Computer Vision: CSL7360
Course Project Report

Object Detection using SIFT

Instructors: Dr. Pratik Mazumder
and Dr. Rajendra Nagar

Team members
Lokesh Chaudhari (B21CS041)
Maithili Mangesh Borage (B21CS042)
Mohit Kumawat (B21CS046)
Shrashti Saraswat (B21CS081)

Contents

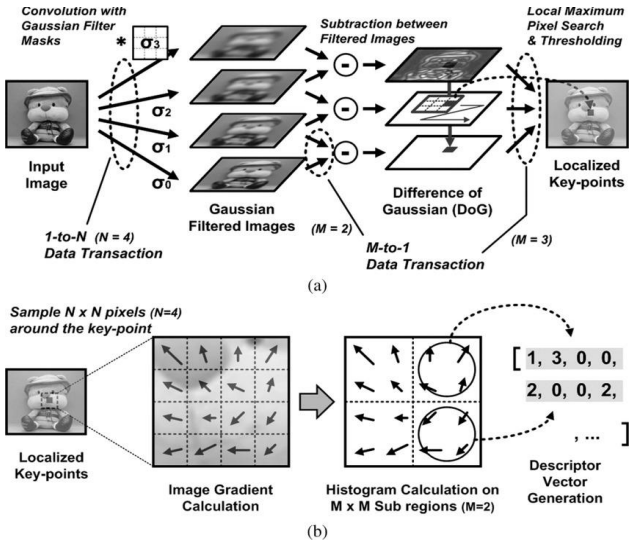
Problem Statement

Image matching is used in various computer vision applications in real life such as object or scene recognition, motion tracking etc. The process of image matching involves extracting image features from the images and comparing them using their properties. Hence, the image features need to be extracted from the images to carry out matching different objects and scenes. As the images can be of different scales, clicked at different angles and at different times of the day affecting the lighting, we need to make sure that the properties of features are invariant to changes in lighting, angles, scales etc.
The main task of the project is to implement Scale Invariant Feature Transform(SIFT) from scratch and understanding the mathematical approach behind image finding/identification and image matching. Other than this, template matching in the images have been implemented using which we can check if a given template is present in the image or not. Implementing SIFT involved various steps in which one of the main steps is identifying the keypoint identification and comparing keypoints in different images. Other than this, the similarity detection has been implemented on live images taken from camera.

Methodology

Implementation of SIFT algorithm involves majorly 4 steps. These four steps can be listed as follows:

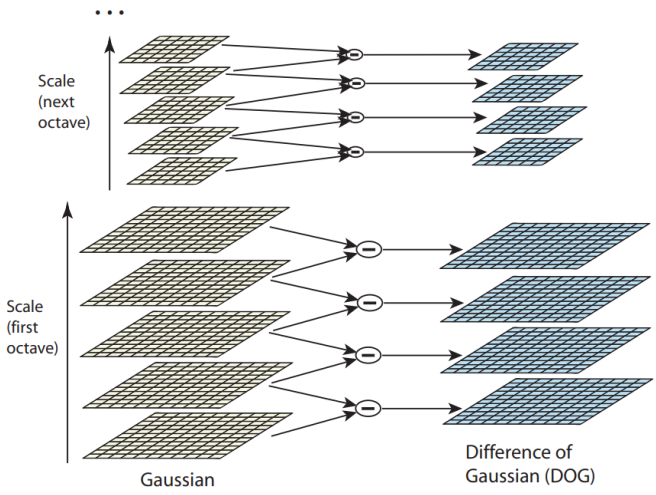
1. Scale-space extrema detection: The first stage of computation searches over all scales and image locations. Utilizing the difference of Gaussian (DoG) function, it efficiently locates potential interest points that are scale and orientation-invariant.
2. Keypoint Localisation: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
3. Orientation assignment: One or more orientations are assigned to each keypoint location based on local image gradient directions.
4. Keypoint Descriptor: The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.



