CV PROJECT

Detection of Harris Corner and SIFT Descriptor

Mahendra Nandi
Sourav Karmakar
Guided By-Tamal Maharaj
MSc. COURSE ON BIG DATA ANALYTICS
Department of COMPUTER SCIENCE
Ramakrishna Mission Vivekananda Educational Research Institute BELUR

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our supervisor, Tamal Maharaj, Department of Computer Science as well as our HOD Swathy Prabhu Maharaj who gave the golden opportunity to do this wonderful project on the topic Hybrid Image , which also helped me in doing a lot of Research and we came to know about so many new things we are really thankful to them. Secondly we would also like to thank our friends who helped us a lot in finalizing this project within the limited time frame.

Abstract

Computer Vision tasks such as stereo and motion estimation require finding corresponding features across two or more views. Corner detection algorithm is an algorithm which helps in identifying the corners in an image. Corners are mainly formed by the combination of two or more edges. These corners may or may not define the boundary of an image. Here the method used is Harris corner detection algorithm. It helps in pointing out the corners in a color image, for each component. This improves the detection efficiency and the experimental results shows that this method is more reliable than traditional methods.

Contents

1. Introduction

The first thing that makes a good feature is simply repeatability or precision, meaning if we find something in the left image, we better find that in the right image. Or if we have a sequence of images that we are trying to line, we want to find that same feature point. The other thing is that we should be able to be precise in our description. The idea is that if we find the point, we find it every time, and we find it more or less in the same place, with respect to the scene. The next important thing about a feature was the saliency or the matchability and what we mean by that is, that it should have a description that's distinctive.

How corner detection works?

We consider a window and run it over the whole image. Shifting an window in any direction should give a large change in intensity. If the window is over a flat area of the image then there will be obviously be no intensity change when the window moves. If the window is over an edge there will only be an intesity change if the window moves in one direction.

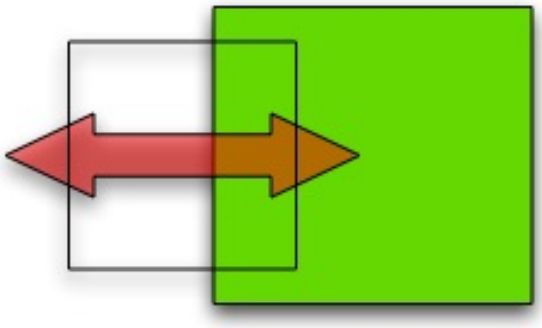


Fig. 1. Gradient change only in one direction

If the window is over a corner then there will be a change in all directions, and therefore we know there must be a corner.

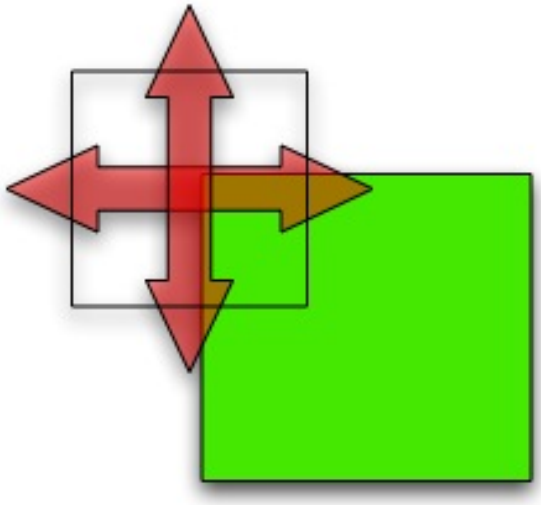


Fig. 2. Gradient change only in all direction

Change in intensity for the shift

Let the image be a scalar function I : $\Omega \rightarrow$ R and h a small increment around any position in the domain, x $\in \Omega$. The function w(x) allows selecting the support region that is typically defined as a rectangular or Gaussian function. Corners are defined as the points x that maximize the following functional for small shifts h,

$$E(h) = \sum w(x) * [I(x+h) - I(x)]^2 \tag{1}$$

$[I(x+h) - I(x)]^2$ will be near 0 for nearly constant patches or will be larger for very distinctive patches.
But this is very slow to compute naively. Computational cost is $O(window\_width^2 * shift\_range^2 * image\_width^2)$

