# Image Filtering and Hybrid Images

Detecting Harris Corners and Matching the keypoints using feature vectors!!

**Dipan banik**

Project 2
Semester 2


Big Data Analytics
RKMVERI
India
14 - 03 - 2021

# Contents

# 1 INTRODUCTION :

In this project we will take two images of the same object from two different angles and will check if we can match them.For matching two images we have to match them with respect to the corners because comparing corners in any two images is easier that any other points on those images.So in this project we will discuss the procedures to identify those corners and extract special features with the help of Harris Corners and SIFT with which we can match two image frames.

# 2 ALGORITHMS :

## 2.1 Harris Corner Detection :

We took the image ,shown below-

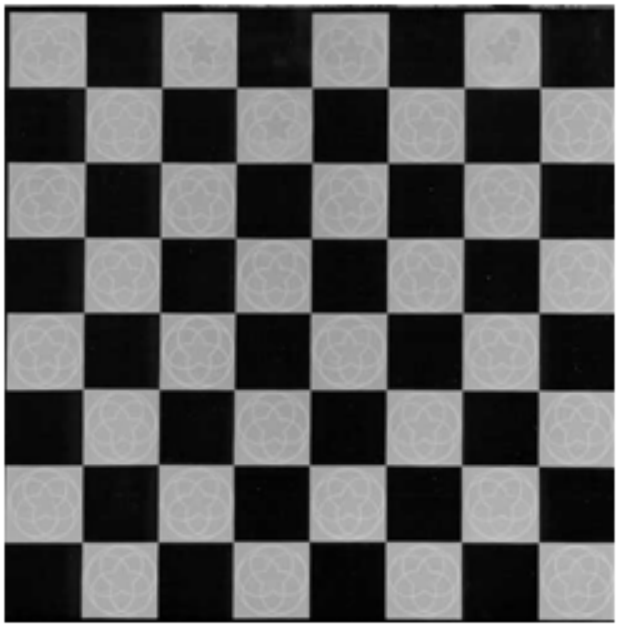


Figure 1: Main Image

Key idea behind the corner extraction is to identify those points at which the image has gradient changes at two or more directions.
**1.** At first we applied convolution with a gaussian filter on that image to remove all the

gaussian noise which can mislead us.

**2.**Then we have taken image derivatives with respect to x(Ix)and y(Iy) axis. The image derivatives are as shown below-

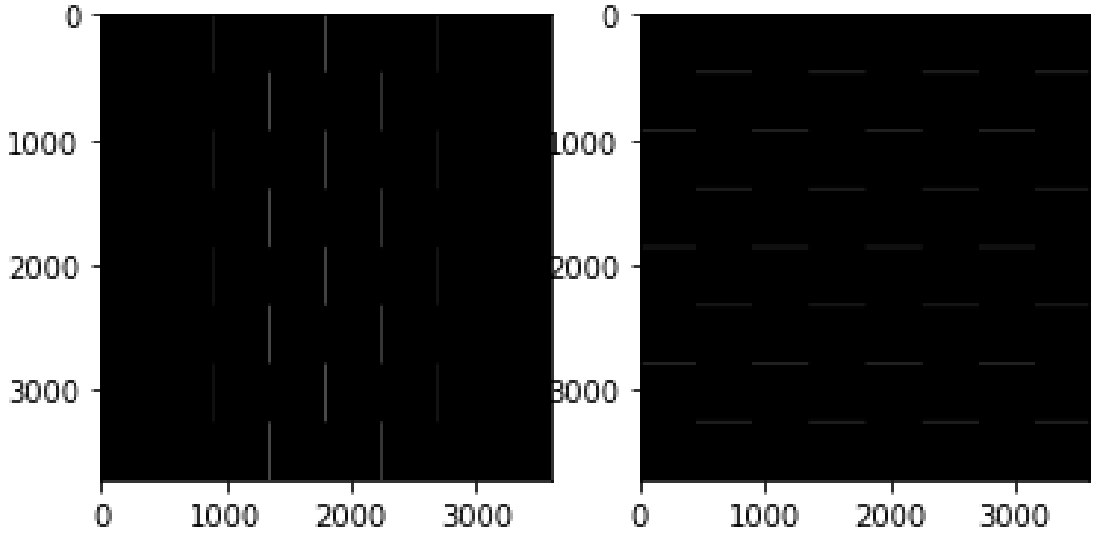



Figure 2: derivatives along x axis and y axis respectively

**3.** Now, to do our desired work , we have to detect the change in appearance of window w(x,y) for the shift [u,v] (Here we will use a Gaussian window) :

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \tag{1}$$

Now basically we want to find out how this function behaves for small shifts.But calculating it naively is a very slow method , so we recall Taylor series expansion.
Local quadratic approximation of E(u,v) in the neighbourhood of (0,0) is given by the second order Taylor expansion :

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + (1/2) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{2}$$

