

Digital Image Processing, Spring 2023

Assignment 2 - Report

R11922133, 廖政華, 資工所碩一

Problem 1: EDGE DETECTION

(a) Sobel edge detection

Below shows the result of Sobel edge detection applied on `sample1.png` with threshold = 75.



(a) sample1.png



(b) result1.png



(c) result2.png

To set proper threshold, one can analyze the cumulative distribution of gray-level values in the gradient image. Lower thresholds tend to preserve more details, but may also introduce more redundant edge points, which is undesirable in terms of single response constraint. In contrast, higher thresholds yield more concise edges, but may result in an overall contour that appears fragmented.

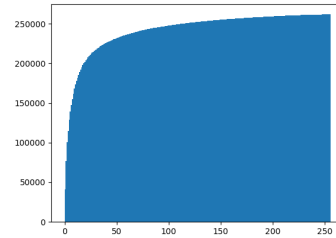


Figure 2: Cumulative distribution of the gradient image



(a) threshold=75



(b) threshold=100



(c) threshold=125

Figure 3: Edge map obtained by different thresholds.

(b) Canny edge detection

The outcomes of each step in Canny edge detection are shown in Figure 4. As the image contains very little noises, I use a Gaussian kernel with $\sigma=1$ and filter size=3 for noise reduction. To apply hysteric thresholding, I set the candidate threshold and edge point threshold at 40 and 75 respectively. Finally, I adopt the eight-neighbor criterion for connected-component labeling.

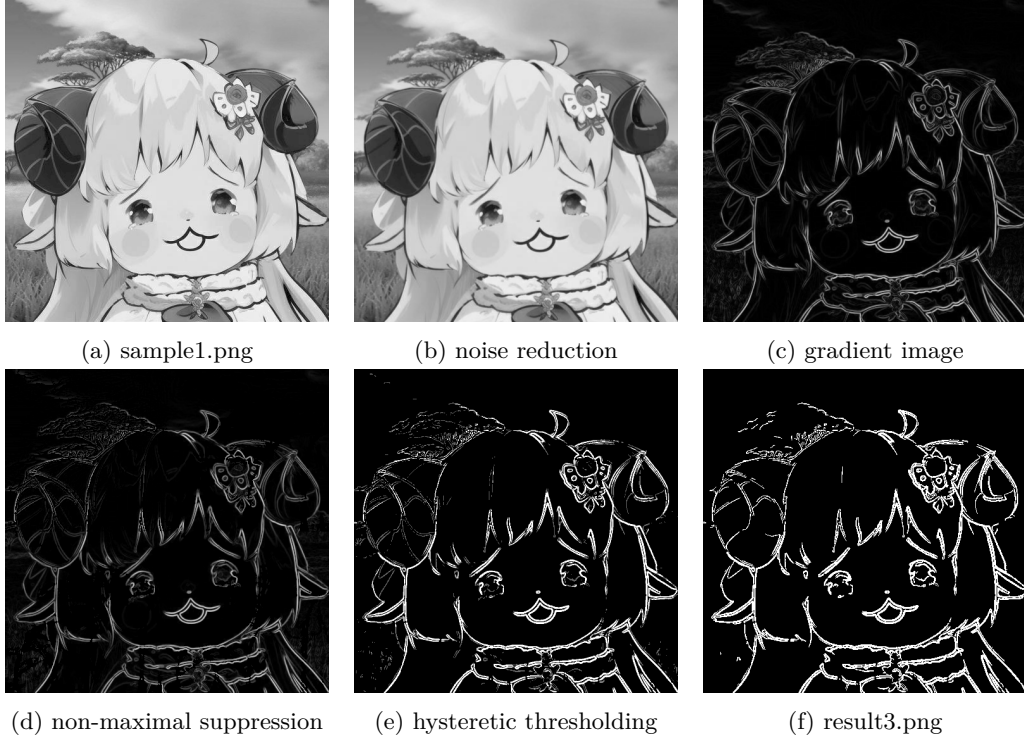


Figure 4: Intermediate results of Canny edge detection.

The effects of setting different threshold values are shown in Table 1.







