University of Hamburg

The Cognitive Systems Laboratory (KOGS)

Masters project part 1

Implementation of Scale Invariant Feature Transform (SIFT)

Combined report by:
Banik Soubarna, Kulkarni Praveen
4banik,4kulkarn@informatik.uni-hamburg.de

May 6, 2016

1 SIFT Implementation

1.1 Introduction

SIFT (Scale Invariant Feature Transform) feature extractor, proposed by David Lowe in [Low04] is widely used for object recognition. In order to match features between objects it is prudent to extract features that are invariant to any translation, rotation, scaling or any change of viewing direction. The previous feature detectors, such as Harris Corner detector are fallible to scale variations. A corner detected at a particular scale, may not be considered as a corner if the image is zoomed in or in other words, if the scale is decreased. SIFT detector can detect the corners irrespective of any scale change. This is why it is becoming a salient choice for feature detection in computer vision.
This project report describes an implementation of SIFT feature detector and descriptor. The implementation is done in C++ using VIGRA library, which is an image processing and analysis library. The report structure is as follows. Section 1.2 starts with an overview of the SIFT algorithm and discusses the keypoint detection and descriptor extraction methods in sections 1.2.1 and 1.2.2 in detail respectively.

1.2 Algorithm

The SIFT algorithm, proposed by Lowe [Low04], has two phases - Keypoint detection and Descriptor extraction. The objective of this algorithm is to find stable interest points across all possible scales. The algorithm detects the points having extreme (maximum/minimum) gradient in the neighborhood and considers it as interest points after passing few elimination criterion.

1.2.1 Keypoint Detector

The interest points or keypoints are blob-like structures associated with some orientation. There are mainly three phases in keypoint detection, which are described in the following sections.

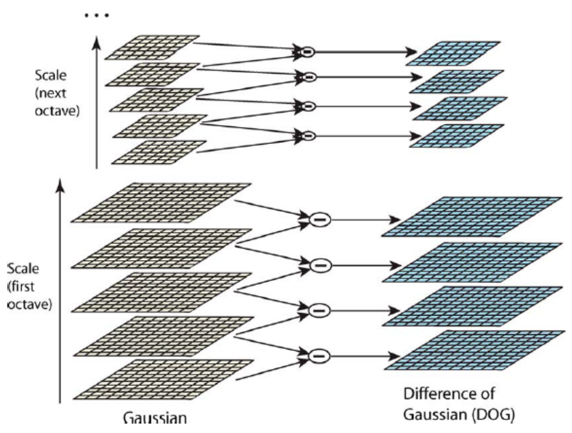

Figure 1.1: SIFT Keypoint Detection [Low04]

Scale Space Extrema Detection

Features are subjective to its neighborhood window. A small corner can be detected using a small window. However, a small window would not be able to detect a large corner - a window large enough to hold the corner entirely is needed to detect it.

Building Scale Space: To detect scale invariant features, the input image is smoothed with windows of different width or scales. The range of scales is called scale space. As described in [Low04] this is achieved by convolving the image intensity function, $I(x, y)$ with Gaussians, $G(x, y, \sigma)$ over the scale space, where σ denotes the scale. In other words, Laplacian of Gaussian (LoG) of the image is computed. LoG acts a blob detector, which detects blobs in various sizes signified by the σ. For, better performance, in addition to variation of scales, the image is sub-sampled and the same smoothing operation is performed. This gives an effect of doubling the size of the smoothing window. This is repeated for a number of octaves. After each octave the image is sub-sampled to half of its size. This is explained in the Figure 1.1.
In our implementation the method ***build_gauss_pyr*** computes the scale space pyramid of the image.The number of octaves is determined from the image size - $\log_2(\min(width, height))$. Number of intermediate scales, s used in each octave is by default set to 3. The scale is increased by a constant factor $k = 2^{1/s}$ to keep the number of layers in an octave, an integer. For detecting an extrema, three consecutive layers are compared. Hence, in each octave, $s + 3$ numbers of layers are computed, to cover all s layers.
In Lowe's method, the input image is doubled initially for increasing the number of keypoints. For simplification, we have omitted this step.

LoG Approximation: LoG is an expensive operation and it can be approximated by using the Difference of Gaussian (DoG) operator as explained in Figure 1.3.

