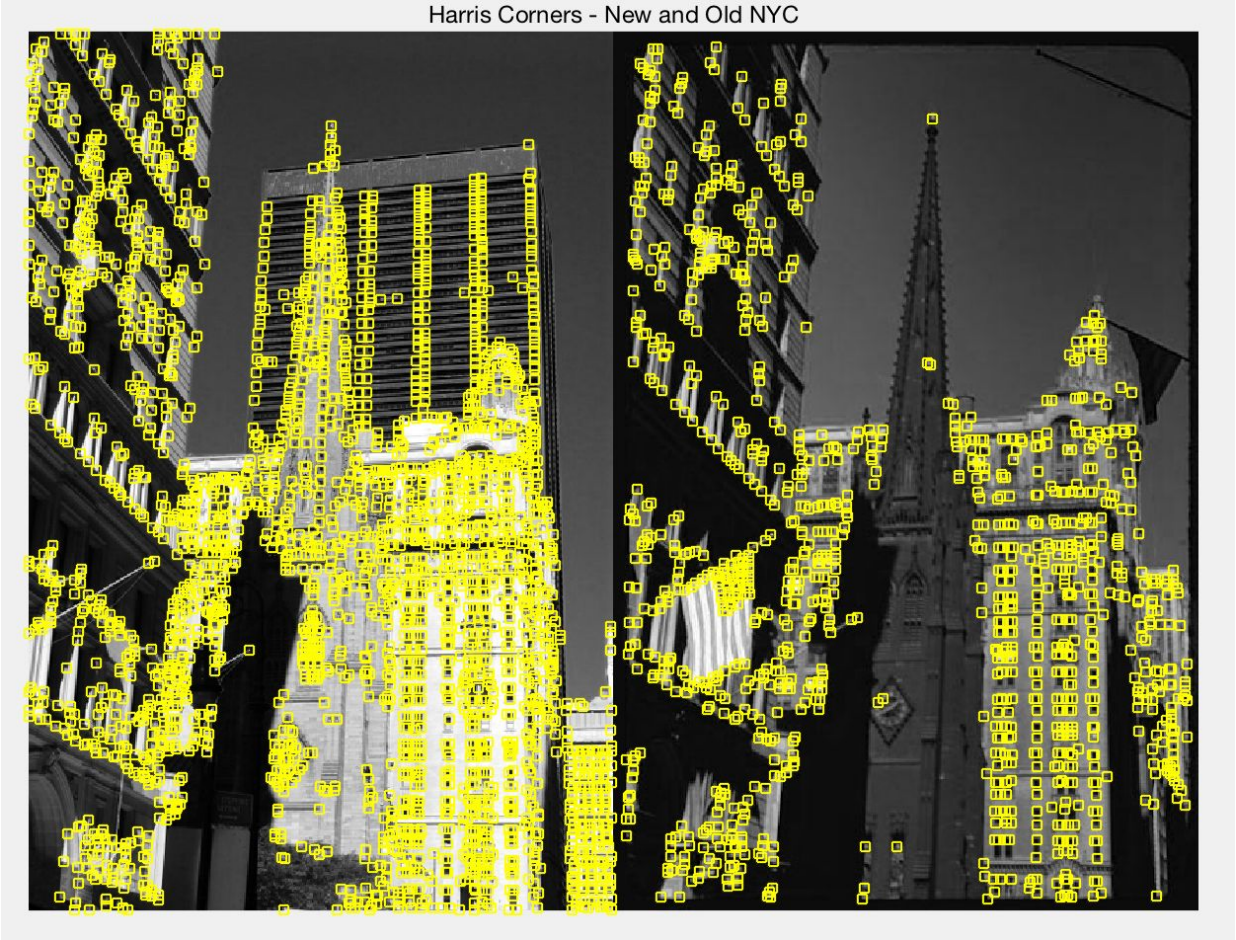
`[Ix, Iy] = intensityGradients(img);`

Filtering `Ix2`, `Iy2` and `Ixy` doesn't make much sense to me.

