# Implementation of Scale Invariant Feature Tranform (SIFT)

Ajay Ramesh[1] and Chandra Kumar Basavaraju[2]

*Abstract*— **More often, a Computer Vision problem involves Image Matching whether it is Object recognition, Object tracking to name a few. These tasks are interesting but difficult as we have deal with transformed images of the scene. Transformation may include scaling, rotation etc. Having to detect the target object in the scene irrespective of these transformations is the key in these tasks. David G. Lowe in his paper[1] has proposed an algorithm, which he has got patented against his name, to extract features from images such that the extracted features are invariant to Scaling, Rotation and to some extent to illumination and 3D camera viewpoint. In his paper[1], David G. Lowe mentions that the features extracted from his algorithm can be used for reliable matching between different views of an object or scene. In the paper, he also mentions that these features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for task of Object and scene recognition.**

**Inspired by his work in the paper[1], we as part of our academic project for the Computer Vision (CS512) course at Illinois Institute of Technology, worked on implementing the algorithm, which we are presenting in this paper.**

In order to explain the problem consider the figure 1, where Harris corner detection fails[2] to detect the corner once the image is scaled. From the fig 1 it is clear that corner looks like curve when it is scaled. So we can't use Harris corner detection to it. We need scale invariant features to detect the objects. So we implemented SIFT[1], next paper is organized as -

*This work was done as part of the academic project for the course CS512 - Topics in COmputer Vision at Illinois Institute of Technology

[1]Ajay Ramesh is a Masters Student in Computer Science department at Illinois Institute of Technology, Chicago . Demo video is present at https://youtu.be/wcaPH01GlHA aramesh6 at hawk.iit.edu

[2]Chandra Kumar Basavaraju is a Masters Student in Computer Science department at Illinois Institute of Technology, Chicago. cbasavaraju at hawk.iit.edu

- Implementation & Explanation of SIFT algorithm in Section I
- Experiments results in Section II
- Conclusion
- Future work

## I. Algorithm & Implementation Details

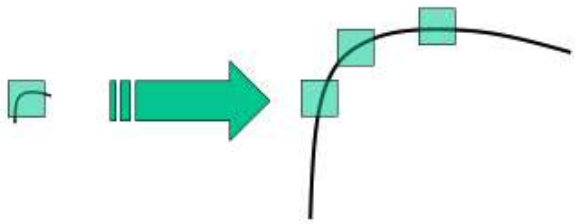
There are four major steps in the algorithm:

- Scale Space Extrema Detection
- Keypoint Localization
- Orientation Assignment
- Keypoint Descriptor.

The algorithm takes Cascade Filetring aproach to minimize the computation cost in extracting the features, where the expensive operations are applied only at locations which pass an initial test.

We briefly explain each of these steps in this section.

Fig. 1. Corner detection fails



### A. *Scale-space Extrema Detection*

In the first stage of the algorithm we will detect keypoints using a cascade filtering approach that uses efficient algorithms to identify candidate locations that are then examined in further detail. The goal is to identify keypoints that can be repeatedly assigned under differing views of the same object. Detecting location that are invariant to scale can be accomplished by searching for

stable features across all possible scales using a continuous function of scale known as scale space.

The scale space kernel used for this task is the Gaussian function. So, the scale space of image is produced from the convolution of a variable scale Gaussian, with an input image I(x, y).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (1)$$

To efficiently detect stable key points locations in scale space, we use scale space extrema in the Difference Of Gaussian function convolved with the image,

$$D(x, y, sigma) \quad (2)$$

, which can be computed from the difference of two nearby scales separated by a constant multiplication factor k:

Fig. 2. .

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$
$$= L(x, y, k\sigma) - L(x, y, \sigma).$$

An efficient approach to construction of DoG is shown in Fig 3. The initial image is incrementally convolved with Gaussian to produce images separated by a constant factor k in scale space, shown stacked in the left column. Adjacent images are subtracted to produce DoG images hown in the right. Once a complete octave has been processed, we resample the Guassian image.