Now,

$$E_u(u,v) = \sum_{x,y} 2w(x,y)[I(x+u, y+v) - I(x,y)]I_x(x+u, y+v) \tag{3}$$

$$E_v(u,v) = \sum_{x,y} 2w(x,y)[I(x+u, y+v) - I(x,y)]I_y(x+u, y+v) \tag{4}$$

$$\begin{aligned} E_{uu}(u,v) &= \sum_{x,y} 2w(x,y)I_x(x+u,y+v)I_x(x+u,y+v) \\ &+ \sum_{x,y} 2w(x,y)[I(x+u,y+v)-I(x,y)]I_{xx}(x+u,y+v) \end{aligned} \tag{5}$$

$$\begin{aligned} E_{vv}(u,v) &= \sum_{x,y} 2w(x,y)I_y(x+u,y+v)I_y(x+u,y+v) \\ &+ \sum_{x,y} 2w(x,y)[I(x+u,y+v)-I(x,y)]I_{yy}(x+u,y+v) \end{aligned} \tag{6}$$

$$\begin{aligned} E_{uv}(u,v) &= \sum_{x,y} 2w(x,y)I_y(x+u,y+v)I_x(x+u,y+v) \\ &+ \sum_{x,y} 2w(x,y)[I(x+u,y+v)-I(x,y)]I_{xy}(x+u,y+v) \end{aligned} \tag{7}$$

So, we can easily check that,

$$E_u(0,0) = E_v(0,0) = 0 \tag{8}$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y)I_x(x,y)I_x(x,y) \tag{9}$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y)I_y(x,y)I_y(x,y) \tag{10}$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y)I_x(x,y)I_y(x,y) \tag{11}$$

The quadratic approximation simplifies to -

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{12}$$

And here our desired M matrix is -

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} Ix*Ix & Ix*Iy \\ Ix*Iy & Iy*Iy \end{bmatrix} = \sum_{x,y} w(x,y) \begin{bmatrix} Ix^2 & Ix*Iy \\ Ix*Iy & Iy^2 \end{bmatrix} \tag{13}$$

**4.** We can say that this M matrix determines the gradient direction at every point on that image,so , we will try find those points at which both the eigenvalues of this M matrix are large . So to achieve this, we have calculated the R score of the whole image at every position as,

$$\begin{aligned} R &= det(M) - \alpha * trace(M)^2 \\ &= ((Ix^2 * Iy^2) - (Ix * Iy)^2) - \alpha * (Ix^2 + Iy^2) \end{aligned} \tag{14}$$

Where, $\alpha$ usually lies between (0.04-0.06) and as we know that det(M)=product of eigenvalues and trace(M)=sum of eigenvalues.

**5.** Now we have taken only those points whose R value is higher than a certain threshold*Max(R).so, our list of corners are -

$$Corners = R[R > Threshold * Max(R)] \tag{15}$$

**6. Non maximum suppression :** Now we have run a window (of size approximately 10*10)over the whole image and taken the points with highest R value within the window, in our final corner-points list.

Now , we have our final list of corner points, and after marking the corners on that image, the image looks like-

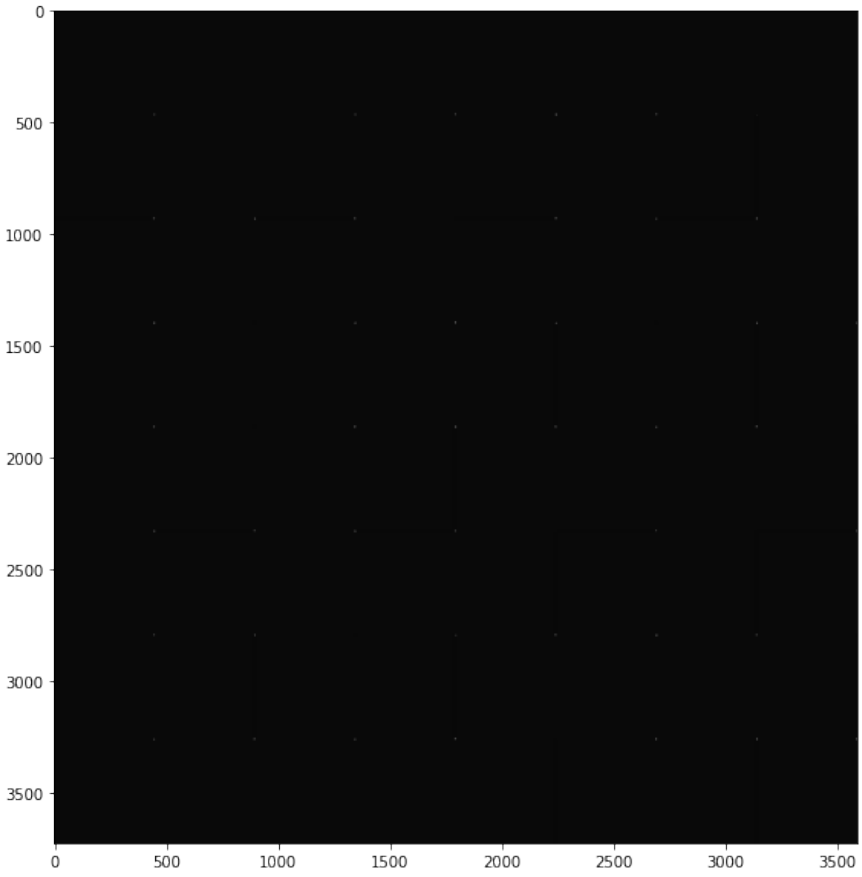



Figure 3: Corner points marked

Now ,to check if our harris corner detection algorithm is working properly or not, we have

used some library functions and tried to match different images of the same object , taken from different perspectives.

