Lecture Note for SIFT feature descriptor

Xiaolong Guo
SYSU
Guangdong, China
guoxlong3@mail2.sysu.edu.cn

Abstract

This article is a lecture note for the Scale Invariant Feature Transform (SIFT) proposed by Lowe in 1999 [1] and 2004 [7]. A brief introduction to the background of SIFT has been make, that SIFT was proposed to achieve scale invariance for interest point features, imposing the scale space theory. The main designs of SIFT are described in detail, in a much formal way with mathematical symbols, referring to the SIFT papers and available public implementations. Using the OpenCV implementation of SIFT, experiments are taken on the robustness and parameter selection of SIFT, with analysis on the matching results of features. Finally, some further work based on SIFT are introduced, and conclusions about how SIFT achieve scale invariance and other robustness and other experiment results are summarized. To understand SIFT more in-depth, the plan on implementing SIFT on GPU to achieve the real time performance at the millisecond level has been made, in the Multi-Core Program Design and Practice course.

1. Introduction

1.1. Interest Point and Local Features

Image features are useful in image tasks like object recognition (e.g., provide distinctive descriptors for different objects for matching). Features should be distinctive enough to identify specific objects and are supposed to be invariant to irrelative changes, such as scale, rotation, illumination, etc. It's a challenge to design such invariant and distinctive features. One possible method is to detect the interest points (e.g., corners) in image, whose surroundings may contain important features of an object. By describing the surroundings, we can get features that are sufficiently distinctive .

In 1990s, some dense local features had been proposed and successfully applied to object recognition. A famous one of them is the Harris corner features proposed by Harris and Stephens [4], which are detected from the peaks in local image variation. But a main disadvantage of these features is that they are not invariant to scale changes. On one hand, there may be not enough interest points in the image at the original scale. On the other hand, different images containing same objects at different scales may generate different features. One method for this problem is to detect features in one image at different scales, and match the features among all of them. A much more effective method is to detect the features invariant to image scale.

1.2. Scale Space Theory

The notion "scale" mentioned above can be referred to the level of details being processed. For example, we see the profile of a tree when we observe it from a distance, which is a large scale. When we observe it close, we see the details of leaves, which is a small scale. There can be various scales in one image, vary from smallest local details to the whole image. For processing methods that process the local image variation, such as Harris corner detector, the features is described at the smallest scale in the image.

To detect features at different scales, some methods for multi-scale representation of image were proposed. The scale-space representation introduced by Witkin [8] comprises a continuous scale parameter and preserves the same spatial sampling at all scales. Lindeberg [6] later shown that the Gaussian kernel is a unique choice as the one-parameter family of kernels for the image to convolve with so as to be embedded into the scale-space representation.

1.3. SIFT

In 1999, Lowe proposed a new method for image feature generation called the Scale Invariant Feature Transform (SIFT) [1] and improved it later in [7]. The main idea of SIFT is to detect the extremas in image scale space, which are naturally irrelative to the original scale of the image and therefore be scale invariant. Following are the main stages of SIFT:

1. Detection of Scale-Space Extrema: Build the scale space with the Gaussian Pyramid, and detect extremas in DoG space as keypoints.

2. Accurate Keypoint Localization: Use 3D quadratic function to fit the DoG space, so as to relocate the keypoints by interpolation.

3. Orientation Assignment: Extract an consistent orientation from the region around each keypoint, to enable the relative orientation descriptors that invariant to rotation.

4. The Local Image Descriptors: Divide the region around each keypoint to subregions and create orientation histograms to summarize the subregions, to form a feature vector containing all the histogram values as the final descriptor.

2. Applications

As a feature extraction method, SIFT can be used in many image tasks. An original application of SIFT is object recognition [7]. In object recognition, the SIFT features of each different objects are extracted to build a database. In order to identify the objects included in the given image, new features are extracted from the image and matched with the features in the database to find the most likely objects the new features belong to. The points that possibly belong to the same object in the image are obtained, then the location of the object in the image can be roughly located by fitting these points through Hough transform with the object model.

In addition to object recognition, SIFT has also been successfully applied to various image tasks, such as image match, image retrieval, image stitch, 3D reconstruction, etc. SIFT can be further applied to video tasks, such as target tracking, object detection, etc.