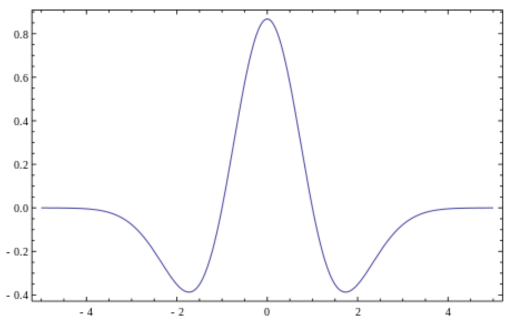

Figure 1.2: Laplacian of Gaussian (LOG) Operator in 1D

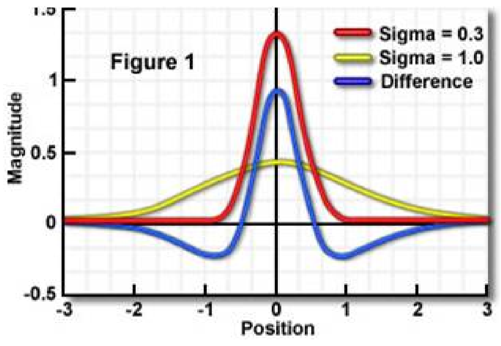

Figure 1.3: Approximation of LOG - Difference of Gaussian Operator in 1D

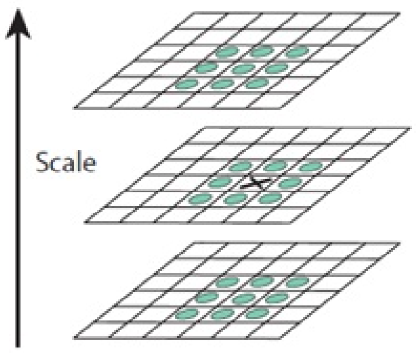

Figure 1.4: Detecting Extrema [Low04]

Once the scale space pyramid is built, successive layers are subtracted to form the Difference of Gaussian pyramid. The method *build_dog_pyr* generates this DoG pyramid.

Detecting Extrema: Each pixel is compared with its immediate neighbors in spatial as well as scale dimension in the DoG pyramid to find out the maxima and minima. As illustrated in 1.4, each pixel in each image from the DoG pyramid is compared to its 26 neighbors. This is implemented in method *is_extremum*.

Keypoint Localization

The extrema obtained are actually approximate results as the maxima/minima almost never lies exactly on a pixel. Due to this, Lowe interpolated the pixels mathematically using Taylor's expansion of the scale-space function, $D(x, y, \sigma)$ or $D(\mathbf{x})$ to "fit to the nearby data for location, scale, and ratio of principal curvatures" [Low04]. By solving the

Taylor expansion of $D(\mathbf{x})$, the extreme points or the extreme offsets can be computed. In our implementation, this is directly calculated from the equation 1.1 [Low04] using the methods *compute_pderivative*, *compute_hessian* and *interpolate_extremum*.

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial x} \tag{1.1}$$

If the offset determined is greater than 0.5, the coordinate of the extrema is updated by adding the offset. This interpolation process is repeated for certain times. After the final offset is obtained, the intensity at the new position is compared with a threshold and the point is rejected if it is less than the threshold.
As LoG is mainly an edge detector, a lot of keypoints are detected at edges, which need to be eliminated. After the contrast threshold check, the interpolated extrema is also tested for edge criterion, which rejects the point if the curvature around the point does not satisfy the edge threshold value. The contrast threshold and edge threshold values are set to 0.04 and 10 respectively, according to the original implementation of Lowe.

Orientation Assignment

Once a keypoint is detected, its orientation is computed to achieve rotation in-variance. For determining the orientation, a neighborhood window is considered whose size is proportional to the scale of the corresponding keypoint. The magnitude of the intensity gradient and direction are calculated for all neighboring points and a histogram of orientation is built with 36 bins representing the 360 degrees directions around the keypoint. All orientations are weighted by their magnitude and the scale of the keypoint. The peak orientation in the histogram is the dominant orientation of the keypoint. This is considered as a feature. Any peak above 80% of the highest peak is also considered as a feature and it is converted into a new keypoint with same coordinate, scale but with the corresponding orientation. Hence, multiple keypoints can be extracted at the same location.

1.2.2 Keypoint Descriptor

At each key-point as discussed above an image descriptor is computed. Previous steps found key-points with at particular scales and assigned orientations to them. This made the features invariant to location, scale and rotation. Now we need to describe these key points such that they are highly distinctive and partially invariant to other variations like illumination, viewpoint etc. Based on the scale of keypoints detected earlier the respective image at the same scale is used to derive the descriptors for the key point.