## 2.2 SCALE-INVARIANT FEATURE TRANSFORM (SIFT)

: SIFT features (called keypoints) are invariant to image scale and rotation, and are robust across a range of affine distortions, changes in 3-D viewpoint, noise, and changes of illumination. The input to SIFT is an image. Its output is an n-dimensional feature vector whose elements are the invariant feature descriptors. We begin our discussion by analyzing how scale invariance is achieved by SIFT.

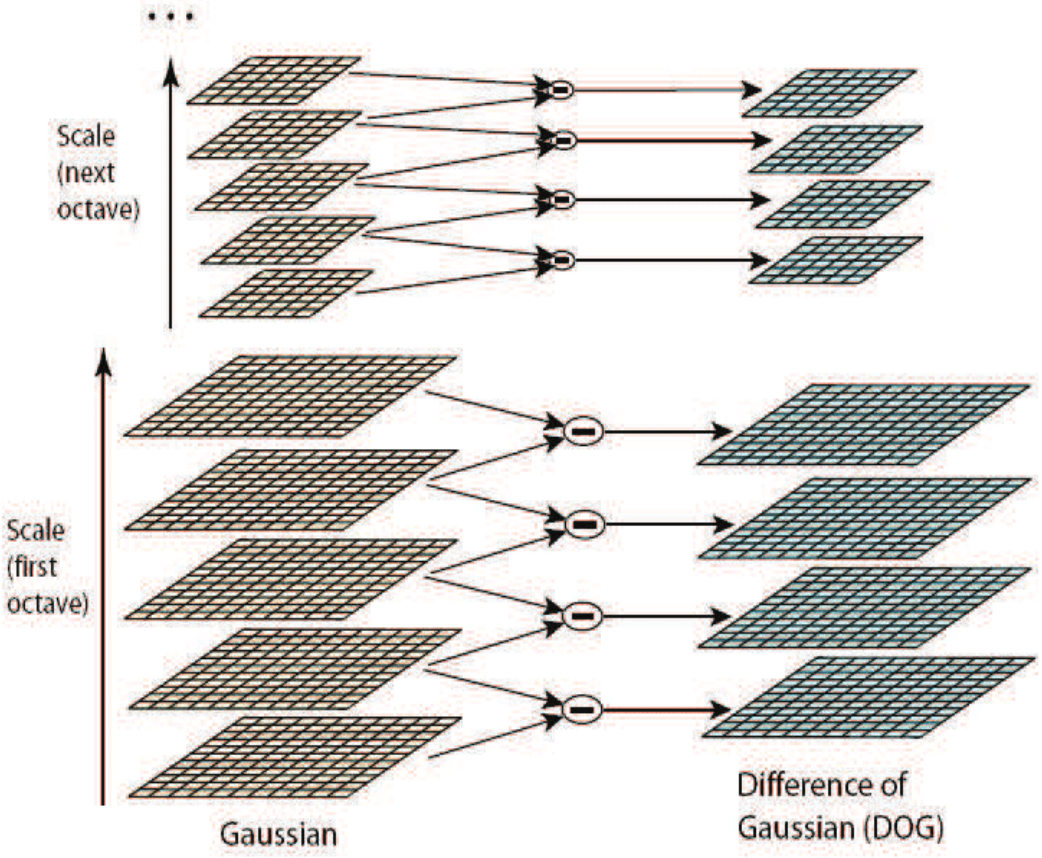
**SCALE SPACE:** The first stage of the SIFT algorithm is to find image locations that are invariant to scale change. This is achieved by searching for stable features across all possible scales, using a function of scale known as scale space. Scale space repre-sents an image as a one-parameter family of smoothed images, with the objective of simulating the loss of detail that would occur as the scale of an image decreases. The parameter controlling the smoothing is referred to as the scale parameter.In SIFT, Gaussian kernels are used to implement smoothing, so the scale parameter is the standard deviation. The reason for using Gaussian kernels in based on work performed by Lindberg [1994], who showed that the only smoothing kernel that meets a set of important constraints, such as linearity and shift-invariance, is the Gaussian lowpass kernel. Based on this, the scale space, $L(x, y, \sigma)$, of a grayscale image, f ( x , y ), † is produced by convolving f with a variable-scale Gaussian kernel,$G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * f(x, y)$$

where the scale is controlled by parameter $\sigma$, and G is of the form:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{\sigma^2}}$$