All in all, SIFT makes it easy to get a considerable number of features that are robust (e.g., scale invariant) and distinctive. For visual tasks that need such features, SIFT had been shown to be a good choice.

3. Approach

3.1. Detection of Scale-Space Extrema

3.1.1 Scale Space

The scale space of an image is defined as a function, $L(x, y, \sigma)$, which is produced from the convolution of a variable-scale Gaussian, $G(x, y, \sigma)$, with the image, $I(x, y)$:

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

where $*$ is the convolution operation in x and y, and

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

Here the function, $L(x, y, \sigma)$, comprises a continuous scale parameter σ and does not change the spatial sampling for different σ, which is in accord with the scale-space representation from the scale space theory [8] (e.g., the finest scale in the representation increases when σ increases).And it uses the family of Gaussian kernel with the only parameter, σ, because the only possible scale-space kernel is the Gaussion function, as shown by Lindeberg in [6].

3.1.2 Exreama Difference in Scale Space

The difference in scale space can be expressed as the difference-of-Gaussian (DoG) function convolved with the image, $D(x, y, \sigma)$, which can be computed from the difference of two nearby scales seperated by a constant multiplicative factor k:

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

The DoG function can provide a close approximation to the scale-normalized Laplacian of Gaussian (LoG), $\sigma^2\nabla^2 G$, which is shown to be the true scale invariance process (e.g., a necessary condition is that the process can be expressed by dimensionless variables) by Lindeberg in [6]:

$$\begin{aligned} G(x, y, k\sigma) - G(x, y, \sigma) &\approx (k\sigma - \sigma)\frac{\partial G}{\partial \sigma} \\ &= (k-1)\sigma^2\nabla^2 G \end{aligned}$$

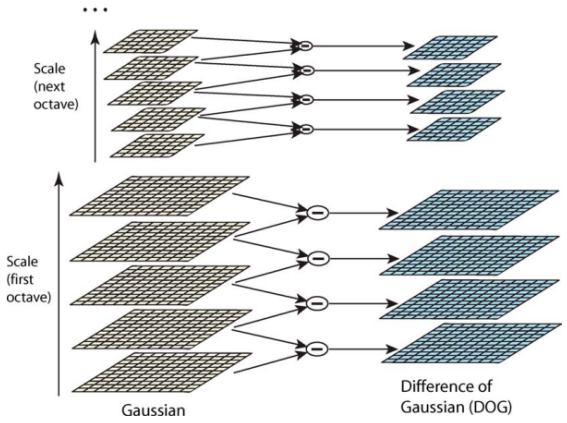

Figure 1. The Gaussian Pyramid to represent the scale space.

To efficiently compute the DoG of image, the scale space are first generated, then each DoG result is obtained by subtracting the neighbors in scale space, as shown in figure 1. The scale space is divided into K octaves, $\{R_i | i \in z, 1 \leq i \leq K\}$. The ith octave contains $S + 3$ representations, $\{R_{ij} | j \in z, 1 \leq j \leq S + 3\}$. Their scale parameter $\{\sigma_{ij}\}$ are determined by:

$$\begin{aligned} \sigma_{i(j+1)} &= 2^{1/S}\sigma_{ij} \\ \sigma_{(i+1)1} &= 2\sigma_{i1} \end{aligned} \tag{1}$$

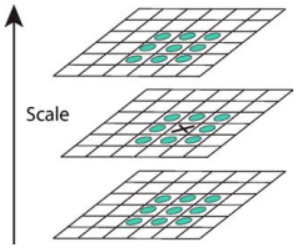

Figure 2. The extrema in DoG space and its 28 neighbors.

The extremas in DoG space, as shown in figure 2, are the interest points that have different scales and therefore provide scale invariant information. The extremas, $\{(x_e, y_e, i_e, j_e)\}$, are taken from the S DoG layers in the middle of each octave, $E = \{D_{ij} | i, j \in z, 1 \leq i \leq K, 2 \leq j \leq S+1\}$, when $D(x_e, y_e, \sigma_{i_e j_e})$ is larger than all the 28 neighbors in the scale space:

$$D(x_e, y_e, \sigma_{i_e j_e}) > D(x_e + \Delta x, y_e + \Delta y, \sigma_{i_e (j_e + \Delta j)}),$$
$$\forall \Delta x, \Delta y, \Delta j \in \{-1, 0, 1\}, (\Delta x, \Delta y, \Delta j) \neq (0, 0, 0)$$

, or smaller than all the 28 neighbors.