In order to detect the local maxima and minima of DoG, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below as shown in Fig 4. An extrema is detected only if it is larger than or smaller than all the neighbors.To make full sense of the input, the initial image is doubled using linear interpolation prior to building the first level of the pyramids.

### *B. Keypoint Localization*

Once a keypoint candidate has been found by comparing a pixel to its neighbors, the next step is to perform a detailed fit to the nearby data for location, scale and ratio of principle curvatures. This information allows points to be rejected that have low contrast (and are therfore sensitive to noise) or poorly localized along an edge.

Fig. 3. Scale space Extrema Detection

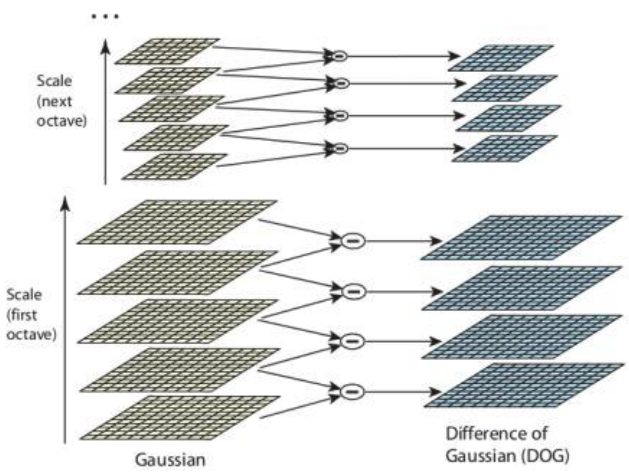
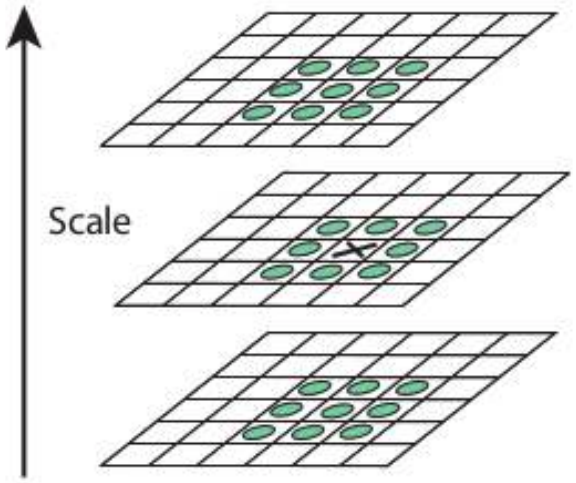



Fig. 4. Maxima and Minima of the DOG




To detect the interpolated location of the maximum, we use Taylor expansion of the scale space function, the DoG, shifted so that the origin is at the sample point:

where D and its derivatives are evaluated at the sample point and x = (x, y, sigma)T is the offset from this point. The location of the extremum, x_hat, is determined by taking the derivative of this function with respect to x and setting it to zero, giving As suggested by Brown, the Hessian and derivative of D are approximated by using differences of neighboring sample points. The resulting 3 by 3 linear system can be solved with minimal cost. If the offset x_hat is larger than 0.5 in any dimension, then it means that the

Fig. 5. LOG and Extremas

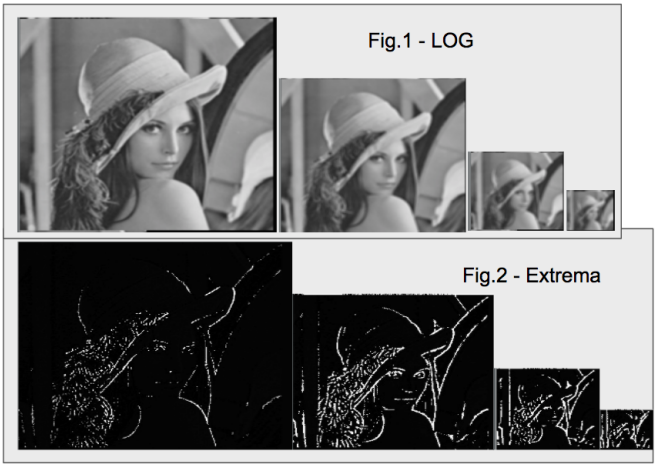



Fig. 6. Result

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

extrema lies closer to a different sample point. In this case, the sample point is changed and the interpolation performed instead about that point. The final offset x is added to the location of its sample point to get the interpolated estimate for the location of the extrema. The function value at the extrema, D(x_hat), is useful for rejecting unstable extrema with low contrast. This can be obtained by substituting Equation in Fig 7. into equation in Fig 6., giving equation in Fig 12.