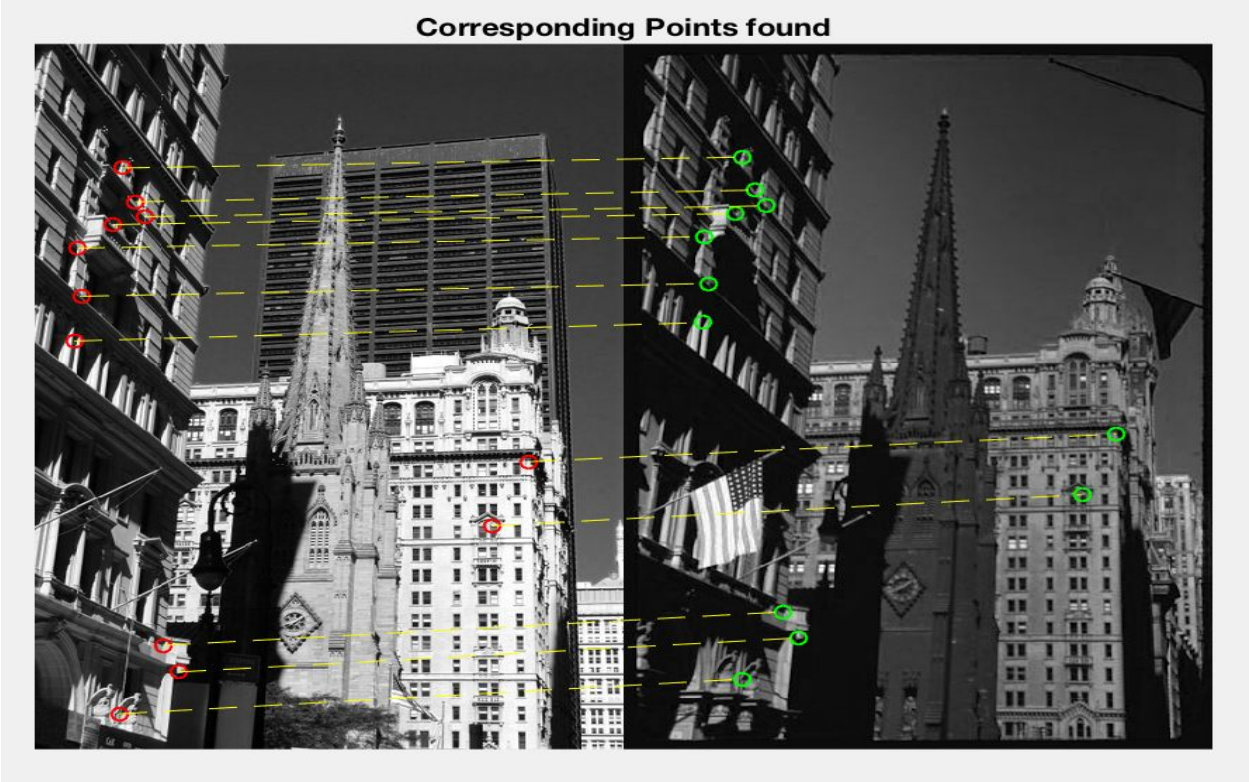
Further, I think your sample code is wrong here (does function `harrisResponse` have two or three input variables?):

```
H = harrisResponse(Ix2, Ixy, Iy2);
[...]

function K = harrisResponse(Ix, Iy)
```

```
% compute the derivatives
[dx, dy]=meshgrid(-1:1, -1:1);
image1_Ix = conv2(double(image1), dx, 'same');
image1_Iy = conv2(double(image1), dy, 'same');

image2_Ix = conv2(double(image2), dx, 'same');
image2_Iy = conv2(double(image2), dy, 'same');

sigma=1;
radius=1;
order=2*radius +1;
threshold=3000;

% gaussian filter
g=fspecial('gaussian', max(1, fix(6*sigma)), sigma);

% calculate entries of M
image1_Ix2 = conv2(double(image1_Ix.^2), g, 'same');
image1_Iy2 = conv2(double(image1_Iy.^2), g, 'same');
image1_Ixy = conv2(double(image1_Ix.*image1_Iy), g, 'same');

image2_Ix2 = conv2(double(image2_Ix.^2), g, 'same');
image2_Iy2 = conv2(double(image2_Iy.^2), g, 'same');
image2_Ixy = conv2(double(image2_Ix.*image2_Iy), g, 'same');

% harris corner response measure
R1 = (image1_Ix2.*image1_Iy2-image1_Ixy.^2)./(image1_Ix2+ima
R2 = (image2_Ix2.*image2_Iy2-image2_Ixy.^2)./(image2_Ix2+ima

% find local maxima
mx1 = ordfilt2(R1, order^2, ones(order));
mx2 = ordfilt2(R2, order^2, ones(order));
harris_points1 = (R1 == mx1) & (R1 > threshold);
```



Here is the harris corners we found. The left side image was taken in 2010, that it detect more corner points on the building than the right side image which taken in 1942.