Approximation with Taylor Series expansion

Taylor Series can be used to linearize the expression I(x+h) as,

$$I(x+h) \approx I(x) + \nabla I(x)^T h$$

so the right hand side of equation (1) becomes

$$E(h) \approx w(x)(\nabla(I(x)h)^2 dx = \sum w(x)(h^T \nabla I(x) \nabla I(x)^T h) \tag{2}$$

$$Let\ M = \sum w(x)(\nabla I(x) \nabla I(x)^T) = \begin{bmatrix} \sum w(x) {I_x}^2 & \sum w(x) I_x I_y \\ \sum w(x) I_x I_y & \sum w(x) {I_y}^2 \end{bmatrix} \tag{3}$$

Interpreting the eigenvalues

The maxima of (2) are found through the analysis of this M matrix. The largest eigenvalue of M corresponds to the intensity variation in its orthogonal direction. Let λ_1 and λ_2 are two eigenvalues of the Matrix M. Analyzing the eigenvalues, we may find three possible situations:

Both eigenvalues are small, i.e., $\lambda_1 \approx \lambda_2 \approx 0$,then the region is likely to be a homogeneous region with intensity variations due to the presence of noise.

One of the eigenvalues is much larger than the other one, $\lambda_1 >> \lambda_2 \approx 0$, then the region is likely to belong to an edge, with the largest eigenvalue corresponding to the edge orthogonal direction.

Both eigenvalues are large, $\lambda_1 > \lambda_2 >> 0$, then the region is likely to contain large intensity variations in the two orthogonal directions, therefore corresponding to a cornere-like structure.

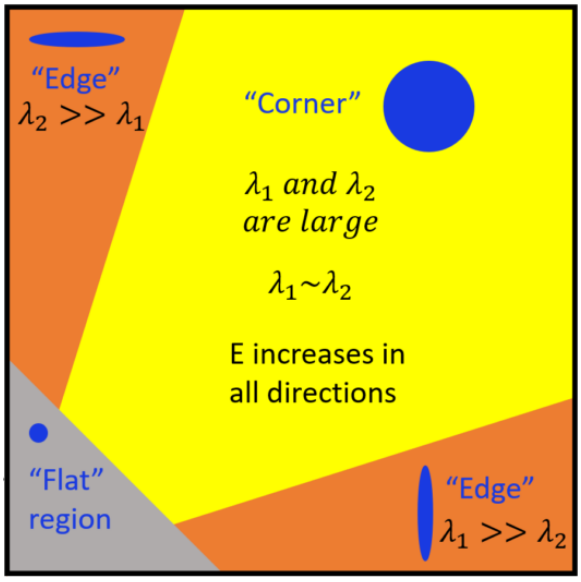


Fig. 3. Eigenvalues to detect corner

2. Harris Corner Detection Algorithm

Finding the corners

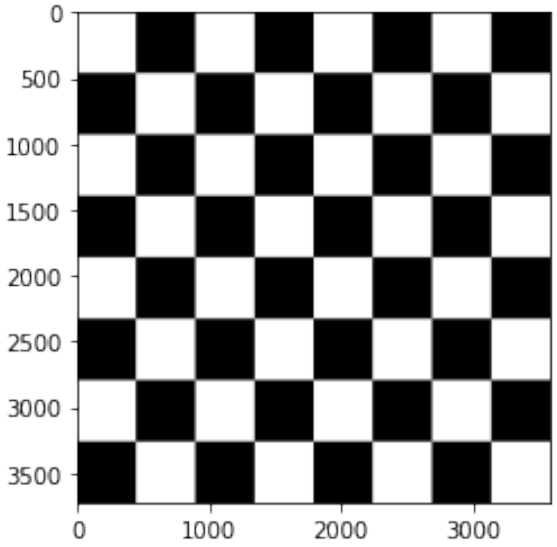
procedure HARRIS_CORNER(image,window_size,alpha,threshold,nms_size):
    apply gaussian blur to the image
    apply sobel filter along x and y direction and store it to I_xI_y
    get I_x^2 by $I_x * I_x$
    get I_y^2 by $I_y * I_y$
    get I_{xy} by $I_x * I_y$
    apply gaussian blur to I_x^2,I_y^2,I_{xy}
    form the matrix M with I_x^2,I_y^2,I_{xy}
    get the determinant of M
    get the trace of M
    form R= det(M) -alpha*trace(M)
    if $each\_pixel < threshold * maximum\_of\_R$ **then**
        $each\_pixel$=0
    take $nms\_wid = floor(nms\_size/2)$
    while i<$image\_height - nms\_wid$ **do** ▷ applying non-maximum supression
        while j<$image\_width - nms\_wid$ **do**
            $window = square\ matrix\ of\ size\ 2 * nms\_wid$
            window=the specified position in R
            apply non-maximum supression to this window
            replace the window in R
            j=j+nms_size
        i=i+nms_size
    return R, I_x, I_y