The input image f ( x , y ) is successively convolved with Gaussian kernels having standard deviations $\sigma$,$k\sigma$, $k^2\sigma$, . . . to generate a "stack" of Gaussian-filtered (smoothed) images that are separated by a constant factor k,SIFT subdivides scale space into octaves, with each octave corresponding to a doubling of $\sigma$




**Obtain the initial keypoints:** Compute the difference of Gaussians, $D(x, y, \sigma)$ from the smoothed images in scale space using

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Extrema (maxima or minima) of the $D(x, y, \sigma)$ images in an octave are detected by comparing a pixel (shown in black) to its 26 neighbors (shown shaded) in $3 \times 3$ regions at the current and adjacent scale images.

**Improve the accuracy of the location of the keypoints:** Interpolate the values of $D(x, y, \sigma)$ via a Taylor expansion. The improved key point locations are given by Equation

$$x = -H^{-1}(\Delta D)$$

where H is the Hessian matrix.

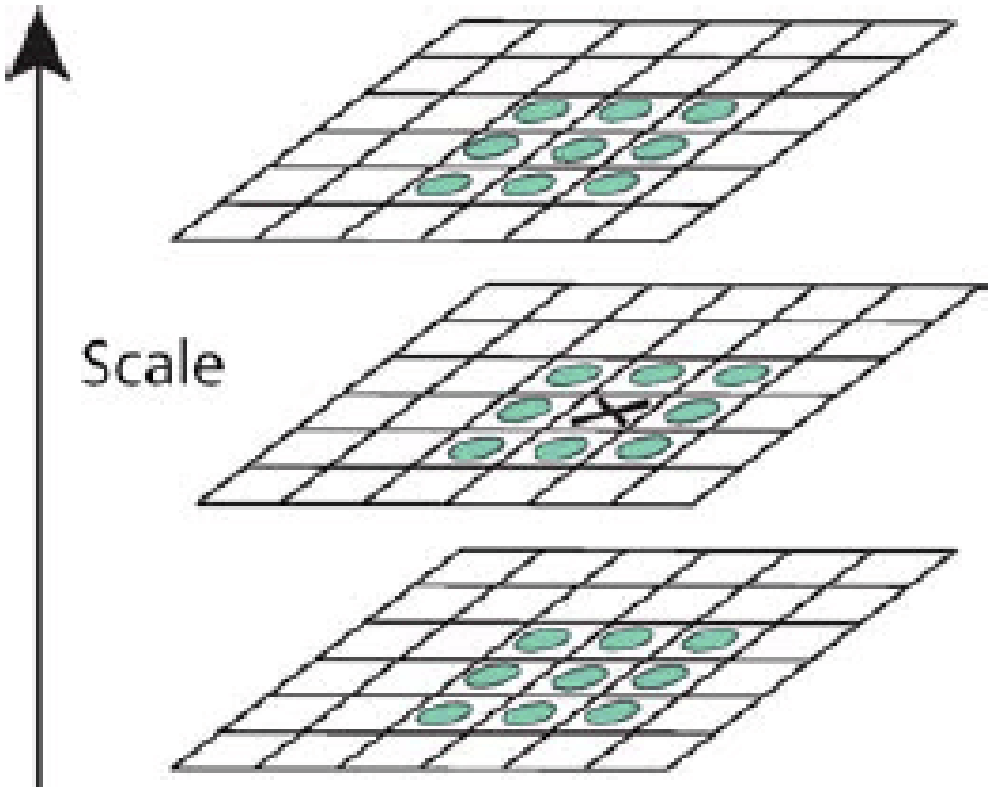



Figure 4: Corner points marked

**Compute keypoint orientations:** Use Eqs.

$$M(x, y) = [(L(x + 1, y) - L(x - 1, y)) + (L(x, y + 1)L(x, y - 1))]^{1/2}$$

and

$$\theta(x, y) = \tan^{-1}[(L(x, y + 1) - L(x, y1))/(L(x + 1, y) - L(x1, y))]$$

to compute the magnitude and orientation of each keypoint using the histogram-based procedure discussed in connection with these equations.

**Compute keypoint descriptors:** Use the method summarized in Fig. to ompute a feature (descriptor) vector for each keypoint. If a region of size 16 $\times$ 16 around each keypoint is used, the result will be a 128-dimensional feature vector for each keypoint.