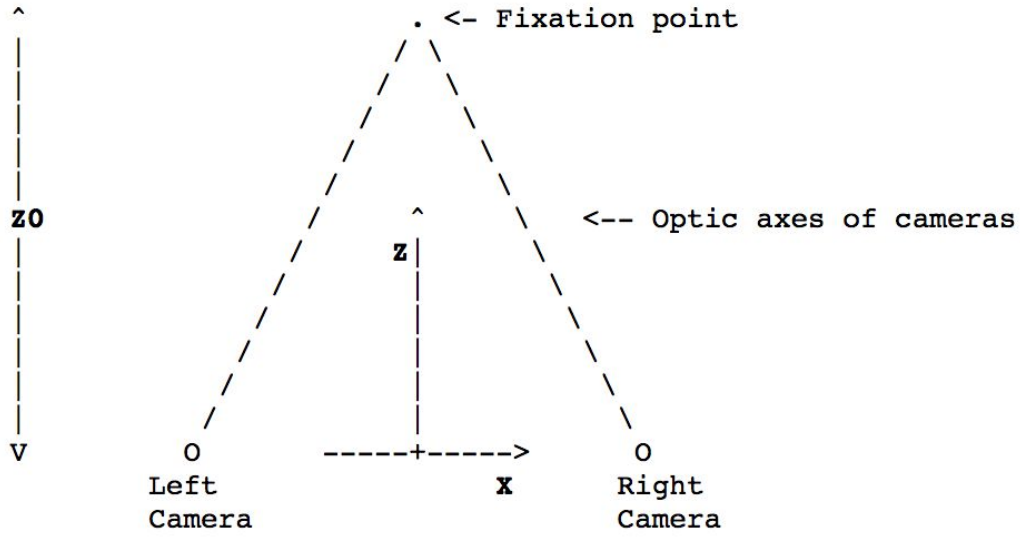
I picked up several points by hand. Then it shows the corresponding points in other image.

Epipolar Line

Converging Cameras

Stereo geometry for converging cameras. It is not usually convenient to set up the cameras with their axes parallel, because this limits the region of space in which objects are visible in both images. ... The image of an object at the fixation point is centred for both cameras, and so has zero disparity.

It is not usually convenient to set up the cameras with their axes parallel, because this limits the region of space in which objects are visible in both images. It is more normal to aim the cameras so that their axes are angled inwards, and converge on the objects of interest. The point in space where the optical axes intersect might be described as the fixation point for the cameras. The image of an object at the fixation point is centred for both cameras, and so has zero disparity.



It turns out that this makes the geometry considerably more complicated, and the exact equations for getting from image points to space points are much less straightforward than for parallel cameras. Matching points will not even generally lie at the same **y** coordinate. Provided the amount of convergence is small, though, we can ignore this *vertical disparity*, and there is a reasonable approximation which can be used for the equations. Call the depth of the fixation point **Z0**. Then the disparities can be adjusted by adding the quantity **fD/Z0**, which is the disparity an object at the fixation point would have if the axes were parallel. The new equations become:

