ECE 847
Digital Image Processing

Homework 3
Canny Edge Detection

September 29, 2006

Name

Homework 3
Canny Edge Detection

Introduction

The goal of this assignment is to implement the Canny edge detector. This process involves three steps—gradient estimation (with smoothing), non-maximum suppression, and hysteresis thresholding. In addition, the Chamfer distance algorithm is implemented to find the correct placement of an image template on another image.

Theory

The Canny edge detector was designed to be the optimal edge detector. The two main goals are to minimize the error rate (i.e. the amount of false positives and false negatives) and to provide localized edges. The Canny edge detector simplifies the process by using convolutions with Gaussians.

The first step is to find the gradient in the x and y directions (g_x and g_y) by smoothing the image and taking the derivative. If a two-dimensional Gaussian G_{xy} is convolved with the image I for smoothing, the operation is given by

$$g_x = \frac{\partial}{\partial x}\left(I * G_{xy}\right) \text{ and } g_y = \frac{\partial}{\partial y}\left(I * G_{xy}\right).$$

The * operator denotes convolution. The form above is not the most computationally efficient method of computing the gradient. Using the fact that the Gaussian is separable, a more efficient form can be found as shown below.

$$g_x = \frac{\partial}{\partial x}\left(I * G_{xy}\right) = I * \frac{\partial}{\partial x}\left(G_{xy}\right) = I * \frac{\partial}{\partial x}\left(G_x * G_y\right) = I * \frac{\partial}{\partial x}\left(G_x\right) * G_y$$

Likewise, the gradient in the y direction is

$$g_y = I * G_x * \frac{\partial}{\partial y}\left(G_y\right).$$

The g_x and g_y give the gradient estimation in the x and y directions, respectively. Let $g_x(x, y)$ be the gradient in the x direction at (x, y), and $g_y(x, y)$ be the gradient in the y direction. The magnitude of the gradient $g(x, y)$ is given by

$$g(x,y) = \sqrt{g_x(x,y)^2 + g_y(x,y)^2}\ .$$

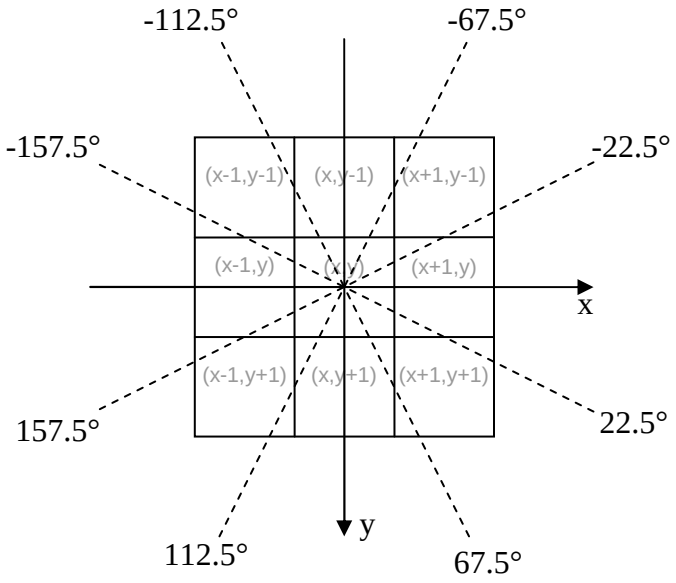
If processing time is a factor, the above equation can be approximated by

$$g(x,y) = |g_x(x,y)| + |g_y(x,y)|.$$

The direction of the maximum slope is the angle formed by the vector addition of the two gradients in the x and y directions. This angle $\Theta(x,y)$ is found from the following equation:

$$\theta(x,y) = \tan^{-1}\left[\frac{g_y(x,y)}{g_x(x,y)}\right]$$

The diagram below shows the angles that will be returned from the arctangent and the pixels to which these angles correspond.



For the non-maximum suppression, the neighboring pixels in the direction of the gradient are needed. To find these pixels, add 180° to Θ if Θ is negative and use the rules below.

