Implementing the SIFT Operator

Venkata Sasank Mudigonda

I. INTRODUCTION

The Scale Invariant Feature Transform (SIFT) operator provides a robust feature descriptor that is invariant to changes in scale, translation, rotation, and partially to changes in illumination and 3D viewpoint. The extracted features are highly distinctive, enabling accurate (with high probability) comparison with a large database of features. The cost of feature extraction is minimized by the cascaded filtering approach (output of a stage is fed as input to the succeeding stage) proposed by the author. The features that do not match the criteria are discarded at each stage of the extraction. This results in reduced amount of computation due to reduction in the number features moving forward through the stages. The major stages are listed below.

1) **Scale-space extrema detection:** In the first stage, we search over various scales of the input image and identify the interest points using *Difference of Gaussians* (DoG).
2) **Keypoint Localization:** The keypoints are idenfitifed by locating the points that are the extrema (either maximum or minimum) in their space and scale neighborhoods. Later we fit 3D-quadratic function to determine the interpolated extrema.
3) **Orientation Assignment:** Each keypoint is assigned with multiple dominant gradients identified in its neighborhood using *Histogram of gradients* (HoG).
4) **Keypoint Descriptor:** The local gradients are identified at appropriate scale around each keypoint. These are normalized and encoded into a vector representation giving us feature descriptors.

II. RELATED WORK

The problem of identifying good features is a subset of a bigger problem called the *image matching problem*. The development of solutions to image matching is first introduced by Moravec, 1981 in his work on stereo matching using corner detector in [7]. Later Harris, 1992, in his seminal work, introduced corner detector in [8], which has been very widely used in image matching space. Although in popular literature, it is addressed as corner detector, it can detect any location that has large gradients in all directions at a predetermined scale. The Harris-corner operator is very sensitive to scale changes. Prior work on identifying rotationally invariant feature descriptors by Schmid and Mohr, 1997 in [9] is a motivation for identifying features descriptors that are scale-invariant. This allowed them to match features under arbitrary orientation changes between two images. In his earlier work in [10], the author tried to extend the local feature approach in [9] to achieve scale-invariance. In this paper ([1]), he introduced a method that significantly improved invariance to scale and affine transformations.

III. SCALE-SPACE EXTREMA DETECTION

A. Background

A pile or prior research on scale-invariant detection, the use of Gaussian filters greatly enabled the idea of the SIFT operator. Witkin in [2] showed that detecting locations that are scale-invariant can be accomplished by search for stable features across all possible scales using a continuous function called *scale-space*. Koenderink, 1984 in [3] and Lindeberg, 1994 in [4] showed that under certain reasonable assumptions, the Gaussian function is the only possible scale-space kernel. Lindberg, 1994 in [4] also showed that normalization of the Laplacian of Gaussian (LoG), $\nabla^2 G$ with the factor σ^2 is required for true scale-invariance.

B. Understanding The Mathematics Of The Filters

The image in the scale-space, $L(x, y, \sigma)$ is characterized by σ, which is produced by convolving the original image, $I(x, y)$ with the Gaussian kernel, $G(x, y, \sigma)$.

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

where $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \cdot e^{\frac{-x^2-y^2}{2\sigma^2}}$, x and y represent the coordinate space of the image.

Now, instead of using scale-normalized LoG as the filter, which is computationally expensive with the exponent and multi-level derivatives for each pixel, the author proposed the use of *Difference of Gaussian* (DoG), $D(x, y, \sigma)$, which is efficient to compute. The DoG is the pixel-level difference between two Gaussian convolved images from adjacent scales separated by a constant factor, k.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, k\sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{1}$$

The DoG operator is an efficient approximation of LoG is it involves a simple image subtraction operation. It's also a close approximation to scale-normalized LoG (see Appendix section A for the proof of the relation between DoG, $\frac{\partial G}{\partial \sigma}$ and LoG, $\nabla^2 G$). From the heat diffusion equation, we get

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$

using finite difference approximation, we get,

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

$$D(x, y, \sigma) = G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G$$

C. Process Of Computing DoG

Before computing DoG, we need to create groups of images called *octaves*. Each octave consists of a groups of images, where each image is produced by convolving a Gaussian filter of specific σ with its previous image so that the last image in the octave has twice the blur (2σ) of the first one (σ). We start by specifying the number of images an octave must contain by s and the first image is convolved with $k\sigma$, where $k = 2^{1/s}$. After s images, the blur becomes 2σ. In addition to the s images, we add 3 more so that the extrema detection covers the first and the last image in the octave.