The condition 1 ensures that the scales of DoG layers E, which generate extremas, increase smoothly at a constant scale of $2^{1/S}$:

$$\begin{aligned} E_l &= E_{(i-1)\cdot S + j - 1} \\ &= D_{ij} \\ &= D(x, y, \sigma_{ij}) \\ &= D(x, y, 2^{(j-1)/S} \sigma_{i1}) \\ &= D(x, y, 2^{(j-1)/S} \cdot 2^{i-1} \sigma_{11}) \\ &= D(x, y, 2^{((i-1)\cdot S + j - 1)/S} \sigma_{11}) \\ &= D(x, y, 2^{l/S} \sigma_{11}), i, j \in z, 1 \leq i \leq K, 2 \leq j \leq S+1 \end{aligned}$$

As shown in figure 1, the sizes of representations are same in one octave but reduce by half in next octave, while the scale of the first DoG result generated by the octave get double. As stated by Lowe, this keep the accuracy of sampling relative to the scale, and greatly reduce the computation increase caused by the scale increase. To be specific, the first representation $R_{(i+1)1}$ in next octave is resample by taking every second pixel in each row and column, from R_{iS}. Then the size reduce by double and $\sigma_{(i+1)1} = \sigma_{iS} = 2^{S/S} \sigma_{i1} = 2\sigma_{i1}$, which is accord with condition 1. In addition, as the size in each octave reduce and the smallest size limited, the max value of K also get limited and determined.

So one parameter in building the scale space is S, while K is determined. When S is larger, the scale space get smoother and extremas can be generated from more DoG layers. Another parameter is the initial scale, σ_{11}, which can modify the initial smoothing degree. To compensate the loss of highest spatial frequencies in the input image cause by the initial smoothing, the input image is expanded by a factor of 2 using bilinear interpolation, prior to building the pyramid.

3.2. Accurate Keypoint Localization

3.2.1 Interpolated Location of the Extremas

The scale space shown in figure 1 is discrete, so the extrema extracted from it can change to one of its neighbors when the original scale changes. In 2002, Brown and Lowe [3] proposed an approach to determine the interpolated location of the extremas, which can improve the stability of the extremas. The approach uses the Taylor expansion to fit the 3D quadratic function to the scale space function, shifted so that the origin is at the sample point (i.e., the previous detected extrema):

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

where $\boldsymbol{x} = (x, y, \sigma)^T$ is the offset from this point and D, $\frac{\partial D}{\partial \boldsymbol{x}}$ and $\frac{\partial^2 D}{\partial \boldsymbol{x}^2}$ are the value at the sample point of the original function and its derivatives. Then the extremum $\hat{\boldsymbol{x}}$ is determined by setting the derivative of the function with respect to $\boldsymbol{x}$ to zero:

$$\frac{\partial \tilde{D}}{\partial \boldsymbol{x}} = \frac{\partial D}{\partial \boldsymbol{x}} + \frac{\partial^2 D}{\partial \boldsymbol{x}^2} \boldsymbol{x} = 0$$

which leads to:

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

If the offset $\hat{\boldsymbol{x}}$ is larger than 0.5 in any dimension, then it means that the extremas lies closer to a different sample point. Then that point will take the place of current point as a candidate extrema, and the same interpolation will be preformed instead about that point, until that the offset is not larger than 0.5.

3.2.2 Low Contrast Rejecting

The extrema value obtain by interpolated is:

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

All extremas with a value of $\tilde{D}(\boldsymbol{x})$ less than 0.03 were discarded, when the image pixel value is in range $[0, 1]$. This results in rejecting the unstable extrema with low contrast.

3.2.3 Eliminating Edge Responses

The DoG function will have a strong response along edges and therefore unstable to small amounts of noise. Such peaks in the DoG 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 the Hessian matrix:

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

Let α be the eigenvalue of H with the largest magnitude and β be the smaller one, then:

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

Let r be the ratio between α and β, then $\alpha = r\beta$, and:

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

To check that the ratio of principal curvatures is below some threshold, r_0, we only need to check:

$$\frac{Tr(\boldsymbol{H})^2}{Det(\boldsymbol{H})} < \frac{(r_0+1)^2}{r_0}$$

which is very easy to compute, without computing the eigenvalues.