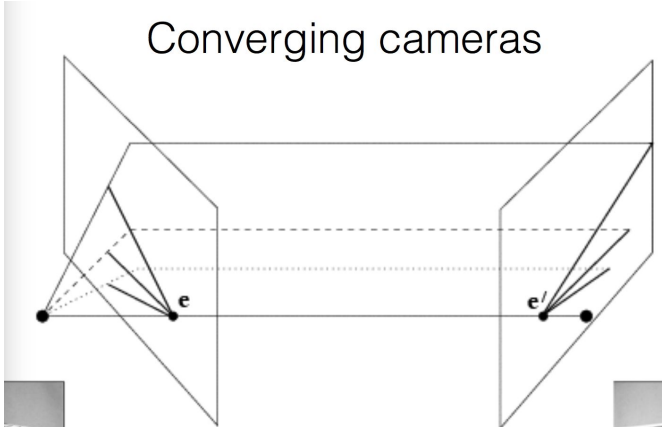
$$X = \frac{D\,(xL + xR)}{2\,p}$$

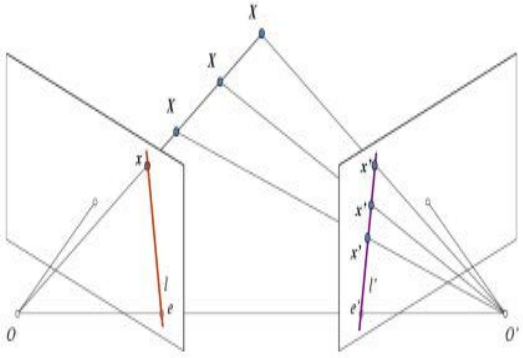
$$Y = \frac{D\,y}{p}$$

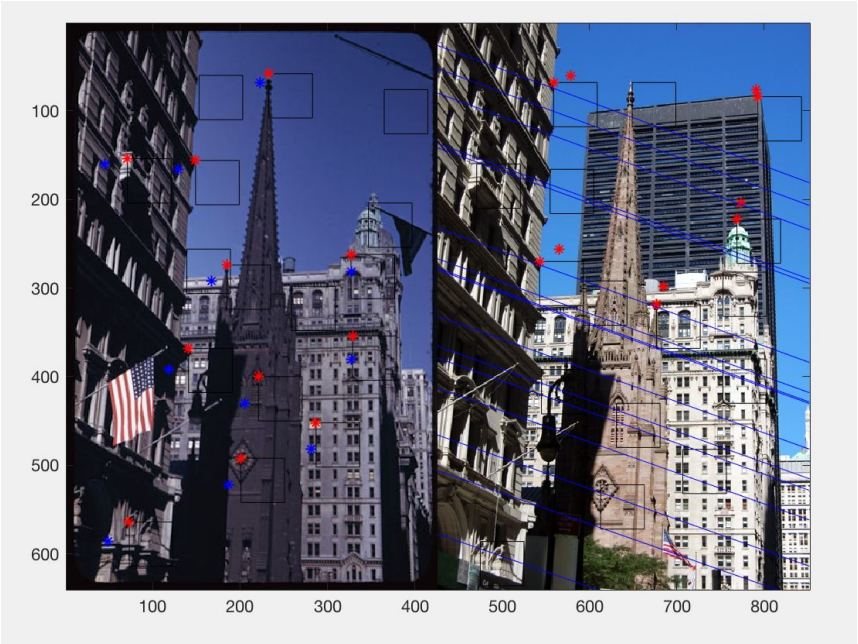
$$Z = \frac{D\,f}{p}$$

$$p = xL - xR + \frac{f\,D}{Z0}$$

Converging cameras







Where is the epipole in this image?
It's not always in the image

Related Matlab Code:

```
%% Generate the A matrix
[a b] = size(pl);
%% Singular value decomposition of A
for i=1:a

    x1 = pl(i,1);
    y1 = pl(i,2);
    x2 = pr(i,1);
    y2 = pr(i,2);
    A(i,:) = [x1*x2 y1*x2 x2 x1*y2 y1*y2 y2 x1 y1 1];

end
[U D V] = svd(A);
```

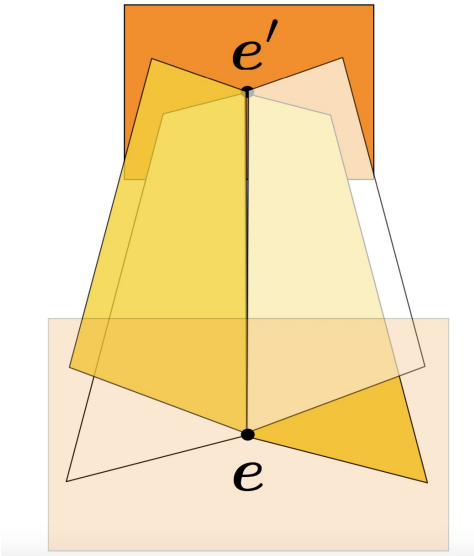
Matrix A is the factorization of A into the product of three matrices A = UDV T where the columns of U and V are orthonormal and the matrix D is diagonal with positive real entries. The SVD is useful in many tasks.