DoG is computed by performing image subtraction between adjacent images. We construct 4 such adjacent octaves with incremental σ values. Fig 1 shows the DoG output of an image.

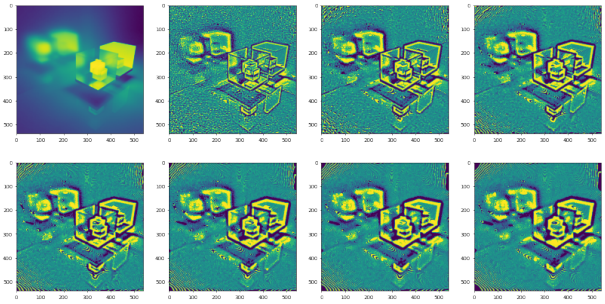

Fig. 1: Computing DoG from octaves

D. Identifying Local Extrema

To identify the local maxima and minima in each image in DoG, each sample point is compared to it 8 neighbors within the image and 9 neighbors in the upper scale and also in the lower scale. We need $s+2$ images so that the extra 2 images are placed as boundaries of an octave to allow us to scan the extremas in upper-most and lower-most images in the octave.

IV. Keypoint Localization

Although, keypoints are idenfitied as extrema in pixel location is a straight-forward way to do so, Brown and Lowe, 2002 in [5] developed a method to fit a 3D quadratic function to the local sample points to determine the interpolated location of the extrema. Their method has been proven to substantially improve matching and stability. They used the Taylor expansion (up to quadratic terms of) the scale-space function, $D(x, y, \sigma)$, shifted so that the detected sample point is at the origin. In other words, due to discretization of the pixel locations and available scales (σ), it is possible that the extrema lies between adjacent scales and inter-pixel locations. To locate them accurately, we use interpolation with Taylor expansion of the scale-space function, $D(x, y, \sigma)$ and accurately locate them. The Taylor expansion of $D(x, y, \sigma)$ is given below.

$$D(\mathbf{x_0} + \mathbf{\Delta x}) = D(\mathbf{x_0}) + \frac{\partial D^T}{\partial \mathbf{x}}\bigg|_{\mathbf{x_0}} \mathbf{\Delta x} + \frac{1}{2}\mathbf{\Delta x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\bigg|_{\mathbf{x_0}} \mathbf{\Delta x}$$

where $\mathbf{x_0} = (x_0, y_0, \sigma_0)^T$, $\mathbf{\Delta x} = (\delta x, \delta y, \delta\sigma)$ and $\mathbf{x} = \mathbf{x_0} + \mathbf{\Delta x}$. The location of the extremum, $\hat{\mathbf{x}}$ is determined by taking the derivative of the function with respect to $\mathbf{x}$ and setting it to 0.

$$\hat{\mathbf{x}} = -\left(\frac{\partial^2 D}{\partial \mathbf{x}^2}\bigg|_{\mathbf{x_0}}\right)^{-1} \frac{\partial D}{\partial \mathbf{x}}\bigg|_{\mathbf{x_0}}$$

To compute $\hat{\mathbf{x}}$, we use finite difference approximation. This in turn, involves computing the Jacobian ($\mathbf{J}$) and the Hessian ($\mathbf{H}$), which we will use later.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial D}{\partial x} & \frac{\partial D}{\partial y} & \frac{\partial D}{\partial \sigma} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} & D_{x\sigma} \\ D_{yx} & D_{yy} & D_{y\sigma} \\ D_{\sigma x} & D_{\sigma y} & D\sigma\sigma \end{bmatrix}$$

$$\hat{\mathbf{x}} = \mathbf{H^{-1}} \cdot \mathbf{J}$$

A. Eliminating Edge Responses

The Difference of Gaussian function has strong responses along edges, even if the location of the edge is poorly determined and therefore unstable to small amounts of noise. Hence, we remove keypoints along edges using the 2×2 Hessian ($\mathbf{H_2}$), computed at the location and scale of the keypoint. Instead of computing eigen values to determine edges, we can use less computationally expensive trace and determinant operations. If α, β are the eigen values, then,

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

To eliminate the dependency of the eigen values, let us consider the ratio of eigen values, $r = \frac{\alpha}{\beta}$. Then the fraction $\frac{(Tr(\mathbf{H}))^2}{Det(\mathbf{H})}$ becomes,

$$\frac{(Tr(\mathbf{H}))^2}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta\beta} = \frac{(r+1)^2}{r}$$

The fraction is at its minimum when the two eigen values are equal and its value increases when one of them becomes larger (r grows larger). If r grows larger, then it's a sign of an edge (either horizontal or vertical) detected.