The keypoints that have a ratio between the principal curvatures greater than r_0 will be eliminated.

3.3. Orientation Assignment

For each key point reserved in interpolated localization, SIFT determines a consistent orientation from the surrounding of the point, and generates description relative to this orientation, so as to eliminate the influence of image rotation on the description.

To determine the orientation for an key point, (x_e, y_e, σ_e), SIFT uses the orientation histogram to count the distribution of orientations in the corresponding smoothed image, $L(x, y, \sigma_e)$, with the weight of the gradient magnitude and a Gausian-weighted circular window with $1.5\sigma_e$. i.e., the contribution of the pixel, which has an offset (x, y) from the key point, to the bin $b = \lfloor \frac{\theta(x,y,\sigma_e)}{10} \rfloor$ in the 10 bins histogram is:

$$h(x, y, b) = m(x, y, \sigma_e) \cdot G(x, y, 1.5\sigma_e)$$

where the gradient magnitude, $m(x, y, \sigma)$, and orientation, $\theta(x, y, \sigma)$, are precomputed using pixel differences:

$$\begin{aligned} M &= (L(x+1, y, \sigma) - L(x-1, y, \sigma))^2 \\ &\quad + (L(x, y+1, \sigma) - L(x, y-1, \sigma))^2 \\ m(x, y, \sigma) &= \sqrt{M} \\ \theta(x, y, \sigma) &= \tan^{-1}\left(\frac{L(x, y+1, \sigma) - L(x, y-1, \sigma)}{L(x+1, y, \sigma) - L(x-1, y, \sigma)}\right) \end{aligned}$$

Peaks in the orientation histogram corresponding to the main orientation, and orientations that have histogram value over 80% of the peak are all used to create duplicated keypoints with them separately as the main orientation θ_e, so as to make the main orientations more stable. In addition, a parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for better accuracy.

3.4. The Local Image Descriptors

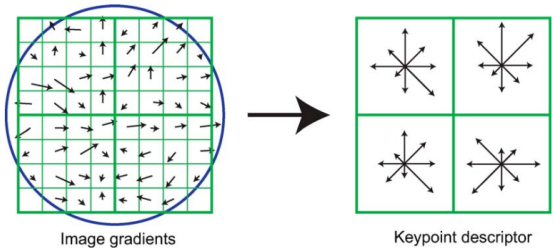

Figure 3. An example of subregions and orientation histograms in 8×8 regions, 2 subregions and 8 orientations in histogram.

To generate descriptors for each key point, (x_e, y_e, σ_e), the $k_r \times k_r$ region around the point is divided to $k_s \times k_s$ subregions, as shown in figure 3. In each subregion, an orientation histogram with k_b orientations, which cover the orientation range of $[0, 360)$ uniformly, is create to summarize the orientations in each pixels of the subregion. All orientations are relative to the main orientation of the key point computed in section 3.3, in order to achieve rotation invariance.

For each pixel in the subregion, the gradient magnitudes weighted with Gaussian is added to the nearest orientation in the histogram. While Gaussian weights improve stability, the histogram allows significant shift of gradient positions inside the subregion. To avoid boundary affects caused by the shift from on subregion to another, a trilinear interpolation is used to distribute the value of each gradient sample into adjacent histogram bins. The contribution of the pixel, which has an offset (x, y) from the key point, to the bth bin in subregion s is:

$$h(x, y, b, s) = G(x, y, 1.5K_r) \cdot m(x, y, \sigma_e) \cdot t(x, y, s)$$

where b is the bin with nearest orientation to $\theta(x, y, \sigma_e) - \theta_e$, and $t(x, y, s)$ is the weight calculated by trilinear interpolation.

The descriptor is formed from a vector containing the values of all the orientation entries, with a length of $L = K_s \times K_s \times K_b$:

$$f[((i-1) \cdot K_s + j) \cdot K_b + b] = \sum_{x,y} h(x, y, b, s)$$
$$\forall i, j, b \in z, i, j \in [1, K_s], b \in [1, K_b]$$

The feature is normalized to unit length to reduce the effects of linear illumination change:

$$f[i] = \frac{f[i]}{\sum_j F[j]}, \forall i \in z, i \in [1, L] \quad (2)$$

And the values in the unit feature vector are threshold to each no larger than 0.2, in order to reduce the influence of non-linear illumination changes:

$$f[i] = max(f[i], 0.2)$$

Finally the feature is renormalized by 2.