Detection of scale-space extrema

The first stage of keypoint detection is to identify locations and scales that can be repeatably assigned under differing views of the same object, invariant to scale change of the image can be accomplished by searching for stable features across all possible scales, using a continuous function of scale known as scale space. Only possible scale-space kernel is the Gaussian function

$$G(x, y, \sigma) = \frac{1}{2\Pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$



To efficiently detect stable keypoint locations in scale space, use scale-space extrema in the difference-of-Gaussian function convolved with the image

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \qquad —(1)$$

Benefits of using DoG -

1. a particularly efficient function to compute
2. the difference-of-Gaussian function provides a close approximation to the scale-normalized Laplacian of Gaussian

- Local Extrema Detection
  In order to detect the local maxima and minima of D(x, y, σ), each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below.
- Frequency of Sampling in Scale
  The sampling frequency that maximizes extrema stability is experimentally determined. Repeatability does not continue to improve as more scales are sampled. The reason is that this results in many more local extrema being detected, but these extrema are on average less stable and therefore are less likely to be detected in the transformed image
- Frequency of Sampling in Spatial Domain (frequency of sampling in the image domain relative to the scale of smoothing)

If we pre-smooth the image before extrema detection, we are effectively discarding the highest spatial frequencies. Therefore, to make full use of the input, the image can be expanded to create more sample points than were present in the original.

Keypoint Localisation

The next step involved in the SIFT algorithm is Keypoint localisation. Once a keypoint has been found, we need to perform a detailed fit to the nearby data for location, scale and ratio of principal curvatures. A method needs to be implemented for fitting a 3D quadratic function to the local sample points to determine the interpolated location of maximum. This approach uses Taylor expansion of scale space function D()

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x$$

The location of extremum can be given by the following equation.

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}$$

The final offset ˆx is added to the location of its sample point to get the interpolated estimate for the location of the extremum. The function value at the extremum, D(ˆx), is useful for rejecting unstable extrema with low contrast.

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x}$$

For stability, it is not sufficient to reject keypoints with low contrast. The difference-ofGaussian function will have a strong response along edges, even if the location along the edge is poorly determined and therefore unstable to small amounts of noise. The principal curvatures can be computed from a 2x2 Hessian matrix, H, computed at the location and scale of the keypoint. The eigenvalues of H are proportional to the principal curvatures of D.

Orientation Assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. The scale of the keypoint is used to select the Gaussian smoothed image, L, with the closest scale, so that all computations are performed in a scale-invariant manne

Calculate - gradient magnitude m(x,y) and orientation θ(x, y)

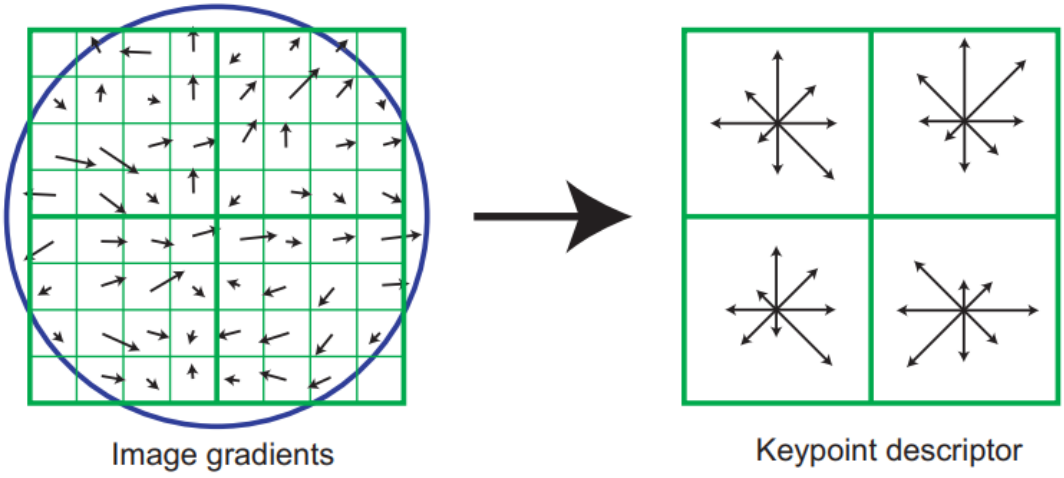
$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + \big(L(x,y+1) - L(x,y-1)\big)^2}$$

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

An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint. The orientation histogram has 36 bins covering the 360 degree range of orientations. Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest peak in the histogram is detected, and then any other local peak that is within 80% of the highest peak is used to also create a keypoint with that orientation.