V. Orientation Assignment

After experimentation with several approaches, the authors found the approach proposed in the paper to give the most stable results.

For each detected keypoint, depending on its scale, the corresponding Gaussian-filtered image $L(x, y, \sigma)$ with nearest σ is selected. This makes all the computation carried out further scale-invariant. We then take a patch around the keypoint from it's corresponding image $L(x, y, \sigma)$ and assign it with an orientation corresponding to its dominant gradient direction. The dominant direction is determined using *Histogram of Oriented Gradients* (HOG). At each pixel in the patch, the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are given by

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

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

An orientation histogram is computed for each pixel of the patch. The histogram has 36 bins (10 degrees for a bin) covering 360 degrees. Each sample added to the histogram is weighted by its gradient magnitude and also by a Gaussian-weighted circular window with a σ that is 1.5 times the scale of the keypoint. The peaks in the histogram represent the dominant directions of local gradients.

After the highest peak in the histogram is identified, any other peak in the histogram with height more that 80% of the highest peak is also used to create a keypoint. This enables creating multiple keypoints at the same location and scale but with different orientations.

Finally, a parabola is fit to the 3 histogram values closest to each peak to interpolate peak position for better accuracy. A parabola with the equation $y = (x-h)^2 + k$ has vertex at (h, k), is upward facing, and it's axis parallel to the Y-axis passing through line $x = h$. This is shown in the Fig 2 (Left). A parabola in a more general form is represented by,

$$y = f(x) = ax^2 + bx + c$$

$$f(x) = a\left(x - \left(\frac{-b}{2a}\right)\right)^2 + \frac{4ac - b^2}{4a}$$

The above equation represents a parabola with vertex at $(\frac{-b}{2a}, \frac{4ac-b^2}{4a})$. The Fig 2 (Right) shows the fitting of the parabola to interpolate the peak position. We consider the three points $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3) from the mid-points of the peak bin and its adjacent bins. We now try to fit a parabola (defined by (a, b, c)) that best passes through these points minimizing the least squares error. Because the points lie on the parabola, they must satisfy its equation, giving rise to the following.

$$y_i = ax_i^2 + bx_i + c, \quad i \in \{1, 2, 3\}$$

The equations can be represented in the matrix form as,

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\mathbf{Ax} = \mathbf{b}$$

Applying the least squares method for solving the above matrix equation gives the best solution for (a, b, c), defining the parabola. Now, it's vertex, given by the x-coordinate value, $\frac{-b}{2a}$ is the new peak value.

VI. Local Descriptor Creation

After assigning scale and orientation each identified keypoint, the next step is to compute a descriptor that is for the patch around the keypoint that is highly distinctive and yet as invariant as possible to variations in illumination and 3D viewpoint. Edelman, Intrator, and Poggio (1997) in [6] demostrated that matching gradients while allowing for shifts in their position results in much better classification under 3D rotation. Their approach works better than the relatively obvious apporach of sampling the local image intensities

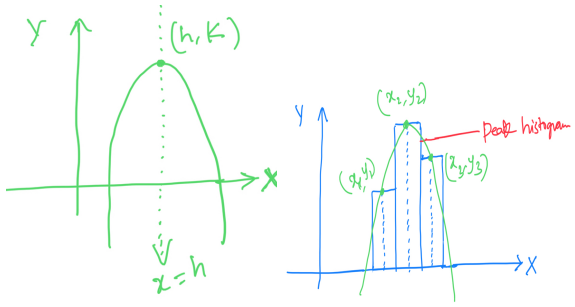

Fig. 2: (Left) Parabola with vertex at (h, k). (Right) Fitting a parabola to the histogram

around the keypoint at the appropriate scale, and to matching these using a normalized correlation measure. The latter approach is more sensitive to affine transformation, changes in 3D viewpoints, and non-rigid deformations.

The author proposed to select a 16×16 patch around each keypoint, using the scale of the keypoint to select the level of Gaussian blur for the image, and divide the patch into 16 (4×4) sub-regions. The polar gradients of each pixel each sub-regions are binned into an 8-bin histogram. The gradients are down-weighted by a Gaussian function with σ equal to half the width of the descriptor window $(16/2 = 8)$. This will reduce the influence of the gradients far from the center. All the bins of 16 histograms of the sub-regions are concatenated into a 128-length vector that defines the feature descriptor.

VII. Results

Fig 3 shows the keypoints detected across multiple octaves of the famous image *lena.png*. Fig 4 demonstrates feature matching on the original image and its rotated version. I used OpenCV's snippets feature matching and plotting them. For this, I had to convert the my keypoint from numpy arrays to OpenCV's `KeyPoint`.