Theta Values	*Direction*	*Pixels*
$\theta \le 22.5^\circ$ *or* $\theta > 157.5^\circ$	$\theta = 0^\circ$	$(x-1,y)$ $(x+1,y)$
$22.5^\circ < \theta \le 67.5^\circ$	$\theta = 45^\circ$	$(x-1,y-1)$ $(x+1,y+1)$
$67.5^\circ < \theta \le 112.5^\circ$	$\theta = 90^\circ$	$(x-1,y+1)$ $(x+1,y-1)$
$112.5^\circ < \theta \le 157.5^\circ$	$\theta = 135^\circ$	$(x,y+1)$ $(x,y-1)$

The next step is to perform non-maximum suppression on the image. The algorithm is to check every pixel in the image. If either of the two neighboring pixels in the direction of the gradient

are greater in magnitude than the current pixel then $I(x, y)=0$, else $I(x, y)=g(x, y)$. The effect of this operation is to create one-pixel thick lines along the local maxima of the gradient.

Once the non-maximum suppression has been performed on the image, the edge detector must differentiate between lines that are edges and lines which are due to noise. To make this distinction, the Canny edge detector uses hysteresis thresholding. This method of thresholding involves a high and a low threshold. All the pixels that have a gradient value above the high threshold will be retained. Similarly, all the pixels that have a gradient value below the low threshold will be rejected. The pixels with gradient values in between the high and low thresholds are retained only if the pixel is connected to another pixel with a gradient value above the high threshold. The output of the hysteresis thresholding produces the final image.

The Chamfer distance is a method of finding the distance to the edges in the image. The Manhattan distance is used in this assignment. The Manhattan distance D for two points $p=(p_x, p_y)$ and $q=(q_x, q_y)$ is given by

$$D(p,q) = |q_x - p_x| + |q_y - p_y| .$$

The Chamfer distance is used in this assignment to find the correct placement of a template image on the original image. This is accomplished by computing a probability map of the sum of the distances to the edges.

Algorithm

main

1. Open image file.
2. Call `canny_edge_detect` to detect the edges in the image. Display intermediate and final results.
3. Display a message box asking the user if the Chamfer distance should be computed and a template image should be opened. If no, stop the program, otherwise continue.
4. Compute the Chamfer distance by calling `chamfer_distance`.
5. Call `probability_map` to compute the probability map using the Chamfer distance and the template image.
6. Overlay a rectangle on the original image.

canny_edge_detect

1. Compute the gradient by calling the `gradient` procedure. The gradient in the x and y directions, along with magnitude and angle is returned.
2. Perform non-maximum suppression (`nonmaximum_suppression`) using the magnitude and direction of the gradient.
3. Use image statistics to compute the threshold values (`imag_stats`).

4. Apply the high and low thresholds. For all the pixels in the image if the value is above, the threshold value set to 255 else set to 0.
5. Use low and high threshold images to perform a double threshold (`double_threshold`).
6. Display results. Invert the image of the edges to show black on white instead of white on black.

chamfer_distance

1. For all the pixels in the image, set value to zero if an edge. Else, set the value to a high number (height of image + width of image + 1).
2. From left to right and top to bottom, find the minimum of the value of the current pixel, the value of the pixel to the left + 1, and the value of the pixel above + 1.
3. From right to left and bottom to top, find the minimum of the value of the current pixel, the value of the pixel to the right + 1, and the value of the pixel below + 1.

probability_map

1. Create a vector of all the points in the template.
2. Sum the distances to the edges for each possible position.
3. Output the probability map and the minimum sum of the distances.

gradient

1. Compute the Gaussian and the Gaussian derivative kernels.
2. Smooth the image with the Gaussian in the y direction and convolve the image with the Gaussian derivative in the x direction. The result is the gradient in the x direction.
3. Smooth the image with the Gaussian in the x direction and convolve the image with the Gaussian derivative in the y direction. The result is the gradient in the y direction.
4. Compute the magnitude and direction of the gradient.

nonmaximum_suppression

1. For all pixels in the image, if the pixel is not maximum in the direction of the gradient, set it to zero.
2. Output an image with one-pixel thick edges.

imag_stats

1. Create a histogram of the values in the image.
2. Starting from the maximum value loop back until 8% of the pixels are found. Set this value as the high threshold.
3. Set the low threshold to 35% of the high threshold.

