Separable and Steerable Gaussian Filters

1. Introduction

This report outlines the implementation of an image processing program designed to extract features such as edges, ridges, and corners using separable and steerable gaussian filters. The program leverages the efficiency of separable filters and the directional flexibility of steerable filters to achieve robust and computationally efficient feature detection. Extension of the basic algorithm includes implementing non maximum suppression for edge refinement and modifying the corner detector with Harris corner detection.

2. Operations

2.1. Input command

To run the program, use the following command format:

```
java SeparableSteerable <input-image> <sigma> <display-mode>
<feature-mode>
```

Example: java SeparableSteerable circle4.pgm 8 1 2
This computes the input image (circle4) in feature mode 2 which is Radial Edges using a sigma value of 8 and visualizes them with display mode 1.

2.2. Padding

When applying convolution, corner and edge elements of the image are less involved in the calculation of the final output as compared to the central element. This can lead to the information of the middle elements having a greater impact on the output. To solve this issue, padding is applied to the input image to ensure edge and corner elements are fully involved in the convolution process.

In this program, padding is added using reflection at the borders, ensuring that values outside the boundary mirror those within. After filtering, the image is cropped back to its original size to maintain consistency.

2.3. Gaussian Function

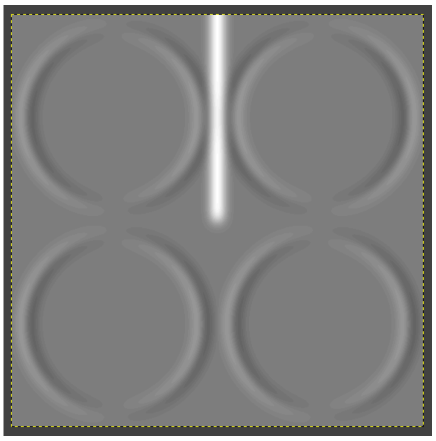
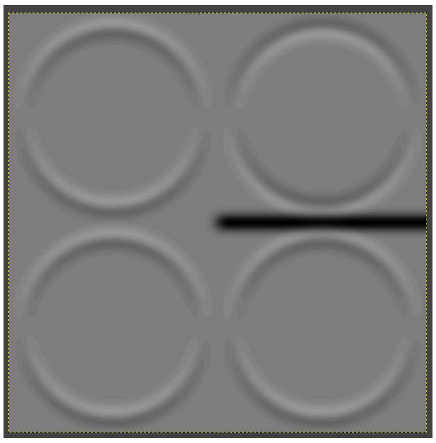
The Gaussian function smoothens the image to reduce noise and detail. For a 2D Gaussian, the equation is: $G(x, y) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2+y^2}{2\sigma^2}}$

For a 1D separable kernel, the equation simplifies to: $G(x) = e^{-\frac{x^2}{2\sigma^2}}$, where x is the distance of a position from the center of the kernel, ranging from -3σ to 3σ.

2.4. First Derivative of Gaussian

The first derivative of the gaussian is used to detect intensity transitions:

$$G'(x) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

Gx: detect horizontal changes	Gy: detect vertical changes
	

Gx is the first derivative of gaussian with 0 degree while Gy is the first derivative of gaussian with 90 degree. By computing both the Gx and Gy, we are able to steer to any direction with the equation: $G_\theta = \cos\theta\, G_x + \sin\theta\, G_y$

2.4.1. Features

2.4.1.1. Standard Edges

Edges usually occur when there are significant changes in intensity. Hence, the computation of standard edges using the magnitude of gradient: $\sqrt{G_x^2 + G_y^2}$

Additionally, non-maximum suppression is applied to refine the edges in gradient intensity which results in thinner and more precise edges. This operation removes the non-maximal pixel response along the gradient direction while preserving strong edge responses.

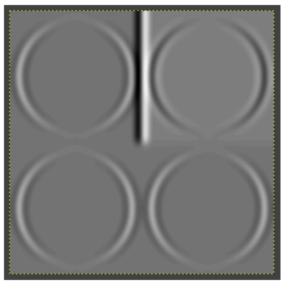
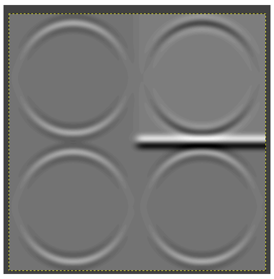
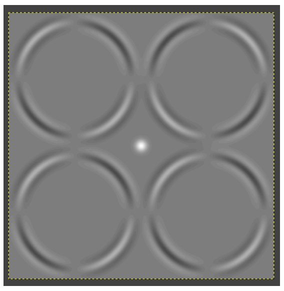
2.4.1.2. Radial Edges

This feature detects edges aligned along radial directions from the center of the image. It is useful for analyzing circular patterns.