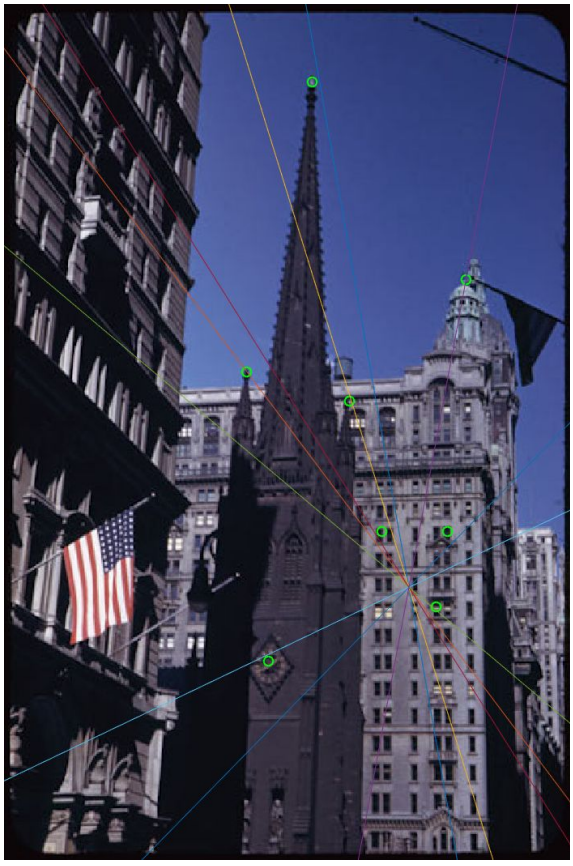
```
%% the estimate of F
f = V(:,9);
F = [f(1) f(2) f(3); f(4) f(5) f(6); f(7) f(8) f(9)];

[FU FD FV]= svd (F);
FDnew = FD;
FDnew(3,3) = 0;
FM = FU*FDnew*FV';

F = FM;
display(F);
```

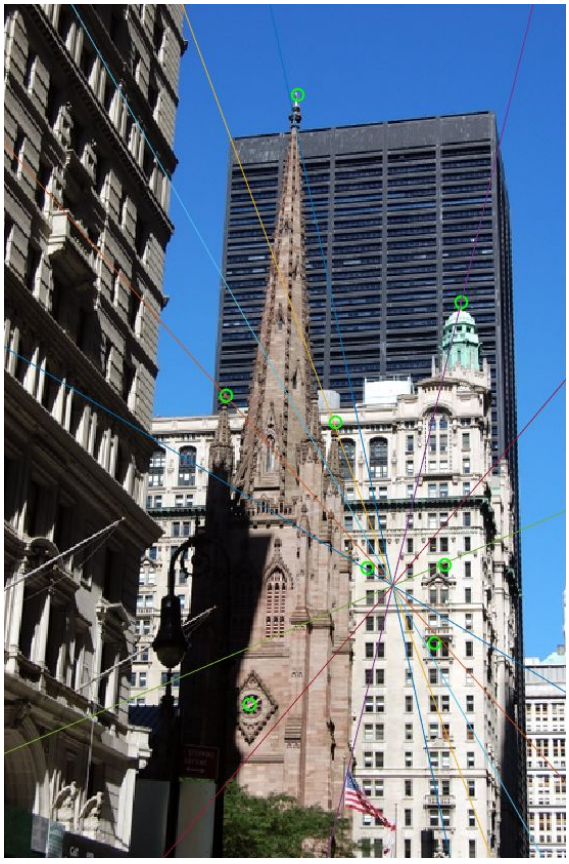
The fundamental matrix is the algebraic representation of epipolar geometry. In the following we derive the fundamental matrix from the mapping between a point and its epipolar line, and then specify the properties of the matrix. from a point in one image to its corresponding epipolar line in the other image.

Forward Moving Camera

Epipole has same coordinates in both images. Points move along lines radiating from e: "Focus of expansion"







There are 8 points picked up by hand, then more than 2 points not in the epipolar line. That we could know there have some mistake over the image.

Epipole has same coordinates in both images. Points move along lines radiating from “Focus of expansion”

```
%Make the x1 and x2 Homogeneous
homo_x2 = [x2, ones(8,1,'double')];
homo_x1 = [x1, ones(8,1,'double')];

%%
%Compute the F matrix
[F,inliers] = estimateFundamentalMatrix(x1,x2,'Method','Norm8Point');

%Find the epipolar lines
epiLines1 = [F * homo_x1']';
points = lineToBorderPoints(epiLines1,size(tr));

%Plot the epipolar lines on the image
line(points(:,[1,3])',points(:,[2,4])');
truesize;

%Find the epipole using the formula
[D1,E1] =eigs(F*F');
e1calculated = D1(:,1)./D1(3,1);
```

5. Conclusions

Compare two NYC images which took on 1942 and 2010. We could found the matched/corresponding points. After test by the epipolar line, most points are located in the line. That there is not change lots of features in these two images. They are similar over periods.

After we did this project, then we could know how the detect software works like we mention above. They usually find out most similar points, then test them by the epipolar line. If we cannot find out the most matched/corresponding points or it cannot pass through epipolar line, then we could definitely say these two images are different or unfound relationship through each other.