Keypoint Descriptor

The descriptor for the local image region is highly distinctive yet is as invariant as possible to remaining variations, such as change in illumination or 3D viewpoint



A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window. Then, these samples are put together to make orientation histograms that show the contents over 4x4 subregions. The length of each arrow shows the sum of the gradient magnitudes near that direction in the region.

In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation. Then a Gaussian weighting function with σ equal to one half the width of the descriptor window is used to assign a weight to the magnitude of each sample point. The purpose of this Gaussian window is to avoid sudden changes in the descriptor with small changes in the position of the window, and to give less emphasis to gradients that are far from the center of the descriptor, as these are most affected by misregistration errors

Trilinear interpolation is used to distribute the value of each gradient sample into adjacent histogram bins. The descriptor is formed from a vector containing the values of all the orientation histogram entries

Finally, the feature vector is modified to reduce the effects of illumination change. First, the vector is normalized to unit length. A change in image contrast in which each pixel value is multiplied by a constant will multiply gradients by the same constant, so this contrast change will be canceled by vector normalization

Results

In our project, we have attempted to implement SIFT algorithm from scratch. Firstly, Gaussian filter and Derivative of Gaussian filter has been applied to the images to obtain an image pyramid.

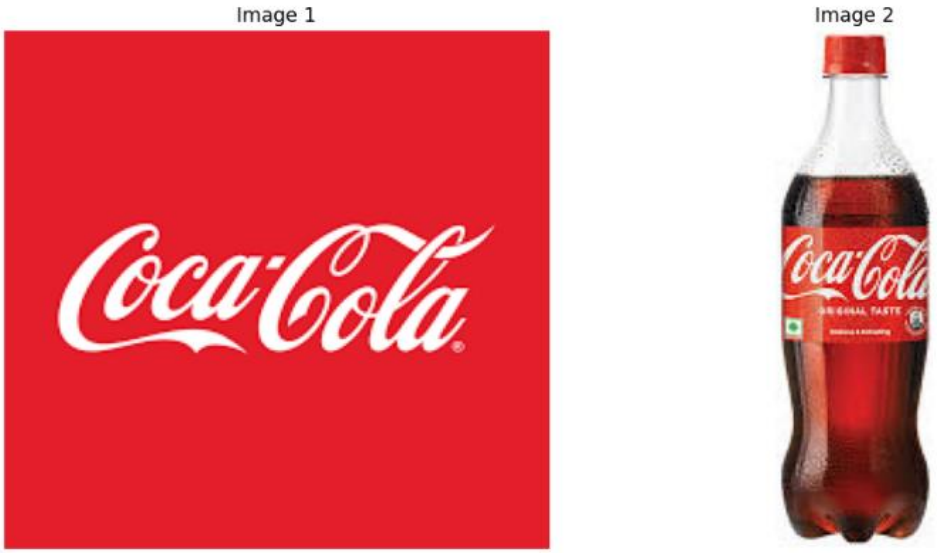


Image 1: Original images used for testing the implementation of SIFT

Image 2: Gaussian Filter applied on the original Image

Image 3:Gaussian and DoG filter applied on the original ikmage to obtain image pyramid

After implementing the SIFT algorithm on above preprocessed images, we can identify the number of key points in both the images as follows:

```
Number of keypoints in image 1 with SIFT from scratch :  95
Number of keypoints in image 2 with SIFT from scratch :  98
```