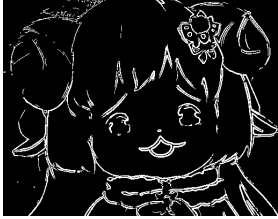
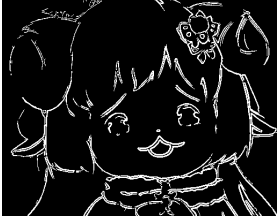
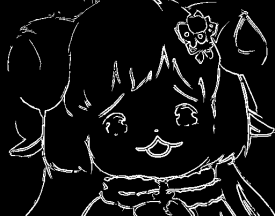
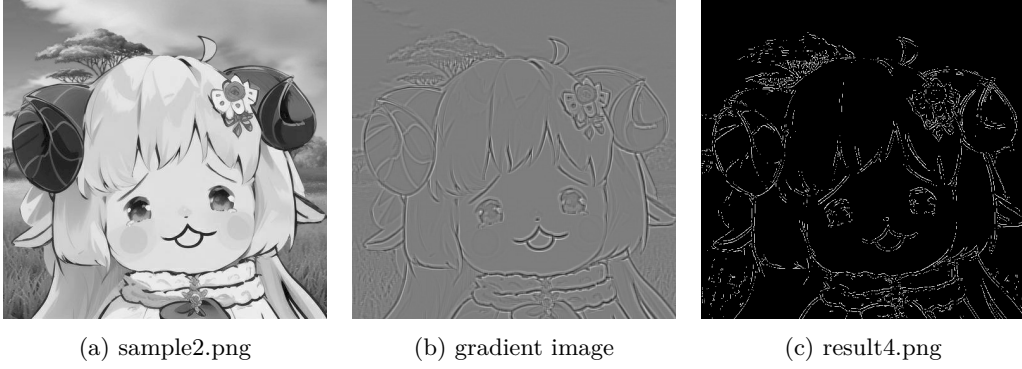
	edge = 75	edge = 95	edge = 115
cand = 20			
cand = 40			
cand = 60			

Table 1: Canny edge detection with different threshold values.

(c) Laplacian of Gaussian edge detection

To obtain `result4.png`, I first conducted Gaussian filtering with $\sigma=3$ and filter size=5, followed by separable-eight-neighbor Laplacian. When performing zero-crossing detection, any gradient values that fall within the range of $[-2, 2]$ will be quantized as zero. I also tried using Gaussian filtering with varying σ and filter sizes, but the outcomes exhibited little difference.



To set up proper threshold to separate zero and non-zero, one can analyze the histogram of the gradient image. As shown in Figure 7, setting a larger threshold may lose more details.

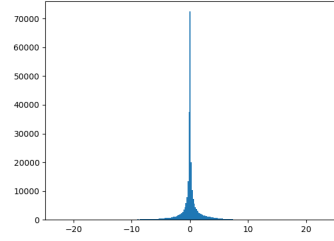


Figure 6: Histogram of the gradient image.

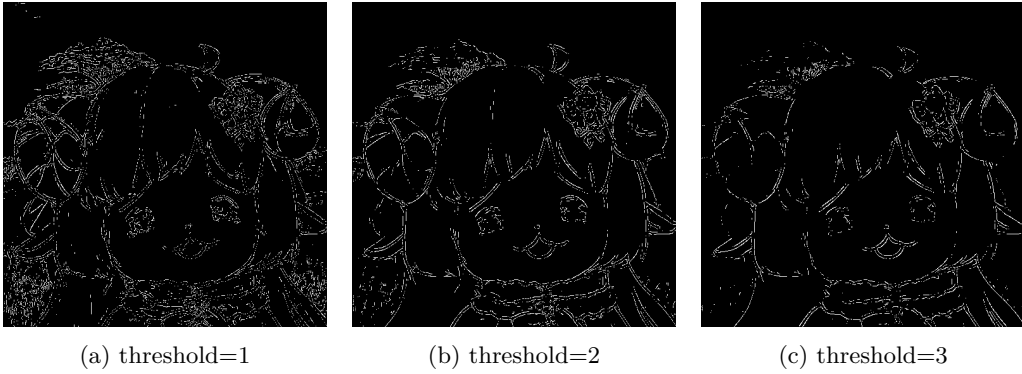


Figure 7: Edge map obtained by different threshold.

Let's now compare `result2.png`, `result3.png`, and `result4.png`. All three preserve the same level of details of the sheep and the tree in the background. `Result4.png` exhibits the most precise edges, satisfying the single response constraint. However, its overall contour appears fragmented. Comparatively, `Result2.png` and `result3.png` are quite similar, except that the latter has a cleaner background due to the utilization of connected-component labeling.

(d) Edge crispening

To obtain `result5.png`, I applied unsharp masking with $L = 5$ and $c = 0.7$. Comparing it to `sample2.png`, it can be observed that the edges in the former image are more prominent than its neighboring elements.



(a) `sample2.png`

(b) `result5.png`

There are two parameters to consider: filter size L and weight c . From Table 2 we can see that the sharpening effect is stronger with larger L and smaller c .

	$c = 0.6$	$c = 0.7$	$c = 0.8$
$L = 3$			
$L = 5$			
$L = 7$			

Table 2: Unsharp masking with different filter size L and weight c .

