

CSC420 Assignment 2

Gabriel Luong
996268275

1

```
1 % Given an image, perform Harris corner detection and return the result image
2 function harris(im)
3 img = rgb2gray(im);
4
5 [Ix, Iy] = imgradientxy(img); % Compute the gradients Ix and Iy
6 Ix2 = Ix.^2;
7 Iy2 = Iy.^2;
8 Ixy = Ix.*Iy;
9
10 % Convolve the computed gradients with a Gaussian filter
11 gaussian = fspecial('gaussian');
12 gIx2 = imfilter(Ix2, gaussian, 'same', 'conv');
13 gIy2 = imfilter(Iy2, gaussian, 'same', 'conv');
14 gIxy = imfilter(Ixy, gaussian, 'same', 'conv');
15
16 height = size(im, 1);
17 width = size(im, 2);
18 alpha = 0.06;
19 R = zeros(height, width);
20
21 % Compute M and R for every pixel
22 for i = 1:height
23     for j = 1:width
24         M = [gIx2(i, j) gIxy(i, j); ...
25              gIxy(i, j) gIy2(i, j)];
26         detM = (gIx2(i, j) * gIy2(i, j)) - gIxy(i, j)^2;
27         traceM = gIx2(i, j) + gIy2(i, j);
28         R(i, j) = detM - (alpha * traceM ^ 2);
29     end
30 end
31
32 % Keep track of the points to plot
33 X = [];
```

```

34 Y = [];
35
36 threshold = max(max(R)) * 0.03; % R threshold
37
38 % Perform non-maximum suppression by checking surrounding R values for a
39 % local maxima with a radius of 1
40 for i = 2:height - 1
41     for j = 2:width - 1
42         if R(i, j) > threshold && R(i, j) > R(i - 1, j + 1) && R(i, j) > R(i, j + 1) &&
43             R(i, j) > R(i + 1, j + 1) && R(i, j) > R(i - 1, j) && R(i, j) > R(i + 1, j) &&
44             R(i, j) > R(i - 1, j - 1) && R(i, j) > R(i, j - 1) && R(i, j) > R(i - 1, j + 1)
45             X(end+1) = j;
46             Y(end+1) = i;
47         end
48     end
49 end
50
51 % Display the results of the computations
52 figure, imshow(Ix, []), title('Ix')
53 figure, imshow(Iy, []), title('Iy')
54 figure, imshow(Ix2, []), title('Ix2')
55 figure, imshow(Iy2, []), title('Iy2')
56 figure, imshow(Ixy, []), title('Ixy')
57 figure, imshow(gIx2, []), title('gIx2')
58 figure, imshow(gIy2, []), title('gIy2')
59 figure, imshow(gIxy, []), title('gIxy')
60 figure, imshow(R, []), title('R');
61 figure, imshow(img), title('corners');
62 hold on;
63 plot(X, Y, 'R+');
64 end

```

(a) Compute I_x^2 , I_y^2 , $I_x I_y$ and the matrix M for every pixel.

```

1 img = rgb2gray(im);
2
3 [Ix, Iy] = imgradientxy(img); % Compute the gradients Ix and Iy
4 Ix2 = Ix.^2;
5 Iy2 = Iy.^2;
6 Ixy = Ix.*Iy;
7
8 % Convolve the computed gradients with a Gaussian filter
9 gaussian = fspecial('gaussian');
10 gIx2 = imfilter(Ix2, gaussian, 'same', 'conv');
11 gIy2 = imfilter(Iy2, gaussian, 'same', 'conv');
12 gIxy = imfilter(Ixy, gaussian, 'same', 'conv');
13

```

```

14 height = size(im, 1);
15 width = size(im, 2);
16 alpha = 0.06;
17 R = zeros(height, width);
18
19 % Compute M and R for every pixel
20 for i = 1:height
21     for j = 1:width
22         M = [gIx2(i, j) gIxy(i, j); gIxy(i, j) gIy2(i, j)];
23     end
24 end

```

(b) Given the matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, the determinant $\det(A) = ad - cb$, and $\text{trace}(A) = a + d$. The equation to compute the corneriness measure R in each pixel is $R = \det(M) - \alpha \text{trace}(M)^2 = ad - cb - \alpha(a + d)^2$.