Get the potential corners

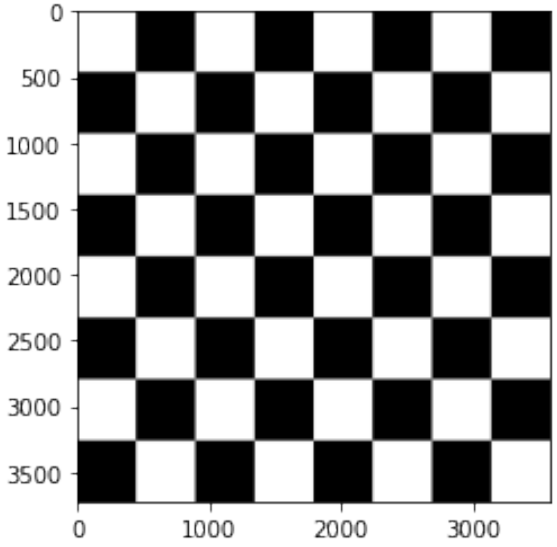
procedure GET_KEYPOINTS(corners,Ix,Iy,threshold):
    keypoints=[]
    while i<height_of_corner_matrix **do**
        while j<width_of_corner_matrix **do**
            if each_corner>threshold **then**
                $a = each\_corners\ with\ its\ gradient\ and\ location\ information$
                append(a) in keypoints
            j=j+1
        i=i+1

3. Harris Corner Detection Step by Step Visualization

Applying gaussian to remove the noise



The image may have some noise. Hence we will apply a gaussian blur with small gaussian kernel to the image to reduce the noise. We should should not blur too much,otherwise we will lose corner infromation from the image.



Applying Derivatives to the Image to have gradient information

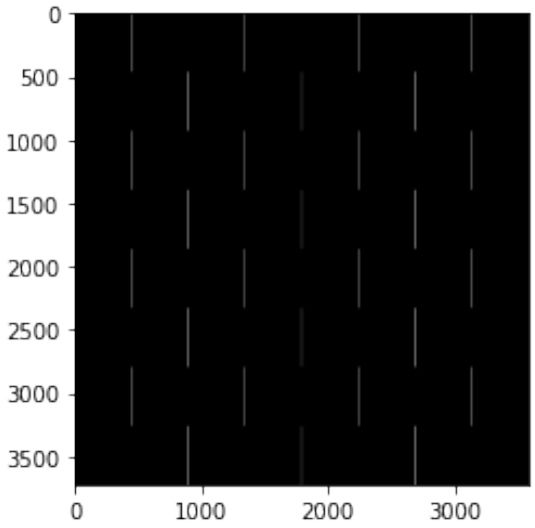


Fig. 4. Gradient along x direction ($\frac{\partial I}{\partial y}$)