As per the SIFT implementation in [Low04] the SIFT descriptor is computed using 16x16 pixel patch of an image at corresponding scale of key point. As shown in figure 1.5 first gradient magnitude and orientation is calculated at every sample around the key point. Then these gradients are weighted using Gaussian as represented by circle overlaid on

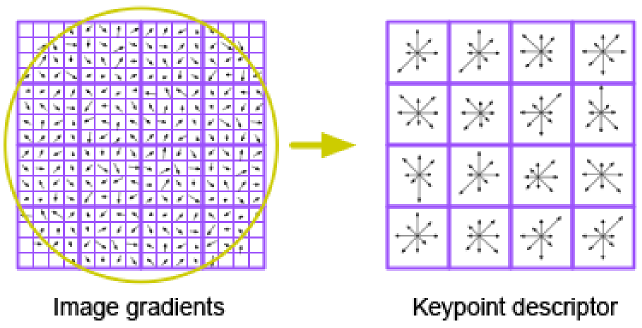

Figure 1.5: SIFT Keypoint descriptor [Low04]

left image. Then a small sub-patch of size 4x4 is used to build histogram of size eight representing eight orientations of gradients. This can be seen in the right image.

Descriptor parameters

The SIFT descriptor is basically a weighted and interpolated histogram of the gradient orientations of sub-patches surrounding the key point. The parameters for SIFT descriptor are as below:

- Magnification factor (m): The size of sub patch based on which the histograms are computed has size $m\sigma$. Where σ is the scale of frame in which the key point resides. In the current implementation the value for m is defaulted to 3.0.
- Number of bins for orientation: Number of bins for orientation histogram. The gradients in 4x4 sub patches around key point are binned based on orientation along eight directions.
- Size of the the descriptor: For 16x16 image patch there are 4x4 sub patches. In each of this patch histogram with 8 bins is computed. So size of the descriptor is equal to $(numberofspatialbins)^2 * (numberofbinsfororientation)$. Which in our case is equal to 128 $(4 * 4 * 8)$.
- Threshold on magnitude of gradient: This parameter is set to 0.2 by default and helps in avoiding more emphasis being given to high contrast measurements.

Position, scale and rotation invariance

In-variance to some extent is achieved because in the process described above, the histogram layout gets projected on image domain according to the frame of reference of the key point. Also, the spatial binning happens in scale space of key point, and finally, the layout is rotated such that the layout is aligned to the direction θ of a key point.

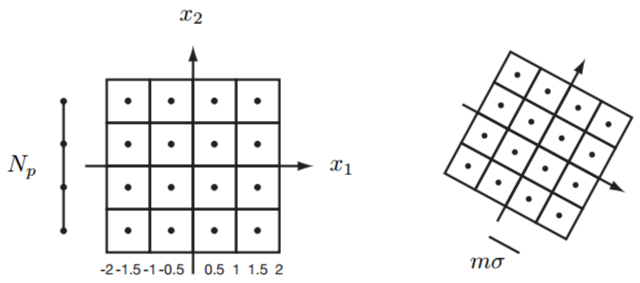

Figure 1.6: SIFT descriptor layout

Figure 1.6 the transformations that SIFT descriptor layout undergoes in order to align key point position, orientation and scale.

Contrast normalization

The weighting of gradients as seen above before binning them into histograms helps achieve contrast normalization and makes descriptor robust under illumination variations. Furthermore, the threshold of 0.2 is used to limit high contrast gradient measurement to be given more emphasis.

Tri-linear interpolation

The accuracy of local histograms can be improved by tri-linear interpolation. It helps to distribute the weighted gradients of a sample in a patch into adjacent histogram bins. This is very important in matching phase where the matching performance is better for viewpoint changes and rotation of key points in the new frame

1.3 Result and Interpretation

Sample input and output of key point detection algorithm is shown in Figure 1.7 and 1.8. The algorithm was tested for rotation and scale invariance. The input image was rotated by 30 degree and the result can be seen in Figure 1.9 and 1.10. Figure 1.11 and 1.12 shows the output for the scaled down image.

The number of keypoints detected varied in the three scenarios.
Number of keypoints detected for normal image: 6592
Number of keypoints detected for rotated image: 13242
Number of keypoints detected for scaled image: 885
The increase of the number of keypoints for the rotated image is caused by the size of