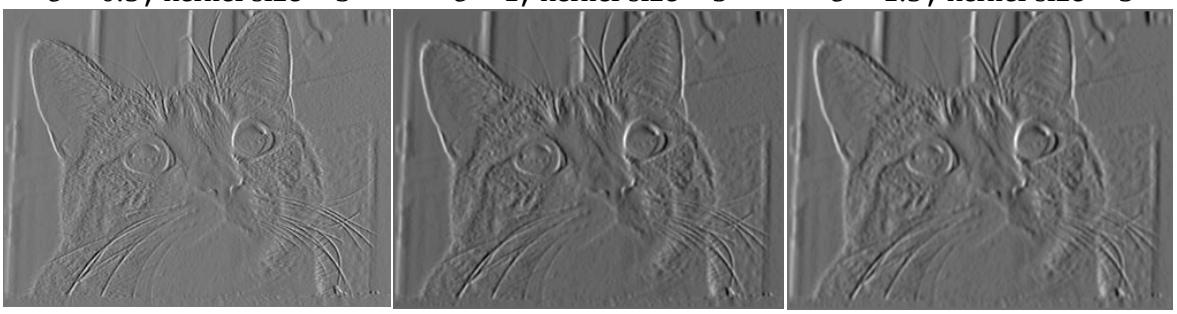
See the source code for descriptions of the parameters required for each procedure.

Results

cat.pgm

original image

$\sigma = 0.5$, kernel size = 3 $\sigma = 1$, kernel size = 5 $\sigma = 1.5$, kernel size = 5