(e) Hough transform



(a) result5.png



(b) result6.png

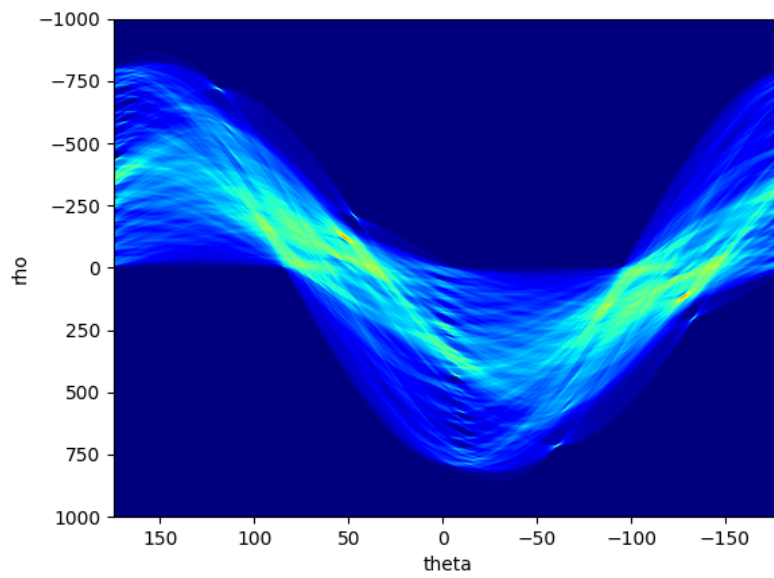


Figure 10: result7.png

Problem 2: GEOMETRICAL MODIFICATION

(a) Borzoi

The algorithm consists of three steps: segment, transform, and paste.

Segment. To apply transformations only to the borzoi, we must first separate it from the image. The segmentation problem can be treated as finding connected components on a two-dimensional grid. Specifically, we consider pixels with gray-level less than 255 as nodes. The criterion for edges can be either four-neighbors or eight-neighbors, which does not affect the outcome in this case. Once we have identified the nodes and edges, we can apply the depth-first-search algorithm to obtain the connected components, and the largest component would correspond to the borzoi.

Transform. Next, we paste the borzoi to an empty image and apply the following transformation sequentially:

1. scaling(0.9, 1.8)
2. translation(400, 100)
3. rotation(-40°)
4. translation(-450, -320)

Paste. Finally, paste the rest elements (i.e. the potato chips and the words) back to the new image.



(a) sample3.png

(b) result8.png

(b) Popdog

Given a point $c = (x_c, y_c)$ and constants k_x, k_y , the transformation from $F[u, v]$ to $G[x, y]$ is defined as follows:

$$\begin{aligned} u &= x + \min(r, (\frac{r_{max}}{r+1})^{k_x}) \cos \theta \\ v &= y + \min(r, (\frac{r_{max}}{r+1})^{k_y}) \sin \theta \end{aligned}$$

where

$$\begin{aligned} r &= \sqrt{(x - x_c)^2 + (y - y_c)^2} \\ \theta &= \arctan2(y - y_c, x - x_c) \\ r_{max} &= \max_{(x,y)} r(x, y) \end{aligned}$$

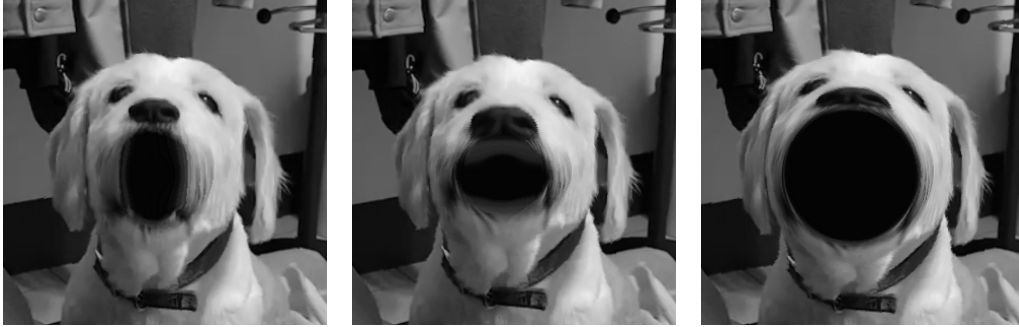
Specifically, I use $c = (148, 148)$ and $k_x = k_y = 2.3$ to produce `result9.png`.



(a) sample5.png

(b) result9.png

The idea of the above transformation is to shift along the direction towards the center point c by specific offset. The closer to the center, the effect of the shift becomes more noticeable. The coefficients k_x and k_y control the magnitude of the shift along the horizontal and vertical axes respectively, as illustrated in Figure 13.



(a) $(k_x, k_y) = (1, 2)$

(b) $(k_x, k_y) = (2, 1)$

(c) $(k_x, k_y) = (3, 3)$

Figure 13: Popdogs