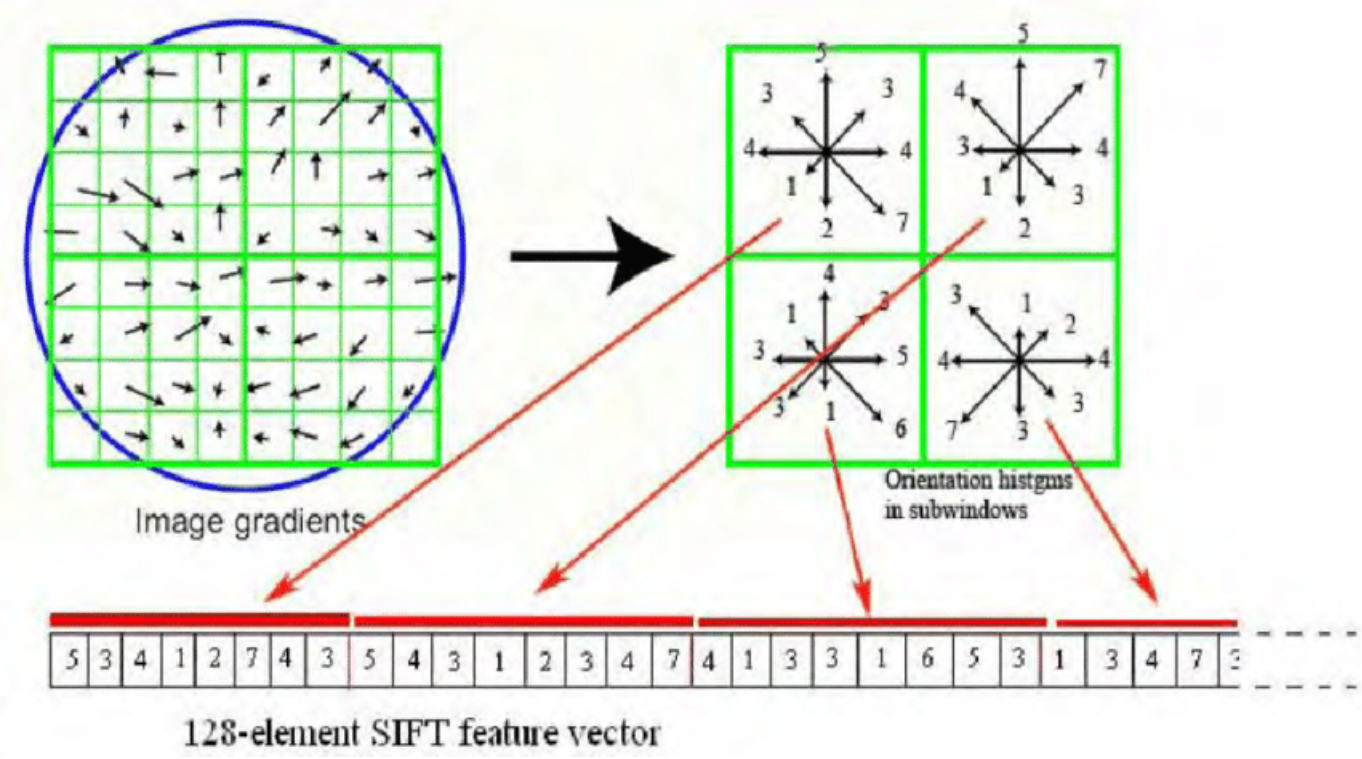



Figure 5: Feature Descriptor

# 3 Results :

## 3.1 Using Harris corner

**1. 1.**

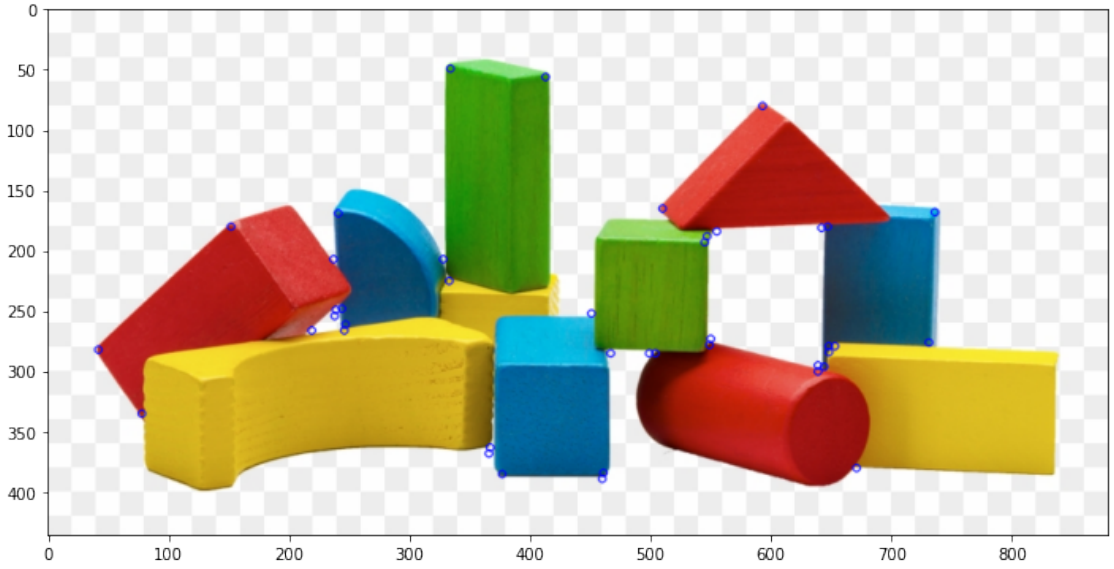



Figure 6: Corner points marked

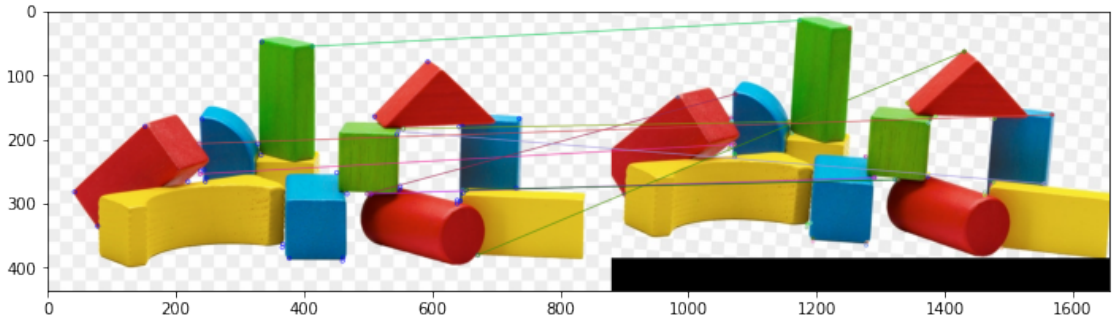



Figure 7: Corner points marked

## 3.2 Using SIFT

**2.**

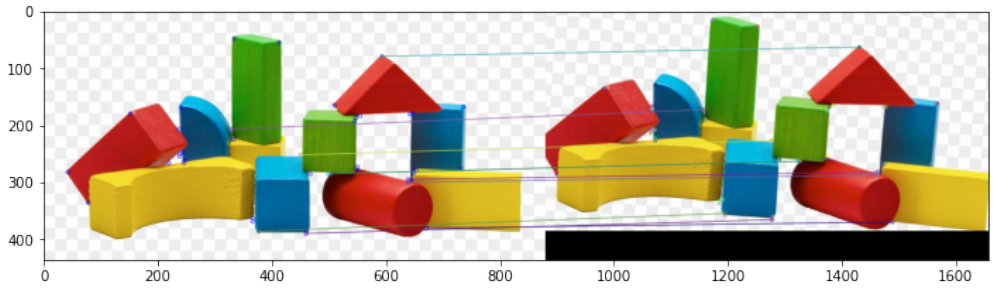



Figure 8: Corner points marked

# 4 Conclusion :

As we can see our algorithm is doing pretty good. Though it hasn't marked all the corners properly but we are quite satisfied with the result.Probably the orientation of few corner points are same though they are not actually same. That may be a reason of our algorithm's failure.Later we will try to improve our algorithm.