A poorly defined peak in the difference-of-Gaussian function will have a large principal curvature across the edge but a small one in the perpendicular direction. The principal curvatures can be computed from a 2X2 Hessian matrix, H (shown in Fig 8.), computed at the location and scale of the keypoint. The derivatives are estimated by taking differences of neighboring sample points.

we can avoid explicitly computing the eigenvalues, as we are only concerned with their ratio. Let be the eigenvalue with the largest magnitude and be the smaller one. Then, we can compute the sum of the eigenvalues from the trace ofHand their product from the determinant as shown in Fig

Fig. 7.

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

Fig. 8.

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

10.

Fig. 9.

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}.$$

to check that the ratio of principal curvatures is below some threshold, r, we only need to check the equation shown in Fig 9.

Fig. 10.

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$
$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

### C. *Orientation Assignment*

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented to this orientation and therefore achieve invariance to image rotation.

Fig. 11.

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

Fig. 12.

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

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 manner. For each image sample, L(x, y), at this scale, the gradient magnitude, m(x, y), and orientation, (x, y), is precomputed using pixel differences. Refer Fig 12 for calculating magnitude and orientations.

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. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a that is 1.5 times that of the scale of the keypoint.

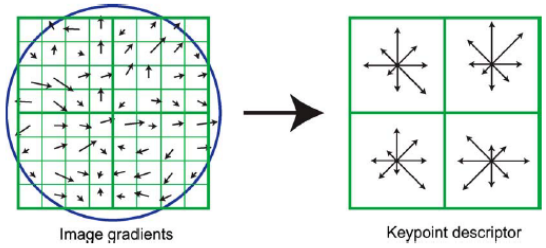
Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest peak in the histogram is detected, and then any other local peak that is within 80% of the highest peak is used to also create a keypoint with that orientation. Therefore, for locations with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale but different orientations.

Fig 5 shows 4-octave and extreamas calcuated from the steps 1-3.

### *D. Keypoint Descriptor*

Now keypoint descriptor is created. A 16x16 neighbourhood around the keypoint is taken. It is divided 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. The descriptor is normalized to unit length. Then each value in the descriptor greater than 0.2 is set to 0 and the descriptor is renormalized to unit length. These steps helps to achieve robustness against illumination changes, rotation etc.

Fig. 13.




First the image gradient magnitudes and orientations are sampled around the keypoint location, using the scale of the keypoint to select the level of Gaussian blur for the image. In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation. For efficiency, the gradients are precomputed for all levels of the pyramid using central difference of the pixels. These are illustrated with small arrows at each sample location on the left side of Fig 9. A Gaussian weighting function with  equal to one half the width of the descriptor window is used to assign a weight to the magnitude of each sample point. This is illustrated with a circular window on the left side of Fig 9. although, of course, the weight falls off smoothly. 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. The keypoint descriptor is shown on the right side of the Fig 9. It allows for significant shift in gradient positions by creating orientation histograms over 4x4 sample regions. The figure shows eight directions for each orientation histogram, with the length of each arrow corresponding to the magnitude of that histogram entry. A gradient sample on the left can shift up to 4 sample positions while still contributing to the samehistogram on the right, thereby achieving the objective of allowing for larger local positional shifts.

## II. Experiments and Results

We have used the parameters determined to best in the paper. Those are listed below-

1) K = 1.3
2) sigma = 1.6
3) threshold = 0.003
4) Number of octaves = 4
5) Number of scales per octave = 6

We tested for 5 kinds of image, such as car, object, scene and person. We tested our implementation across ORD implemention[3] of opencv and sift implementation from opencv. We see that our algorithm needs much more optimization due to that we are detecting little more keypoints than other implementation. Please refer 14 for the comparision of our implementation to the original SIFT and ORD algorithms implemented in OpenCV.