Image 4: Key points detected in both the images implementing SIFT from scratch

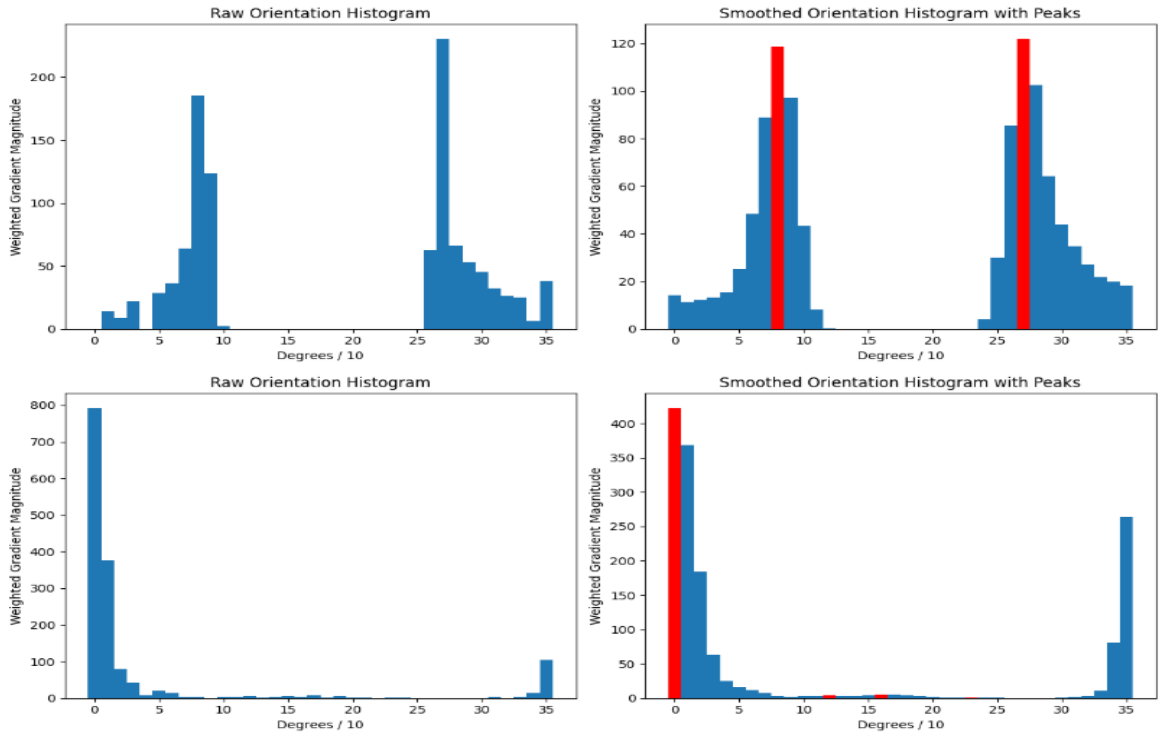


Image 5: Orientation Histograms

Visual representation of keypoints identified on both the images

Image 6: Keypoints identified in the original image

Number of matches with SIFT implementation from scratch here is obtained to be 13.