4. Experiment

4.1. Scale Invariance

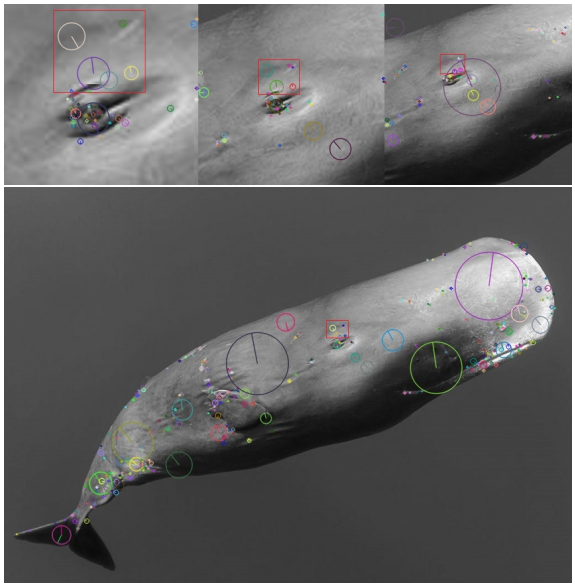

Figure 4. An example to show the scale invariance. Generated by opencv. Each feature is represented by a circle region and an orientation.

To show if the features extracted by SIFT are invariant to scale changes or not, we show the features from different scale images in figure 4. It can be found that some features from different images are similar in position, region size and orientation. The features surrounded by a red rectangular provide an clear example.

4.2. Other Robustness

We use the KNN method to make matches between the features extracted by SIFT using opencv in different images that include rotation change, 3D view point change, occlusion, Gaussian Blur, addition of noise or illumination change. The results shown from figure 5 to figure 8 are good examples that show the robustness of SIFT to these changes.

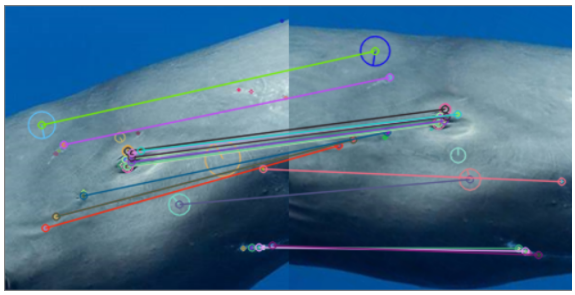

Figure 5. The match results in rotation changing images. The result shows SIFT's good invariance to rotation changes, as most matches are correct.

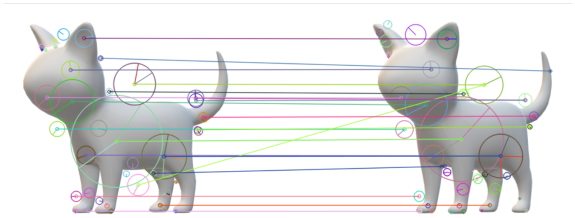

Figure 6. The match results in rotation 3D view point images. We create the 3D images with a 3D model of cat, and change the view point to observe it. The results show that SIFT has good robustness to the 3D view point changes, as many features are successfully matched.

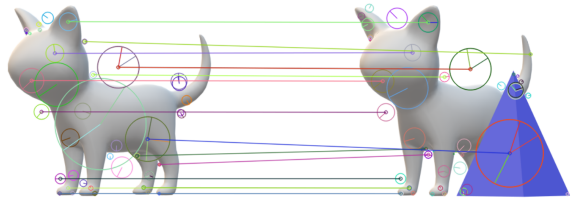

Figure 7. The match results in images with occlusion. We add another irrelevant 3D model in front of the cat to make occlusion. The results show that SIFT has some robustness to the occlusions, if there are enough features outside the occlusion.

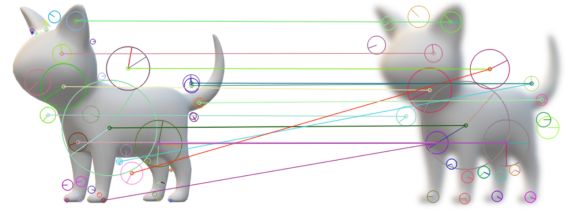

Figure 8. The match results in images with Gaussian Blur. We create an Gaussian Blur with $\sigma = 5$ from the 3d view point changing image. The results show that SIFT has some robustness to the Gaussian Blur, which can be considered as scale change. As we can see, the features from small scales are not successfully matched since the change of region size, but the features from large scales still provide a good match between the images.