# 5 Some More Examples :

## 5.1 Using Harris corner

**1.**

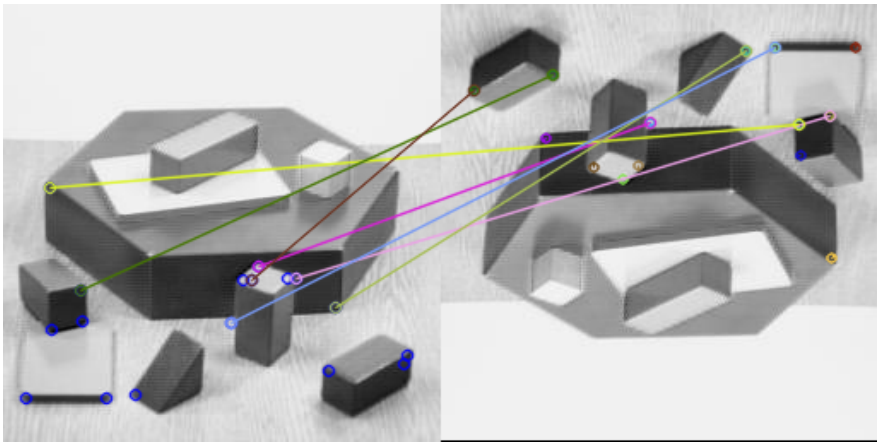


Figure 9: Corner points marked

## 5.2 Using SIFT

**2.**

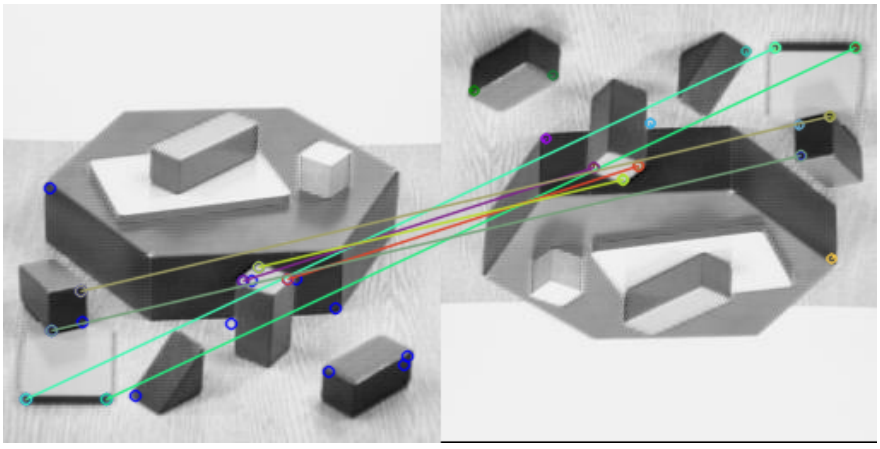


Figure 10: Corner points marked

## 5.3 Using Harris corner

**1.**

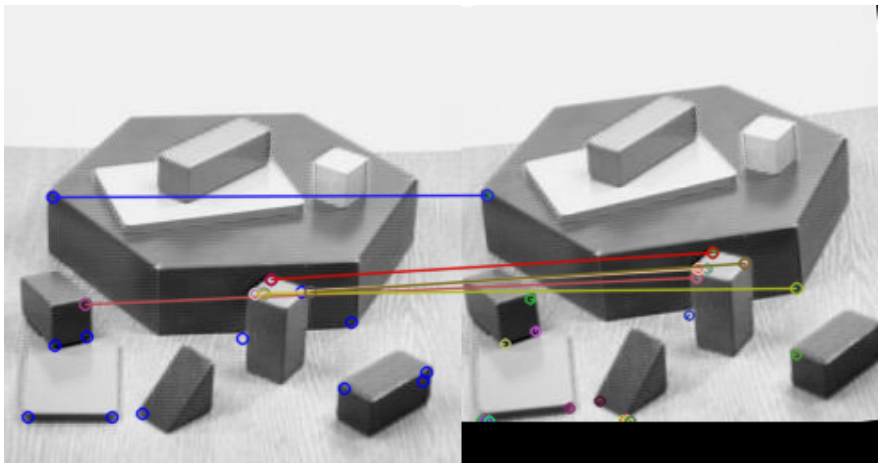


Figure 11: Corner points marked

## 5.4 Using SIFT

**2.**

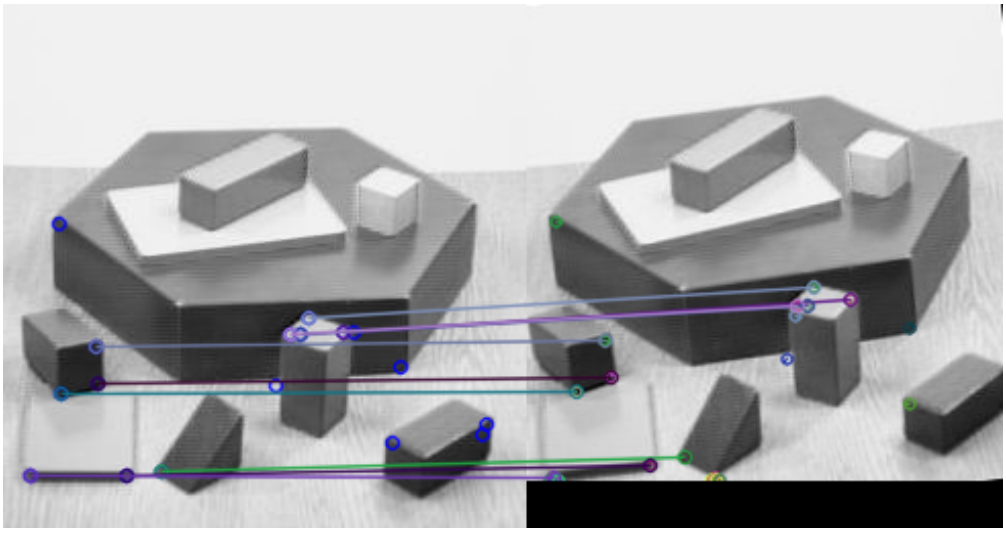


Figure 12: Corner points marked

## 5.5 Using Harris corner

**1.**

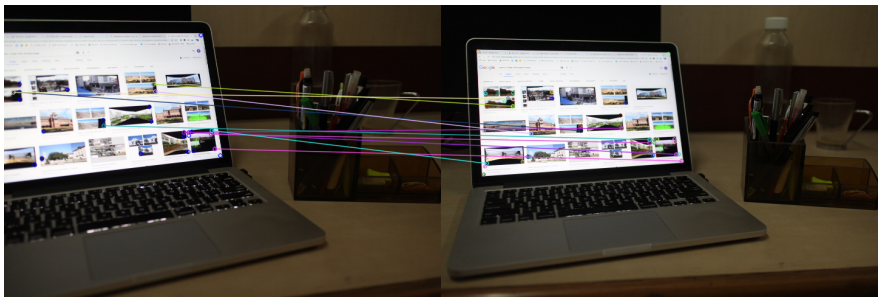