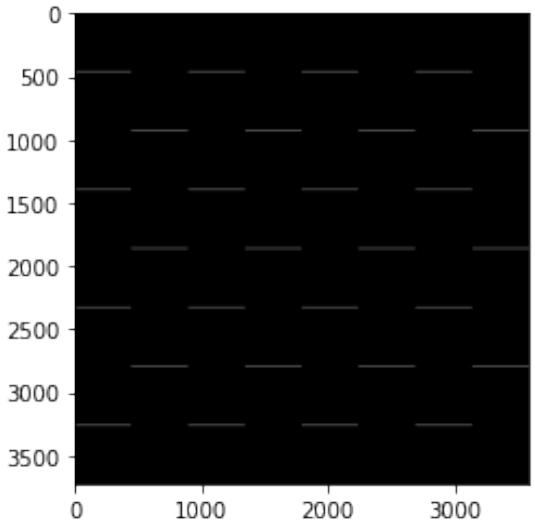
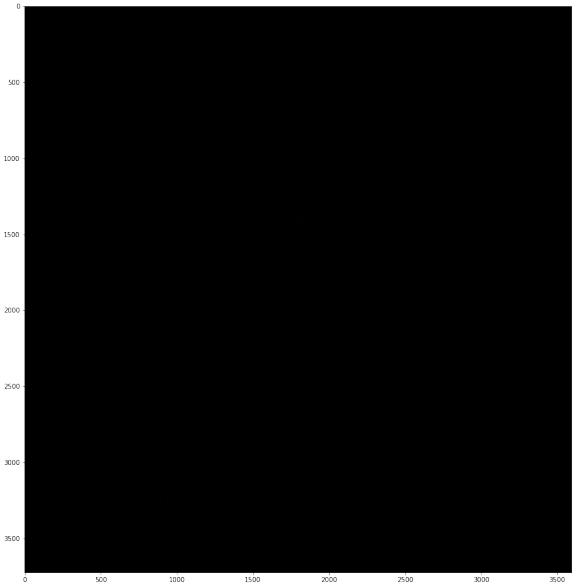


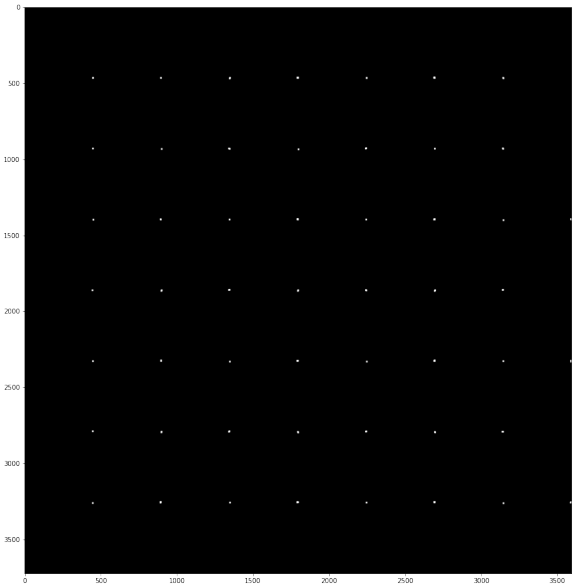
Fig. 5. Gradient along y direction ($\frac{\partial I}{\partial y}$)

Sobel filter along x and y direction are used to get the derivatives of the gaussian blurred image along x and y directions.

Applying thresholding and non-maximum supression



This Response matrix contains thresholding and non maximum supression of pixels. Corner pixels aren't visible since they are very few in number. Thresholding requires for taking the large corner response points and Non-maximum supression is required for taking the local maxima of that large corner response points.



This is the same Response Matrix with dialation of the corner pixels.

4. Keypoint Matching between two Stereo Images

Finding the kepoints between two images

Fig. 6. Corner points of first image

Fig. 7. Corner points of second image

Matched corners using Harris Corner Method

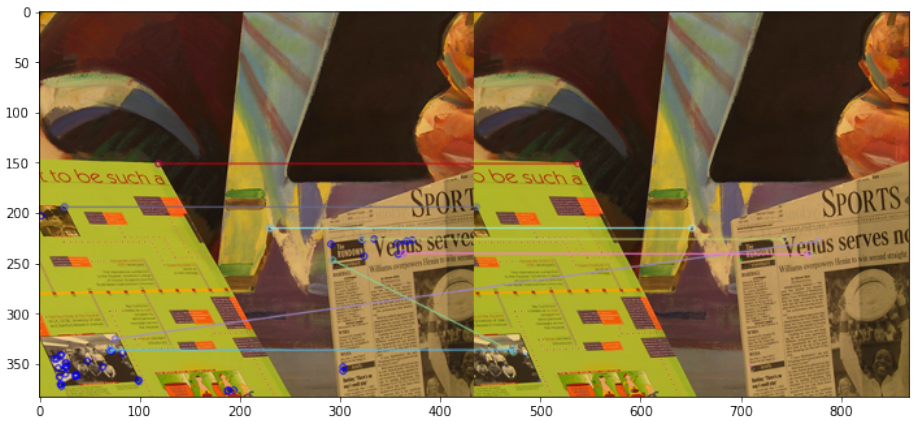


Fig. 8. Corners of two Stereo Images have matched

Two Stereo images have been matched with more richer information of the corner pixels.