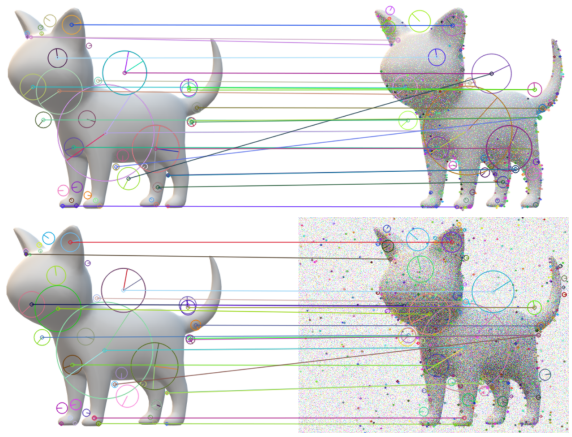

Figure 9. The match results in images with salt noise and Gaussian noise. The noise are randomly generated at each pixel in the 3D view point changing image. As we can see, though the noise does bring some irrelevant features, the relevant features are not much affected. So these good features still provide an considerable number of successful matches between the images.

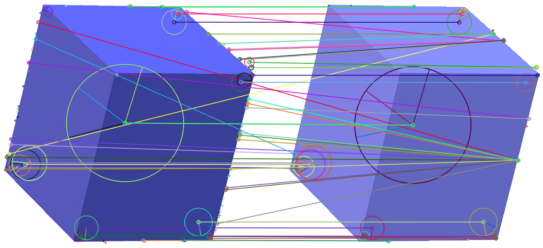

Figure 10. The match results in images with illumination. We add linear illumination changes to the test image by adding a constant value to the pixels, which results in the increase of brightness. The result shows that SIFT has good robustness to linear illumination changes.

4.3. Selection of Parameter

The parameters in SIFT includes the number of extreme layers in each octave, S, the contrast threshold to reject low contrast points, the ratio threshold r to eliminate edge response, the region size, subregion number, histogram size and so on. Our experiment mainly focus on the selection of S and r.

As shown in figure 11, as S increasing, the number of features also increase. But the number of incorrect matches also increase to some extent, and the computation time increases as expected. So a moderate value of S (e.g., $S = 4$) is better, which brings enough features, less unstable features and less computation comsumption.

As shown in figure 12, when r is too small, most keypoints will be eliminated and therefore fewer features are obtained. When r is too large, the elimination does not produce a marked effect, and there are more unstable features that do not good at matching. So a moderate value of r (e.g., $r = 10$) is better.

