



(a) Original



(b) Roberts Cross Operator

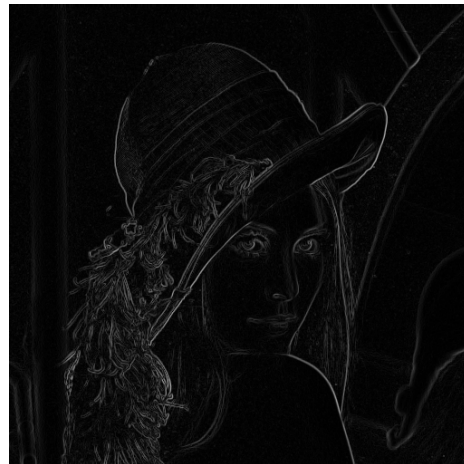


(c) Sobel Operator

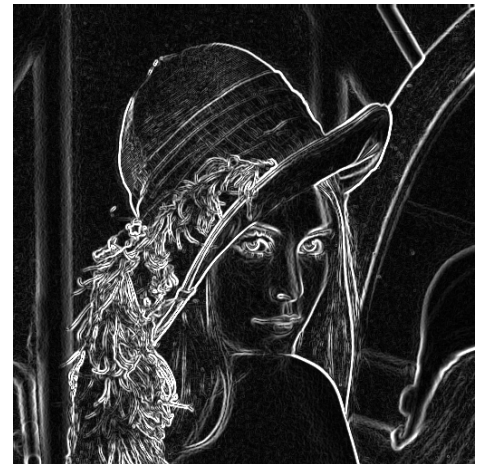
Figure 1: Cameraman



(a) Original

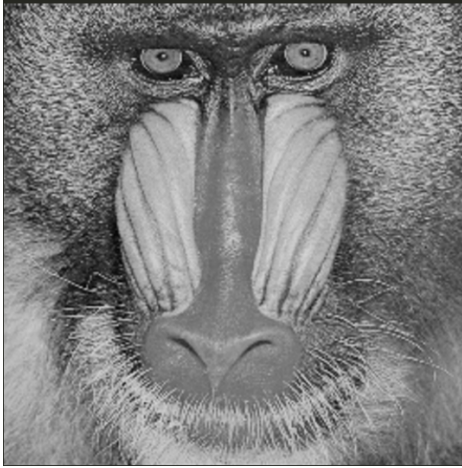


(b) Roberts Cross Operator



(c) Sobel Operator

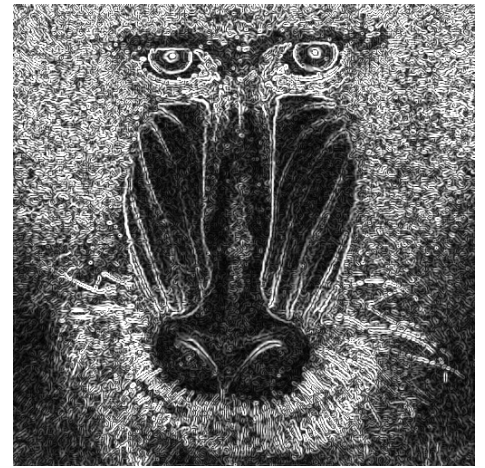
Figure 2: Lena



(a) Original

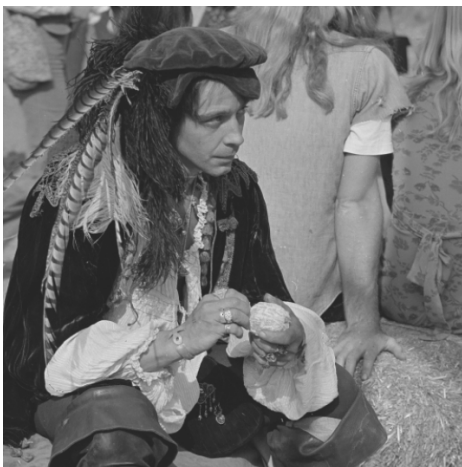


(b) Roberts Cross Operator



(c) Sobel Operator

Figure 3: Mandril



(a) Original



(b) Roberts Cross Operator



(c) Sobel Operator

Figure 4: Pirate

```

1  /*
2      File: edge_detection.h
3  */
4
5  /* C Includes */
6  #include <stdio.h>
7
8  /* C++ Includes */
9  #include <cmath>
10 #include <fstream>
11 #include <iostream>
12 #include <string>
13 #include <vector>
14 #include <utility>
15
16 /* Project Includes */
17 #include "convert/cameraman.h"
18 #include "convert/lena.h"
19 #include "convert/mandril.h"
20 #include "convert/pirate.h"
21
22 /* Output Directory */
23 static const std::string OUTPUT_DIR = "../output/";
24
25 /* Input Image Info */
26 typedef std::pair<std::string, const unsigned char *> Image;
27 static const std::vector<Image> IMAGES = {
28     Image("cameraman", cameraman_image),
29     Image("lena", lena_image),
30     Image("mandril", mandril_image),
31     Image("pirate", pirate_image)
32 };
33
34 /* Input Image Dimensions */
35 static const int IMAGE_WIDTH = 512;
36 static const int IMAGE_HEIGHT = 512;
37 static const int PIXEL_COUNT = IMAGE_WIDTH * IMAGE_HEIGHT;
38
39 /* Sobel Operator */
40 static const int SOBEL_V[3][3] = {{-1, 0, 1},
41                                     {-2, 0, 2},
42                                     {-1, 0, 1}};
43 static const int SOBEL_H[3][3] = {{-1, -2, -1},
44                                     {0, 0, 0},
45                                     {1, 2, 1}};
46
47 /* Roberts Cross Operators */
48 static const int ROBERTS_V[2][2] = {{1, 0},
49                                       {0, -1}};
50 static const int ROBERTS_H[2][2] = {{0, 1},
51                                       {-1, 0}};
52
53 /* Helper Functions */
54 int sobel(const bool, const int, const int, int **);
55 int roberts(const bool, const int, const int, int **);
56 void write_pgm(const std::string&, int **);
57