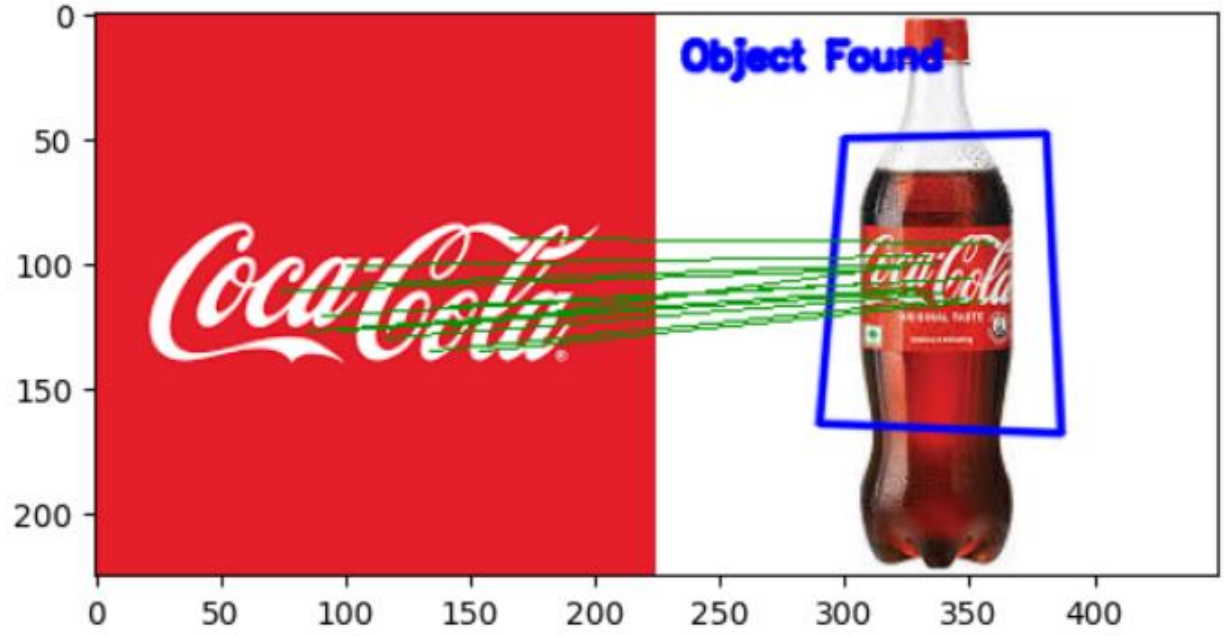


Image 7: Template matching using the images

The Results of OpenCV implementation of SIFT algorithm.

Image 8: Keypoint matching in Open CV

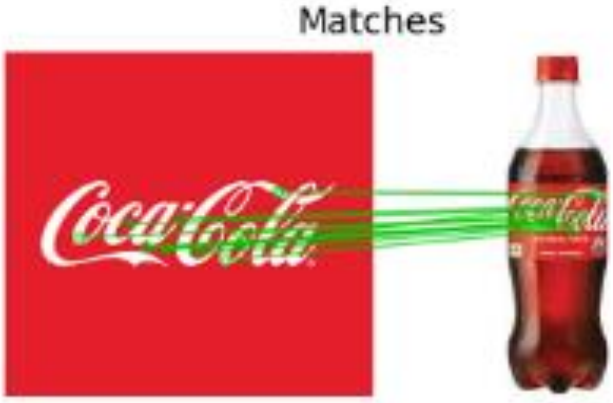


Image 9: Template matching using the original images in Open CV

```
Number of keypoints in image 1 with OpenCV :  89
Number of keypoints in image 2 with OpenCV :  101
Number of Matches with OpenCV :  13
```

Image 10: Number of keypoints identified in both images using Open CV

Observation and Conclusion

When comparing the SIFT implementations made from start and those made with OpenCV, the results for finding and matching keypoints were the same. Even though there were small differences in the number of keypoints found, both methods gave about the same number of matches between the two images. The fact that it stays the same shows that the SIFT algorithm that was built from scratch is strong and can do jobs like keypoint detection and matching as accurately as well-known libraries like OpenCV. In conclusion,

the successful comparison proves that the applied SIFT algorithm works and shows that it can be used in real-world computer vision tasks.
The custom SIFT algorithm, implemented from scratch, exhibited significantly longer execution times compared to the OpenCV implementation.
The OpenCV implementation typically runs faster due to optimizations and possibly parallel processing capabilities.

References

1. Research Paper: https://people.eecs.berkeley.edu/~malik/cs294/lowe-ijcv04.pdf
2. https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/
3. https://www.naukri.com/code360/library/scale-invariant-feature-transform-sift
4. https://link.springer.com/chapter/10.1007/978-3-319-01796-9_7