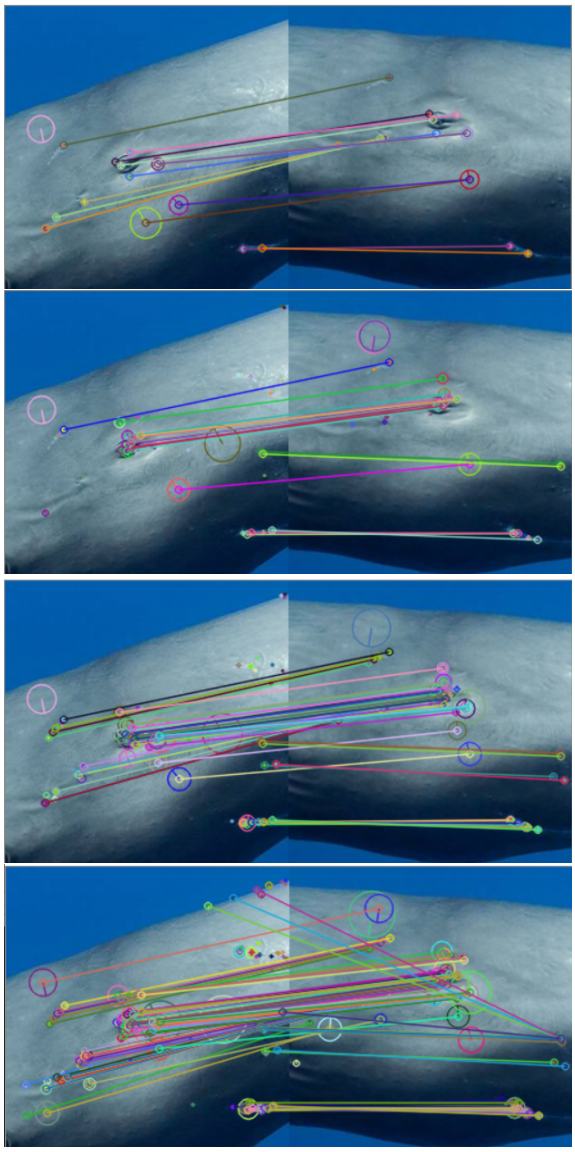

Figure 11. The match results with different value of S. From top to bottom, $S = 1, 4, 16, 32$, respectively. As we can see, as S increases, more features and matches are obtained, but the mistakes also increase to some extent.

4.4. Smooth Shapes

A special case for smooth shape objects are studied in figure 13, where SIFT shows its weakness. When a moderate values of r is taken, the number of features are too small to provide effective matches. When we increase r, the number of features seems large enough, but most features fail to make correct matches. So it's hard to use SIFT for such smooth shapes.

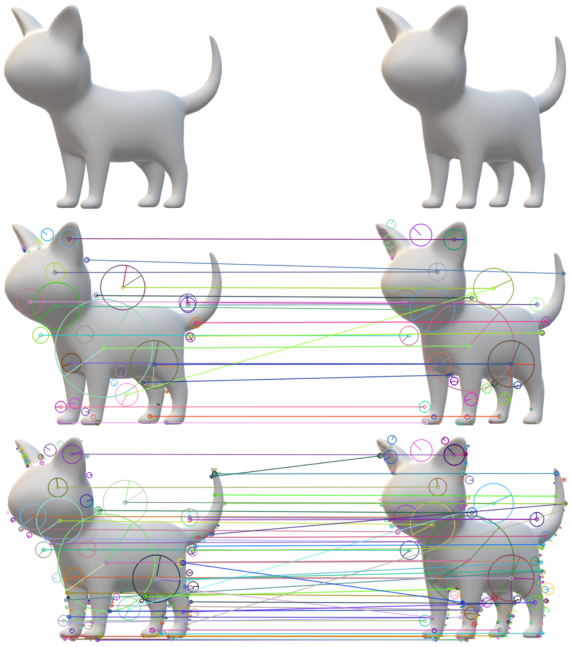

Figure 12. The match results with different value of r. From top to bottom, $r = 1, 10, 100$, respectively. As we can see, the moderate value of r leads to the best result.

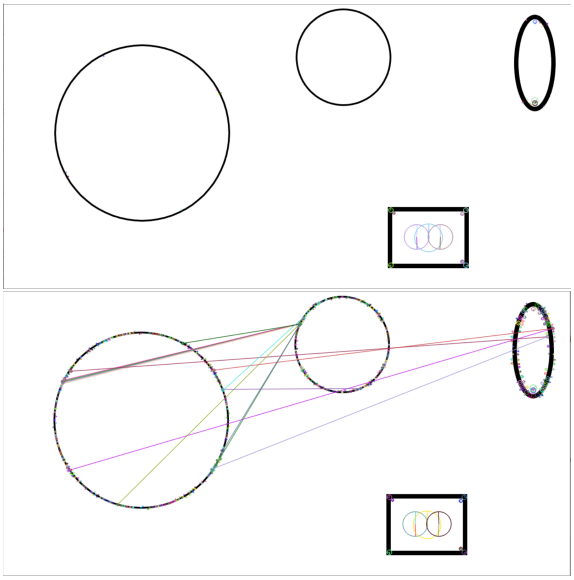

Figure 13. The match results for smooth shape objects with different value of r. From top to bottom, $r = 10, 100$, respectively. As we can see, the features are sparse and not distinctive enough to make good matches.

5. Further Work

5.1. SURF

Speeded-Up Robust Features (SURF) is another scale- and rotation-invariant detector and descriptor proposed by Bay *et al*. [2] in 2006, which can be computed faster while still maintaining competitive performance with respect to repeatability, distinctiveness, and robustness.