Figure 13: Corner points marked

## 5.6 Using SIFT

**2.**

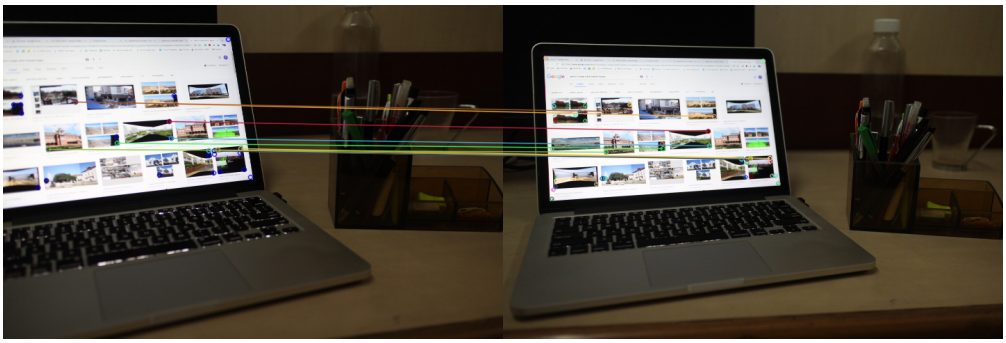


Figure 14: Corner points marked

## 5.7 Using Harris corner

**1.**

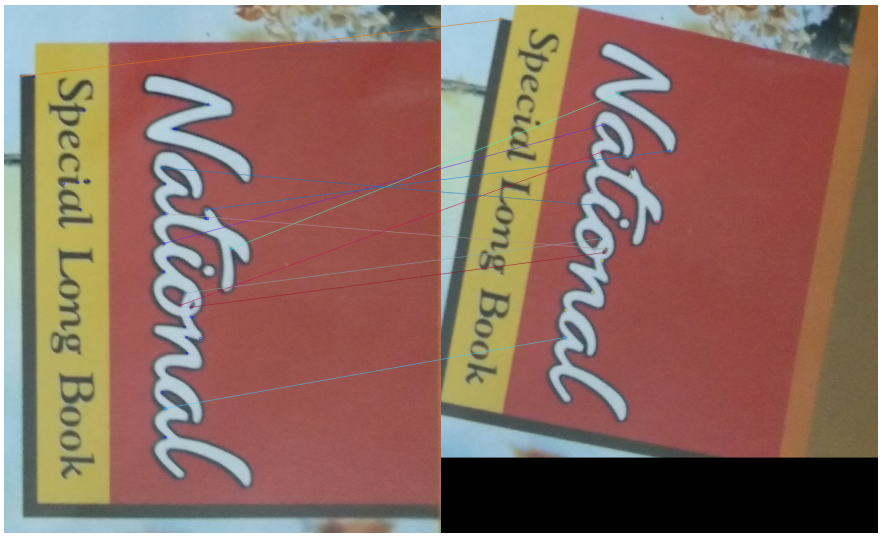



Figure 15: Corner points marked

## 5.8 Using SIFT

**2.**

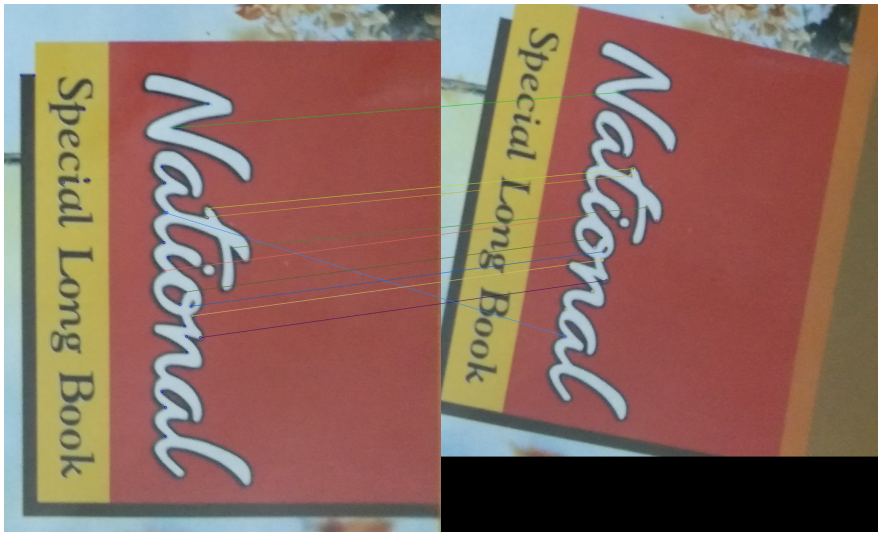



Figure 16: Corner points marked

## 5.9 Using Harris corner

**1.**

Figure 17: Corner points marked

## 5.10 Using SIFT

**2.**

Figure 18: Corner points marked