2.5. Second Derivative of Gaussian

The second derivative of the gaussian highlights the changes in gradient magnitude which is useful for ridge and corner detection:

$$G''(x) = \frac{x^2 - \sigma^2}{\sigma^4} e^{-\frac{x^2}{2\sigma^2}}$$

Gxx	Gyy	Gxy
		

2.5.1. Feature Modes

2.5.1.1. Standard Ridges

Identifies ridges by computing eigenvalues of the Hessian matrix:

$$\{k_1, k_2\} = \frac{1}{2}\left(G_{xx} + G_{yy} \pm \sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}\right)$$

2.5.1.2. Radial Ridges

Radial ridges focus on detecting elongated structures aligned radially from the image center.

2.5.1.3. Corners

Corners are points where intensity changes in multiple directions. Hence, we can use the determinant of the Hessian matrix to compute the corner response: $G_{xx}G_{yy} - G_{xy}^2$

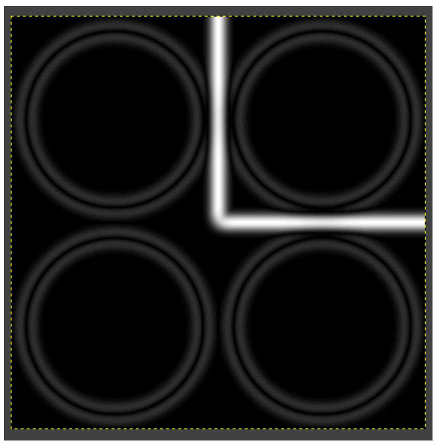
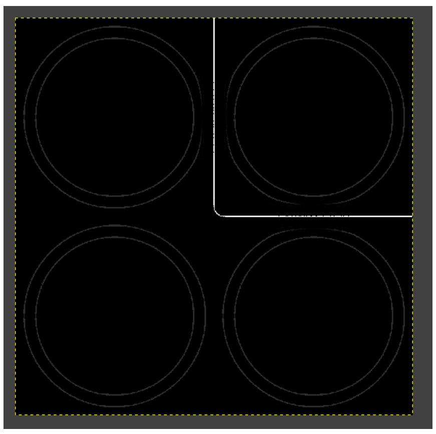
This response is further refined using Harris corner detection, which enhances the sensitivity of corner detection by suppressing weaker responses. Additionally, the standard deviation (s) of the corner response image is computed, and a threshold of 0+3s is applied to identify significant corner points. The resulting corner pixels are overlaid in red on the input image.

2.6. Extensions

2.6.1. Non-maximum suppression for edge refinement

The algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions. The algorithm works as follows:

1. Compute the gradient angle at each pixel using the arctan of the gradient components Gx and Gy.
2. Then compares the gradient magnitude of each pixel to the magnitudes of its neighbors in the edge direction.
3. If the current pixel's magnitude is not greater than or equal to both of its neighbors along the edge direction, the current pixel is set to zero.

Without non-max suppression	With non-max suppression
	

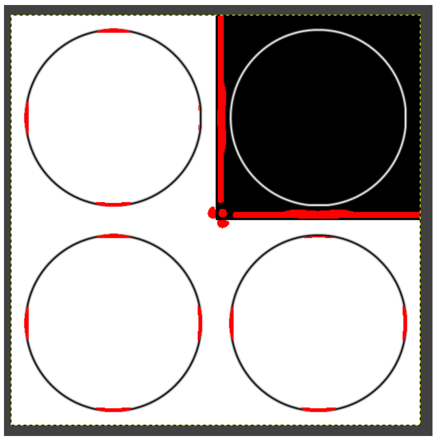
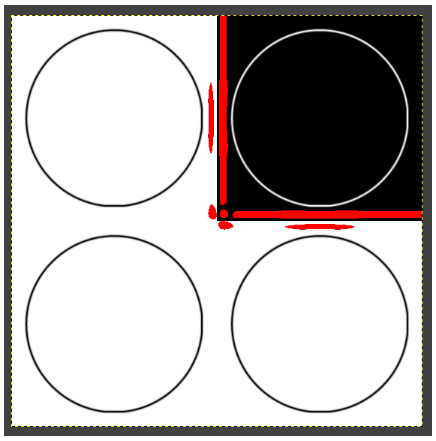
The final result with non-maximum suppression shows thinner edges compared to the image without it. This is because only the strongest edge pixels are retained which improves the quality of edge detection.