The main stages of SURF are similar to SIFT, and some of them are designed to reduce computation:

1. Detection of Scale-Space Extrema: The Gaussian filters in SIFT are approximated with the box filters in SURF, which can be evaluated at a very low computational cost using integral images. The DoG function are replaced by the determinant of Hessian matrix, which represents the blob response at the image point. There are no downsampling in the building of scale scape in SURF, and the scale space is formed by upscaling the filter size rather than iteratively reducing the image size. Therefore different layers of the scale space can be generated independently and even in parallel, which brings more computational efficiency.

2. Accurate Keypoint Localization: A fast variant of non-maximum suppression is used, and the maxima of the determinant of the Hessian matrix are interpolated in scale and image space, which is similar t the interpolation location of keypoints in SIFT.

3. Orientation Assignment: The dominant orientation is detected in a sliding orientation window, where the sum of Gaussian weighted Haar wavelet responses is the largest, within a circular neighbourhood around the interest point. The integral images are used again for fast filtering.

4. The Local Image Descriptors: An oriented quadratic grid with 4×4 square sub-regions is laid over the interest point, and the wavelet responses are computed for each square. Each sub-region has a four-dimensional descriptor vector containing the information about the responses. All sub-region vectors are concatenated to be the SURF descriptor.

5.2. SIFT on GPU

The high computation complexity of SIFT limits its application in real-time systems and large-scale data processing tasks. To improve the real-time performance of SIFT, several efficient optimizations have been proposed by Li *et al*. [5] by exploiting the computing resources of CPUs and GPUs in a heterogeneous machine.

6. Conclusion

6.1. Why SIFT is Scale Invariant

SIFT extract keypoints in a discrete scale space range from the original scale of the image to a larger scale at a constant rate, $2^{1/S}$, of increase. When S is large enough, the scale space becomes smooth enough to include some

similar scales for different images, regardless of their original scales. Therefore, similar features can be extracted from different scales images.

6.2. Why SIFT is Robust

In addition to the scale invariance, SIFT also enjoys the robustness to the followings:

- Rotation change: this is achieved by the assignment of main orientations and the relative orientations descriptors.
- Affine distortion and 3D viewpoint change: this should be owed to the subregion histograms and trilinear interpolation to distribute the gradients, which allows larger local positional shifts.
- Addition of noise: On one hand, SIFT can extract a considerable number of features, so the influence of noise is relatively small. On the other hand, SIFT eliminates unstable keypoints from the poorly determined edges, and keypoints with low contrast. The eliminations reduce the effect of noise.
- Change in illumination: The final features are normalized and modified to be invariant in illumination changes.

In addition, the keypoints and features are designed to be as stable as possible. For example, the keypoints are located precisely by interpolation, the gradient magnitudes contributed to subregion histograms are weighted by a Gaussian window to improve the stability, and so on.

6.3. Pros and Cons of SIFT

The advantages of SIFT include its high distinctiveness, large quantity and high robustness, especially the good invariance to scale changes.

One disadvantage of SIFT has been shown in experiment, for objects with smooth shapes, the number of distinctive features may be too few for downstream tasks. Another weakness of SIFT is that it's not fast enough to meet the requirement of real time performance.

6.4. How to Choose SIFT Parameters

As shown in section 4.3, moderate values of the number of middle layers in each octave, S, and the ratio threshold to eliminate edge responses, r, are recommended (e.g., $S = 4, r = 10$).

References

[1] Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999. 1

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *Proceedings of the 9th European conference on Computer Vision - Volume Part I*, 2006. 7

[3] M. Brown and D. G. Lowe. Invariant features from interest point groups. In *Proceedings of the British Machine Vision Conference 2002, BMVC 2002, Cardiff, UK, 2-5 September 2002*, 2002. 3

[4] C. G. Harris and M. J. Stephens. A combined corner and edge detector. In *Alvey vision conference*, 1988. 1

[5] Z. Li, H. Jia, Y. Zhang, S. Liu, Shigang Li, Xiao Wang, and H. Zhang. Efficient parallel optimizations of a high-performance sift on gpus. *Journal of Parallel and Distributed Computing*, 124(FEB.):78–91, 2019. 7

[6] Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21:224–270, 09 1994. 1, 2

[7] D. G. Low. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004. 1, 2

[8] A. P. Witkin. Scale-space filtering. *Proc.int.joint Conf.artif.intell*, 1983. 1, 2