x gradient

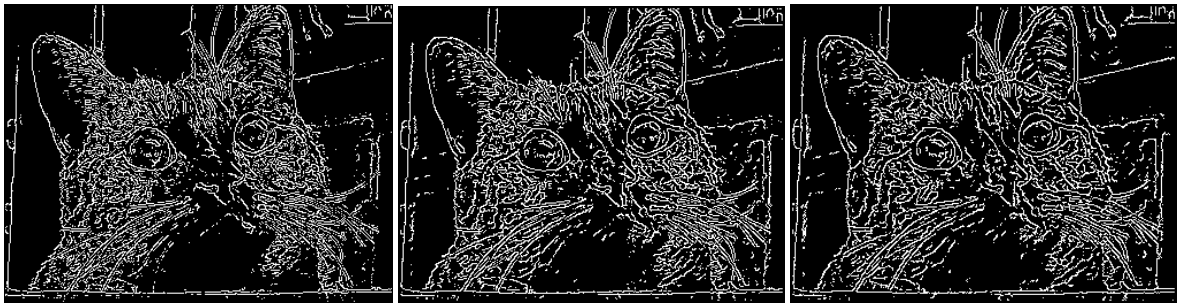
y gradient

magnitude of gradient

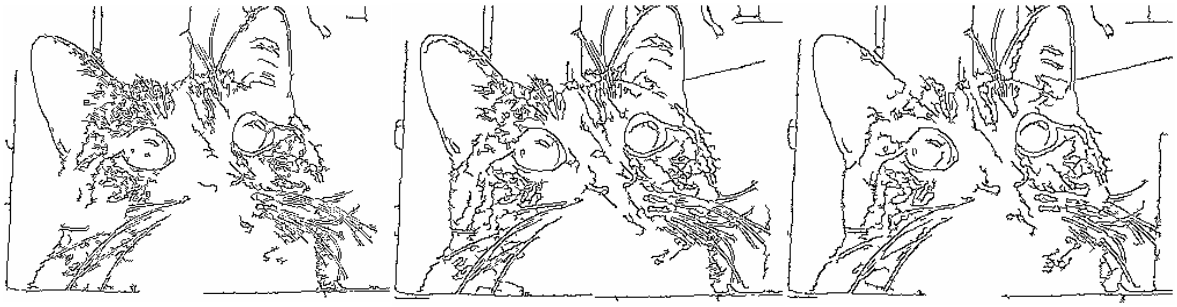
direction of gradient

non-maximal suppression

high-threshold image



low-threshold image

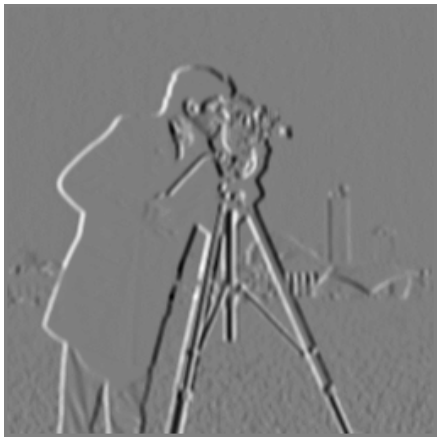


edges

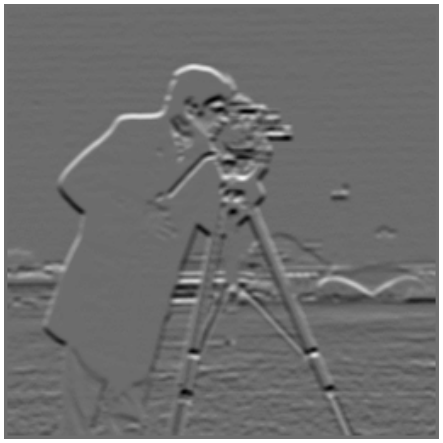
All the results below are for $\sigma = 1.5$ and a kernel size of 5.