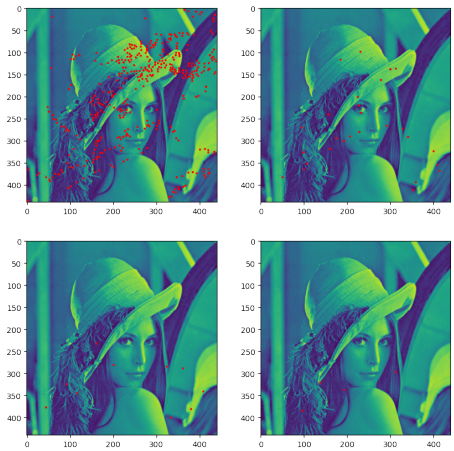

Fig. 3: Keypoints detected across multiple octaves on *lenna.png*

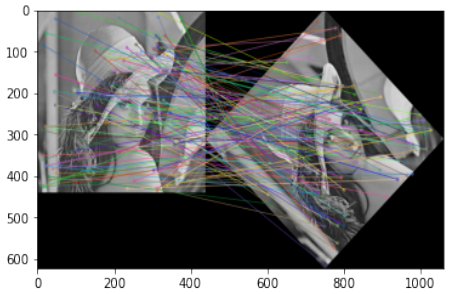

Fig. 4: Feature Matching from **custom** SIFT implementation

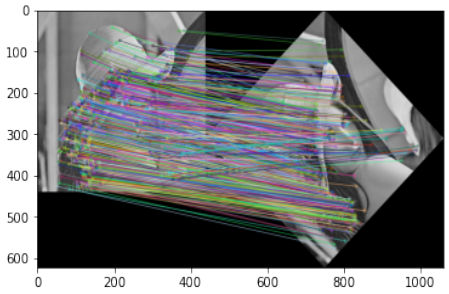

Fig. 5: Feature Matching from **OpenCV's** SIFT implementation

Figs 4, 5, 6 show the differences between feature matches from custom implementation, **OpenCV** and **TFeat**. **TFeat's** is a deep learning pre-trained CNN that produces learned feature descriptors. It's feature descriptors are extracted from OpenCV's detected keypoints

VIII. CONCLUSIONS

My implementation provides decent amount of descriptors when compared to OpenCV. I ignored those patches that are on the edges of the images due to time contraints, which require more work. It has many runtime bottlenecks fundamentally with the usage of **for** loops, which can be vectorized. There is some redundant computation of gradients of the images in the octave which can be avoided. Unfortunately, I couldn't take the time to make these improvements. It takes 10x multiples of runtime when compared to OpenCV.

REFERENCES

[1] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94
[2] Witkin, A.P. 1983. Scale-space filtering. In International Joint Conference on Artificial Intelligence, Karlsruhe, Germany, pp. 1019-1022.
[3] Koenderink, J.J. 1984. The structure of images. Biological Cybernetics, 50:363-396.
[4] Lindeberg, T. 1994. Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics, 21(2):224-270.
[5] Brown, M. and Lowe, D.G. 2002. Invariant features from interest point groups. In British Machine Vision Conference, Cardiff, Wales, pp. 656-665.
[6] Edelman, Shimon & Intrator, Nathan & Poggio, Tomaso. (1997). Complex Cells and Object Recognition.

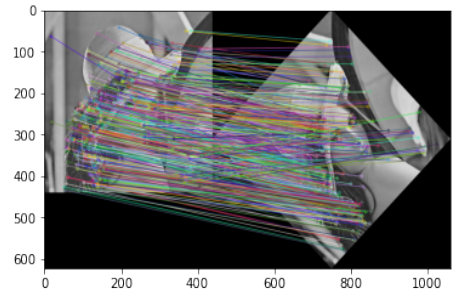

Fig. 6: Feature Matching from **TFeat's** descriptors

[7] Moravec, H. 1981. Rover visual obstacle avoidance. In International Joint Conference on Artificial Intelligence, Vancouver, Canada, pp. 785-790.
[8] Harris, C. 1992. Geometry from visual motion. In Active Vision, A. Blake and A. Yuille (Eds.), MIT Press, pp. 263-284.
[9] Schmid, C., andMohr, R. 1997. Local grayvalue invariants for image retrieval. IEEE Trans. on Pattern Analysis and Machine Intelligence, 19(5):530-534.
[10] Lowe, D.G. 1999. Object recognition from local scale-invariant features. In International Conference on Computer Vision, Corfu, Greece, pp. 1150-1157.