```



```

1  /*
2      File: main.cpp
3  */
4  #include "edge_detection.h"
5
6  int main(void) {
7      /* Allocate memory to hold input/output images. */
8      int **input_image = new int*[IMAGE_HEIGHT];
9      int **output_image = new int*[IMAGE_HEIGHT - 2];
10     for (int i = 0; i < IMAGE_HEIGHT; i++) {
11         input_image[i] = new int[IMAGE_WIDTH];
12         output_image[i] = new int[IMAGE_WIDTH - 2];
13     }
14
15     /* Iterate through each image. */
16     for (Image image_pair : IMAGES) {
17         /* Unpack image parameters. */
18         std::string image_label = image_pair.first;
19         const unsigned char *image = image_pair.second;
20
21         /* Read flat 1D image array to 2D array. */
22         for (int i = 0; i < IMAGE_HEIGHT; i++) {
23             for (int j = 0; j < IMAGE_WIDTH; j++) {
24                 input_image[i][j] = static_cast<int>(image[i * IMAGE_HEIGHT + j]);
25             }
26         }
27
28         /* Detect image's edges using the Sobel operator. */
29         for (int i = 1; i < IMAGE_HEIGHT - 1; i++) {
30             for (int j = 1; j < IMAGE_WIDTH - 1; j++) {
31                 int V = sobel(true, i, j, input_image);
32                 int H = sobel(false, i, j, input_image);
33                 output_image[i - 1][j - 1] = static_cast<int>(round(sqrt(pow(V, 2) + pow(H,
34 2))));
35             }
36         }
37         /* Generate PGM file for Sobel output. */
38         write_pgm(image_label + "_sobel", output_image);
39
40         /* Detect image's edges using the Roberts cross. */
41         for (int i = 1; i < IMAGE_HEIGHT - 1; i++) {
42             for (int j = 1; j < IMAGE_WIDTH - 1; j++) {
43                 int V = roberts(true, i, j, input_image);
44                 int H = roberts(false, i, j, input_image);
45                 output_image[i - 1][j - 1] = static_cast<int>(round(sqrt(pow(V, 2) + pow(H,
46 2))));
47             }
48         }
49         /* Generate PGM file for Roberts output. */
50         write_pgm(image_label + "_roberts", output_image);
51     }
52     return 0;
53 }
54
55 int sobel(const bool vertical, const int i, const int j, int **I) {
56     if (vertical) {
57         return I[i - 1][j - 1] * SOBEL_V[0][0]

```

```

57     + I[i - 1][j] * SOBEL_V[0][1]
58     + I[i - 1][j + 1] * SOBEL_V[0][2]
59     + I[i][j - 1] * SOBEL_V[1][0]
60     + I[i][j] * SOBEL_V[1][1]
61     + I[i][j + 1] * SOBEL_V[1][2]
62     + I[i + 1][j - 1] * SOBEL_V[2][0]
63     + I[i + 1][j] * SOBEL_V[2][1]
64     + I[i + 1][j + 1] * SOBEL_V[2][2];
65 }
66 else {
67     return I[i - 1][j - 1] * SOBEL_H[0][0]
68         + I[i - 1][j] * SOBEL_H[0][1]
69         + I[i - 1][j + 1] * SOBEL_H[0][2]
70         + I[i][j - 1] * SOBEL_H[1][0]
71         + I[i][j] * SOBEL_H[1][1]
72         + I[i][j + 1] * SOBEL_H[1][2]
73         + I[i + 1][j - 1] * SOBEL_H[2][0]
74         + I[i + 1][j] * SOBEL_H[2][1]
75         + I[i + 1][j + 1] * SOBEL_H[2][2];
76 }
77 }
78
79 int roberts(const bool vertical, const int i, const int j, int **I) {
80     if (vertical) {
81         return I[i][j] * ROBERTS_V[0][0]
82             + I[i][j + 1] * ROBERTS_V[0][1]
83             + I[i + 1][j] * ROBERTS_V[1][0]
84             + I[i + 1][j + 1] * ROBERTS_V[1][1];
85     }
86     else {
87         return I[i][j] * ROBERTS_H[0][0]
88             + I[i][j + 1] * ROBERTS_H[0][1]
89             + I[i + 1][j] * ROBERTS_H[1][0]
90             + I[i + 1][j + 1] * ROBERTS_H[1][1];
91     }
92 }
93
94 void write_pgm(const std::string& label, int **W) {
95     std::cout << OUTPUT_DIR + label + ".pgm" << std::endl;
96     std::ofstream pgm(OUTPUT_DIR + label + ".pgm");
97
98     if (!pgm) {
99         std::cerr << "Error occurred opening " << label << std::endl;
100         return;
101     }
102
103     pgm << "P2" << std::endl; // Header
104     pgm << IMAGE_WIDTH - 2 << " " << IMAGE_HEIGHT - 2 << std::endl; // Dimensions
105     pgm << "255" << std::endl; // Maximum Value
106
107     for (int i = 0; i < IMAGE_HEIGHT - 2; i++) {
108         for (int j = 0; j < IMAGE_WIDTH - 2; j++) {
109             pgm << W[i][j] << " ";
110         }
111         pgm << std::endl;
112     }
113 }
114

```