5. Overview of SIFT algorithm

Main steps in SIFT algorithm are as follows:

Scale-space peak selection

Key-point Localization

Orientation Assignment

Key-point descriptor

Key-point Matching

However in this project our main focus will be on Key-point Descriptor. Here we will take the ners as keypoints from the Harrish corner detection method previously described then we make a descriptor by taking the neighbourhood of that keypoints . Here we represent the neighbourhood as a 128 dimensional vector having direction of gradients. Below it is completely described how to make it.The complete steps are -

1. Scale-space peak selection

From the image above, it is obvious that we can't use the same window to detect keypoints with different scale. It is OK with small corner. But to detect larger corners we need larger windows. For this, scale-space filtering is used. In it, Laplacian of Gaussian is found for the image with various σ values. LoG acts as a blob detector which detects blobs in various sizes due to change in σ . In short, σ acts as a scaling parameter. For eg, in the above image, gaussian kernel with low σ gives high value for small corner while guassian kernel with high σ fits well for larger corner. So, we can find the local maxima across the scale and space which gives us a list of (x,y,σ) values which means there is a potential keypoint at (x,y) at σ scale.

But this LoG is a little costly, so SIFT algorithm uses Difference of Gaussians which is an approximation of LoG. Difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different σ, let it be σ and kσ. This process is done for different octaves of the image in Gaussian Pyramid. It is represented in below image:

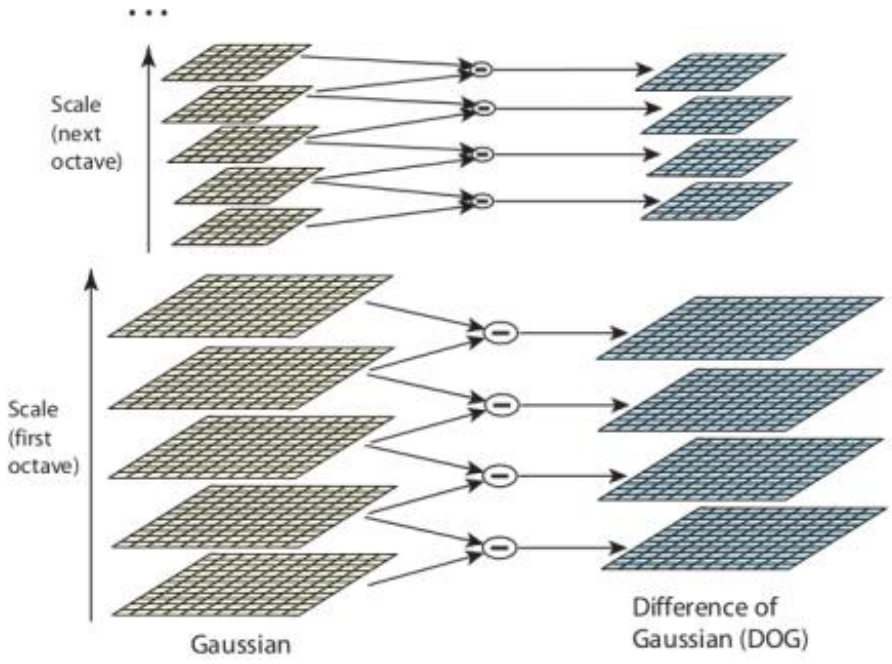


Fig. 9. Scale space with DoG

Once this DoG are found, images are searched for local extrema over scale and space. For eg, one pixel in an image is compared with its 8 neighbours as well as 9 pixels in next scale and 9 pixels in previous scales. If it is a local extrema, it is a potential keypoint. It basically means that keypoint is best represented in that scale. It is shown in below image:

Regarding different parameters, the paper gives some empirical data which can be summarized as, number of octaves = 4, number of scale levels = 5, initial σ=1.6, k=$\sqrt{2} etc as optimal values$.