2.6.2. Implementation of Harris corner detection

Harris corner detection uses the determinant and trace of the Hessian matrix to identify the corner: $H = det(M) - k\,(trace(M))^2$
where,

- $det(M) = \lambda_1\lambda_2 = G_{xx}G_{yy} - G_{xy}^{\ 2}$
- $trace = \lambda_1 + \lambda_2 = G_{xx} + G_{yy}$
- k is a constant , $0.04 \leq k \leq 0.2$

By tuning the value k, the algorithm will balance the sensitivity of both corners and edges.

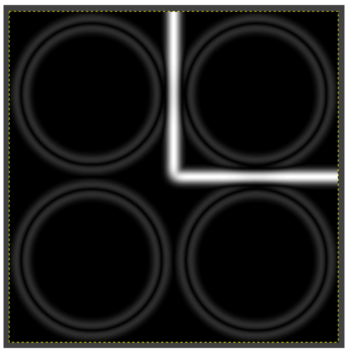
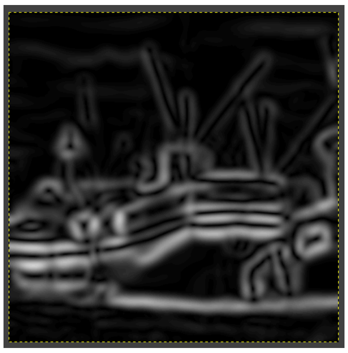
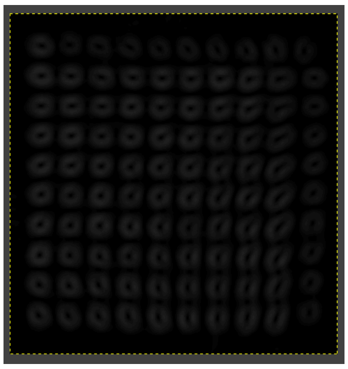
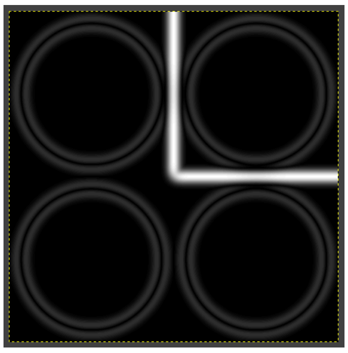
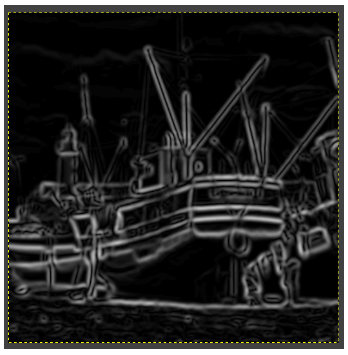
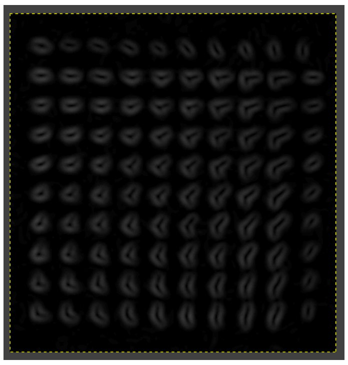
K = 0.04	K = 0.1
	

Looking at the figure above, more edges are falsely detected as corners when k is set to 0.04. By increasing the value of k to 0.1, it reduces the edge responses and improves corner accuracy. This suggests that the larger k values suppress responses from edges in order to improve corner detection.

3. Discussion

Larger sigma (σ) values result in blurrier images, while smaller sigma values produce sharper features. The optimal sigma value varies depending on the characteristics of the image. To get the best feature detection for different images, it's recommended to adjust the sigma value accordingly.

For example, using the same sigma value of 8 across the different images results in suboptimal edge detection. Tailoring the sigma to each image (8 for Circle4, 3 for Sailboat, 5 for Fibres) produces clearer and more appropriate feature extraction.

Circle4 with sigma = 8	Sailboat with sigma = 8	Fibres with sigma = 8
		
Circle4 with sigma = 8	Sailboat with sigma = 3	Fibres with sigma = 5
		

4. Reference

https://sbme-tutorials.github.io/2018/cv/notes/6_week6.html
https://medium.com/@itberrios6/a-quick-introduction-to-steerable-filters-3fd8813f2e63
https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123 https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_harris.html