cameraman.pgm

original image



x gradient

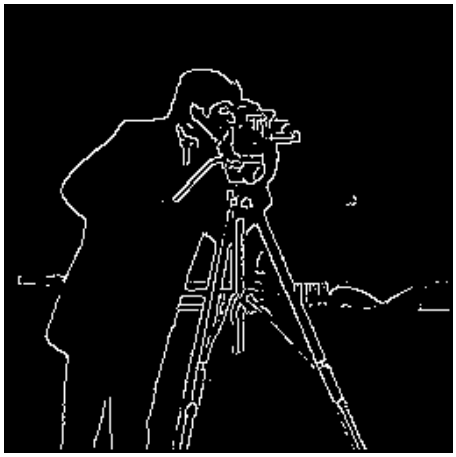


y gradient

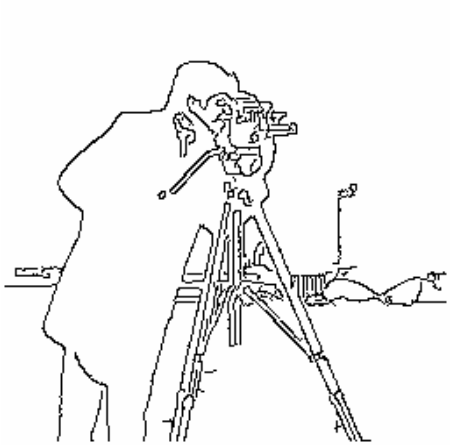
magnitude of gradient


direction of gradient


non-maximal suppression


high-threshold image

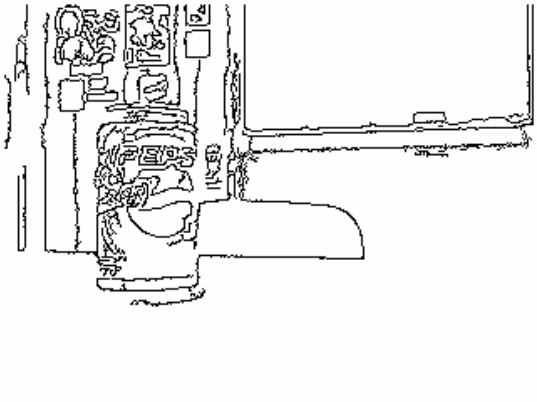

low-threshold image


edges

Chamfer distance

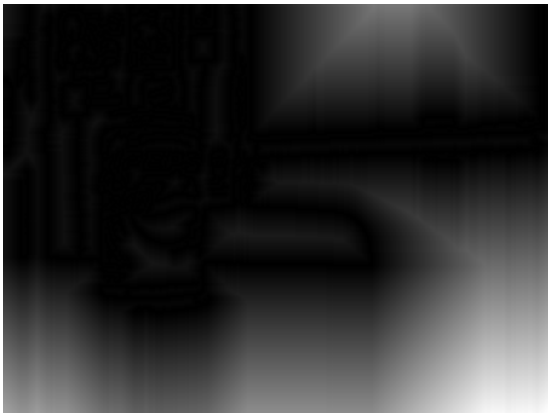
original image

template image



edges

edges of template image

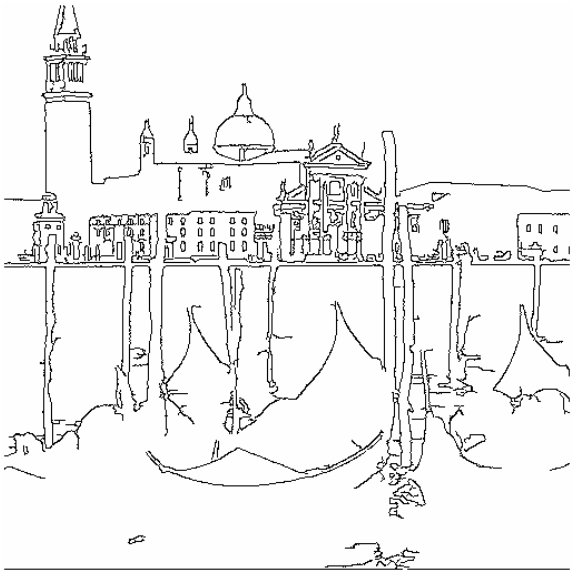


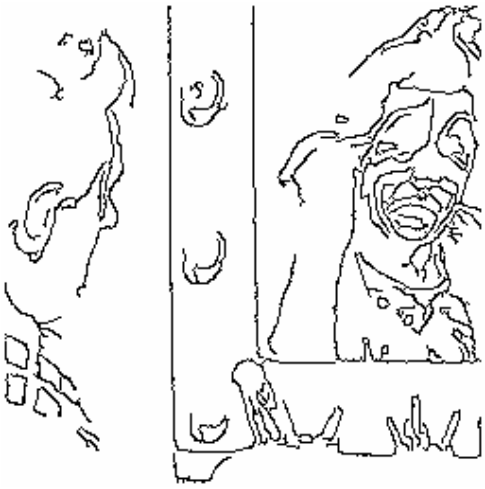
Chamfer distance

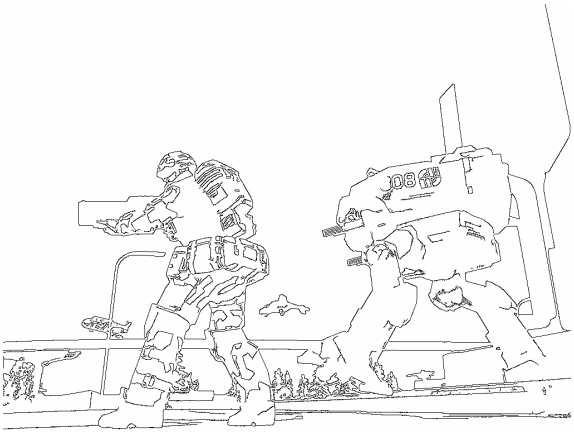
probability map

original image with rectangle

Other images







Conclusion

The Canny edge detector implemented in this assignment works well with a variety of input images. Naturally, the best detection is for images that have high contrast. The last image of the results shows a scene from a video game. Since the image is computer generated, the image has very distinct edges and the edge detector works well. On a more natural image with less contrast, such as the picture of Venice in the results, the edges are not clearly defined. However, the edges that are detected are acceptable.

Various factors affect whether or not an edge is detected. For instance, applying different σ values and kernel sizes produces different results. For larger values of σ, the gradient images are smoother. Larger σ values require a larger kernel size for a good Gaussian kernel. The values used for thresholding with hysteresis also affect the edges that are detected. In this implementation, a value for the high threshold is selected so that 8% will remain after the high threshold is applied. The low threshold is 35% of the high threshold.

Using Chamfer distance to determine the placement of the template image on the original image was successful. The method was tested and was also successful with different σ values and different threshold values. As long as the template is the same size as the object in the original image, the algorithm works well.