Fig. 14. Final Output sample against other implementation

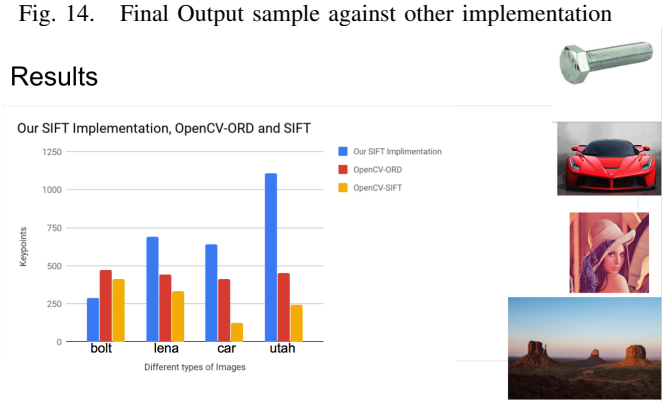
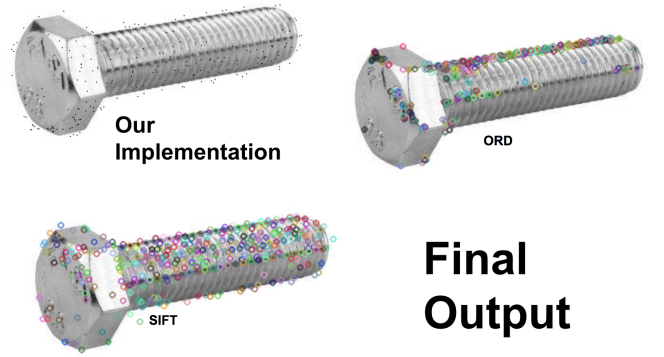



Fig. 15. Comparison




## III. Conclusion and Future Work

### A. *Conclusion*

Our implementation even though is finding more keypoints than the original implementation of SIFT, is performing well and is comparable to the original implementation in terms of speed and accuracy.

By working on this project, we understood the SIFT algorithm very well and effectiveness of the keypoint descriptors extracted from this method. These keypoints are distinctive and are very useful and reliable for object matching across different scale, rotation. The fact that the keypoints are detected over different scales means that small local features are available for matching small and highly occluded objects, while large keypoints perform well for images subject to noise and blur.

The algorithm is efficient as it uses the Cascade filtering approach i.e, performs expensive operations over locations which pass the initial test.

### B. *Future Work*

1) We could achieve better performance by Optimizing our implementation.
2) Matching across a large database would be helpful.
3) We have not handled corner cases in our implementation, which we have mentioned at places in our source code as comments. Handles those scenarios can be our one area to work on in future.
4) Conversion from python to cpython/c code


## Acknowledgment

Our sincere thanks to our professor Gady Agam, for giving us an opportunity to do this project. We also would like to thank our department and the university for providing the infrastructure for the study of this project. Gratitude to David G. Lowe whose papers and idea we have used in this project.


## List of Contribution by each individuals

We worked together and periodically reviewed each others work.

Ajay Ramesh -

- Keypoint localization
- Keypoint description
- PPT
- Analysis

Chandra Kumar

- Scale space extrema detection
- Orientation assignment

- Report
- Code Review

## REFERENCES


[1] Lowe, D.G. International Journal of Computer Vision (2004) 60: 91. doi.org/10.1023/B:VISI.0000029664.99615.94

[2] OpenCV Introduction to SIFT (Scale-Invariant Feature Transform)

[3] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 2564-2571. doi: 10.1109/ICCV.2011.6126544

[4] UCF Computer Vision Video Lectures 2012 Instructor: Dr. Mubarak Shah , Subject: Scale-invariant Feature Transform (SIFT)

[5] en.wikipedia.org/wiki/Lenna - Lena.jpg is used for demo.

[6] www.learnopencv.com/histogram-of-oriented-gradients/