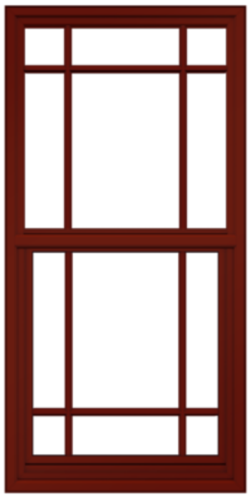
Figure 1.7: Sample Input Image

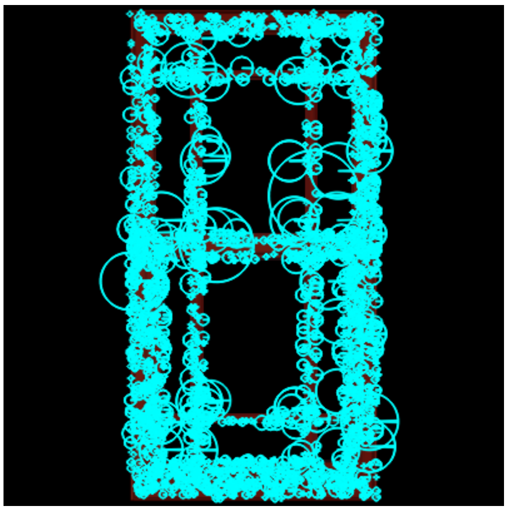
Figure 1.8: Output Image

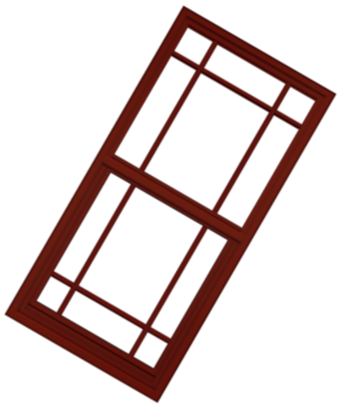
Figure 1.9: Input Image - Rotated by 30 degree

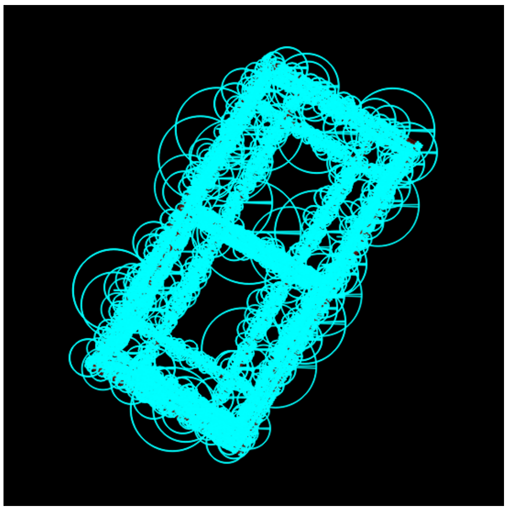
Figure 1.10: Output Image - Rotation Invariance

the image, which changed when the image was rotated. As the number of octaves is dependent on the size of the image, the rotated image was processed for more number of octaves than the normal image. Hence, the difference.

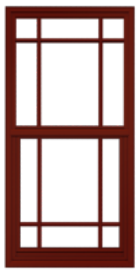

Figure 1.11: Input Image - Scaled down by 50%

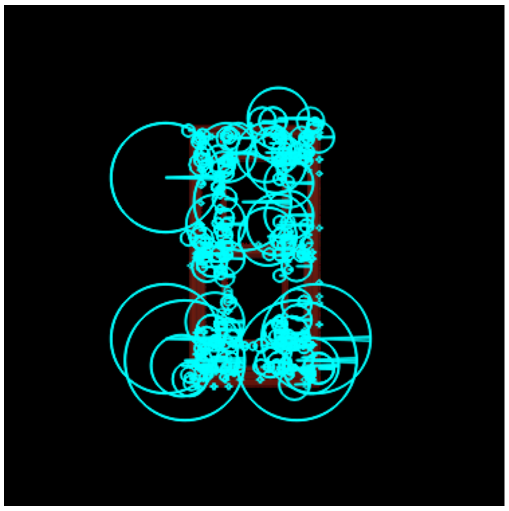

Figure 1.12: Output Image - Scale Invariance

1.4 Summary

Below we will summarize some of the aspects of SIFT that we have implemented with c++, Vigra and Eigen:

- Finding scale space extrema:
  1. Building Laplacian of Gaussian pyramid
  2. Building Difference of Gaussian pyramid
  3. Detecting extrema points.
- Key point Localization:
  1. Interpolation of Keypoints
- Keypoint filtering:
  1. Rejecting low contrast points
  2. Rejecting points with strong edge response
- Remove effects of rotation and scale
  1. Binning in respective scale space of key point
  2. Rotating SIFT descriptor layout to align with orientation of key point
  3. Create gradient histogram for 8 direction weighted by Gaussian window
  4. Directions selection based on highest peak

- Calculate SIFT descriptor
    1. Weighting gradients with Gaussian window centered at key point
    2. Threshold selection to limit high contrast gradients
    3. Re-normalize after threshold selection
    4. Tri-linear interpolation to improve accuracy of local histogram

1.5 Future Work

This project implements the basic algorithm of SIFT and there is scope for improvement. Currently the keypoint detection method is processing every pixel of all the images in the DoG pyramid and this is very time consuming. Also, our method is returning a lot of redundant keypoints. These processes can be optimized by scanning the images in parallel. For now the implementation is done only for gray scale UInt8 images. However this can be easily changed to be made more generic and handle both UInt8 and float images. We excluded the initial step of doubling the image and re-adjusting it at the end. This can be included in future versions.

Bibliography

[Low04] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.