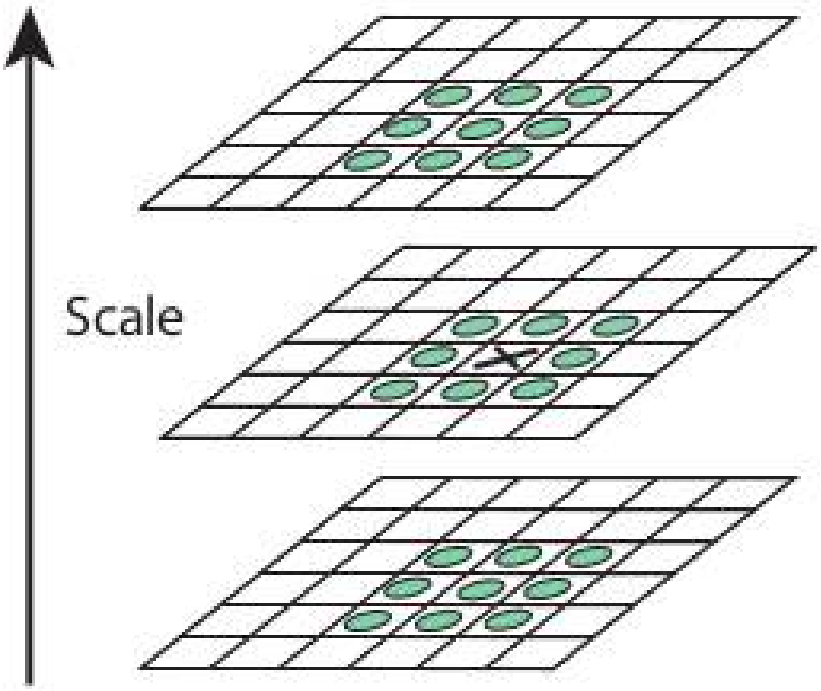


Fig. 10. Finding Local Extrema

2. Keypoint Localization

Once potential keypoints locations are found, they have to be refined to get more accurate results. They used Taylor series expansion of scale space to get more accurate location of extrema, and if the intensity at this extrema is less than a threshold value (0.03 as per the paper), it is rejected. This threshold is called contrastThreshold in OpenCV

DoG has higher response for edges, so edges also need to be removed. For this, a concept similar to Harris corner detector is used. They used a 2x2 Hessian matrix (H) to compute the pricipal curvature. We know from Harris corner detector that for edges, one eigen value is larger than the other. So here they used a simple function,

If this ratio is greater than a threshold, called edgeThreshold in OpenCV, that keypoint is discarded. It is given as 10 in paper.

So it eliminates any low-contrast keypoints and edge keypoints and what remains is strong interest points.

3. Orientation Assignment

Now an orientation is assigned to each keypoint to achieve invariance to image rotation. A neigbourhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created. (It is weighted by gradient magnitude and

gaussian-weighted circular window with σ equal to 1.5 times the scale of keypoint. The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions. It contribute to stability of matching.

4. Keypoint Descriptor

Now keypoint descriptor is created. A 16x16 neighbourhood around the keypoint is taken. It is devided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available. It is represented as a vector to form keypoint descriptor. In addition to this, several measures are taken to achieve robustness against illumination changes, rotation etc.

5. Keypoint Matching

Keypoints between two images are matched by identifying their nearest neighbours. But in some cases, the second closest-match may be very near to the first. It may happen due to noise or some other reasons. In that case, ratio of closest-distance to second-closest distance is taken. If it is greater than 0.8, they are rejected. It eliminates around 90% of false matches while discards only 5% correct matches, as per the paper.

So this is a summary of SIFT algorithm. For more details and understanding, reading the original paper is highly recommended. Remember one thing, this algorithm is patented. So this algorithm is included in Non-free module in OpenCV.

5.1 Comparison of Sift and Harris Corner Detection

Here we have shown the efficiency of descriptor from SIFT upon that of Harris Corner Detection in the next two images by matching the corresponding keypoints .It is clearly shown that SIFT gives more accurate matching , whereas we can see some mismatch in Harris corner descriptor because it is not invariant to scaling , it is only invariant on rotation. But SIFT is invariant of both scaling and rotation. Here are these..

Fig. 11. Matched keypoint using Harris corner for scaled images
We can see some mismatched keypoints bacause harris corner is not invariant to scale.

Fig. 12. Matched keypoint using Descriptors from SIFT
We can see more axccurate mismatched keypoints bacause SIFT is invariant to scale.

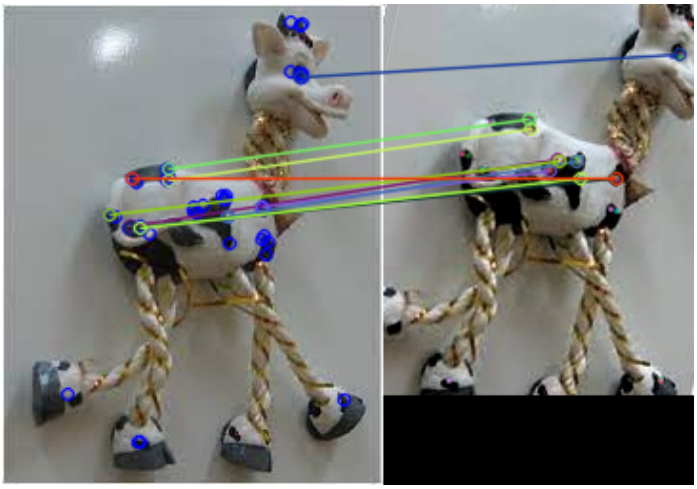


Fig. 13. Matched keypoint using Harris corner
Here also we can see some mismatched keypoints .

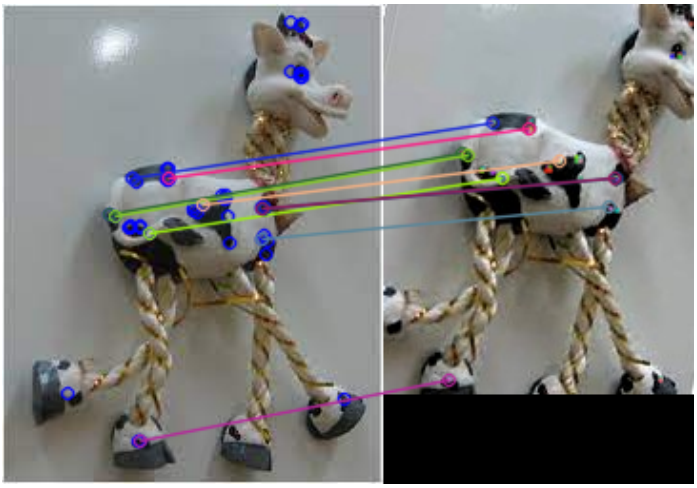


Fig. 14. Matched keypoint using Descriptor from SIFT
We can see more accurate matches.

6. Conclusion

This is a nice project to understand the basic difference between the descriptor of the key-point from the Haris corner detector and that from SIFT. We have get a clear conception after doing the project and implementing the theory knoledge. We have earlier discussed that if we use Harris corner detection method to detect the corners then taking thse corner we find a feature point and using these feature point we further match two picture. And in this case if the picture is translated or rotated then it performs well enough. But if the image which we want to match with another image is scalled then it does not perform well rather we have seen that there are many mismatched keypoints . Now in this case we definitely need the SIFT to get a good feature which can math the feature points from another image even if that is scaled along with rotation and translation, because we have also discussed that key point descriptor from the feature point gotten from SIFT algorith is scale invariant. So we in general try to use SIFT when we need a good match.

List of Figures

7. References

1. Computer Vision:Algorithms and Applications,Richard Szeliski
2. Digital Image Processing, 3rd edition Book by Rafael C. GONZALES and Richard E. Woods
3. Distinctive Image Features,from Scale-Invariant Keypoints,David G. Lowe,Computer Science Department,University of British Columbia,Vancouver, B.C., Canada,lowe@cs.ubc.ca
4. opencv python tutorial

5. https://www.southampton.ac.uk/ msn/book/new$_{d}emo/corners/$
6. http://datahacker.rs/opencv-harris-corner-detector-part1/
7. https://iitmcvg.github.io/summer$_{s}chool/Session3/$
8. http://www.cse.psu.edu/ rtc12/CSE486/lecture06.pdf
9. https://en.wikipedia.org/wiki/Scale-invariant$_{f}eature_{t}ransform$
10. https://www.youtube.com/watch?v=NPcMS49V5hgt=625s