(c) Compute R for every pixel.

```

1 % Convolve the computed gradients with a Gaussian filter
2 gaussian = fspecial('gaussian');
3 gIx2 = imfilter(Ix2, gaussian, 'same', 'conv');
4 gIy2 = imfilter(Iy2, gaussian, 'same', 'conv');
5 gIxy = imfilter(Ixy, gaussian, 'same', 'conv');
6
7 height = size(im, 1);
8 width = size(im, 2);
9 alpha = 0.06;
10 R = zeros(height, width);
11
12 % Compute M and R for every pixel
13 for i = 1:height
14     for j = 1:width
15         M = [gIx2(i, j) gIxy(i, j); ...
16             gIxy(i, j) gIy2(i, j)];
17         detM = (gIx2(i, j) * gIy2(i, j)) - gIxy(i, j)^2;
18         traceM = gIx2(i, j) + gIy2(i, j);
19         R(i, j) = detM - (alpha * traceM ^ 2);
20     end
21 end

```

(d) Compute Threshold R and perform non-maxima suppression.

```

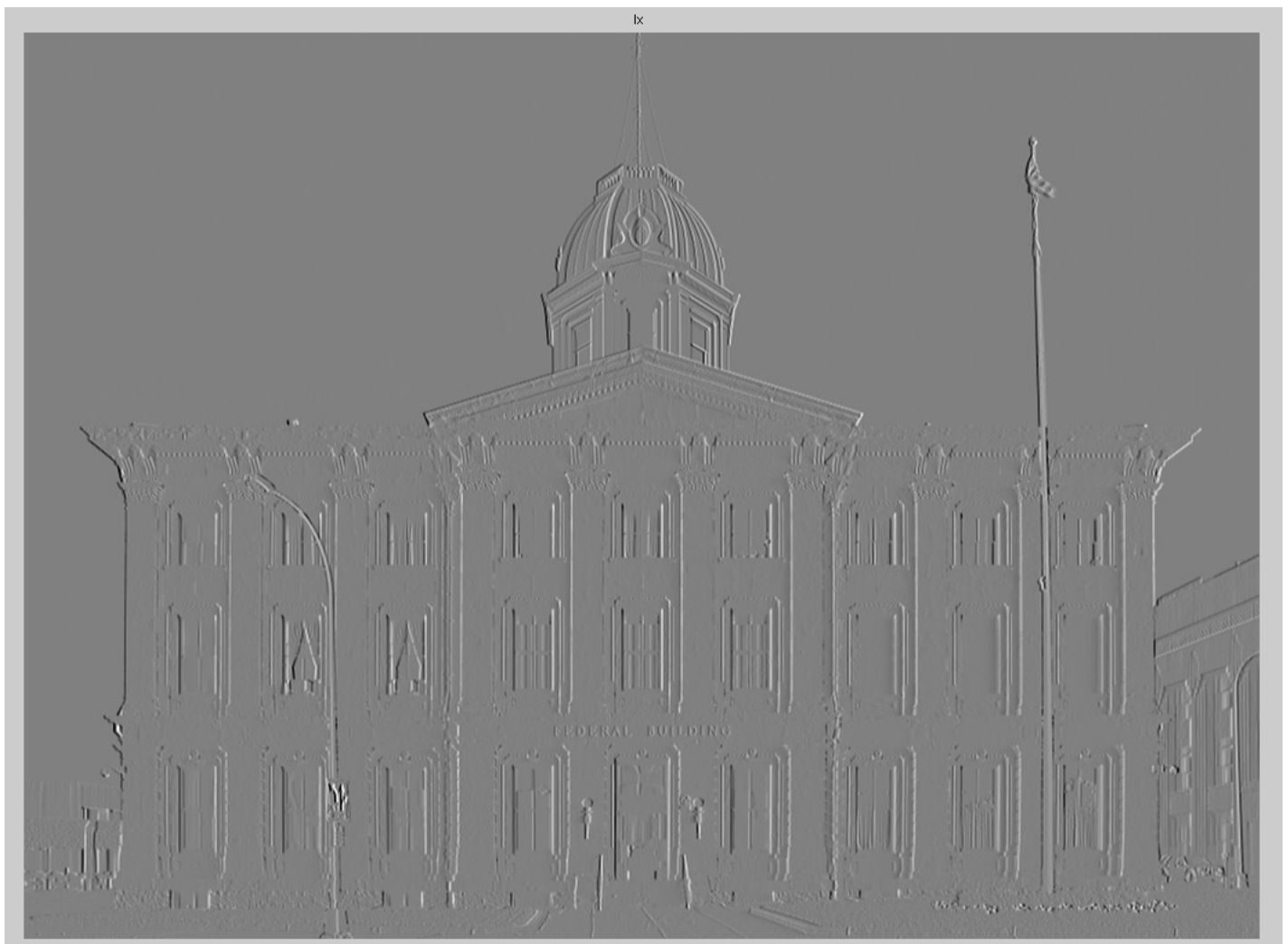
1 % Keep track of the points to plot
2 X = [];
3 Y = [];
4

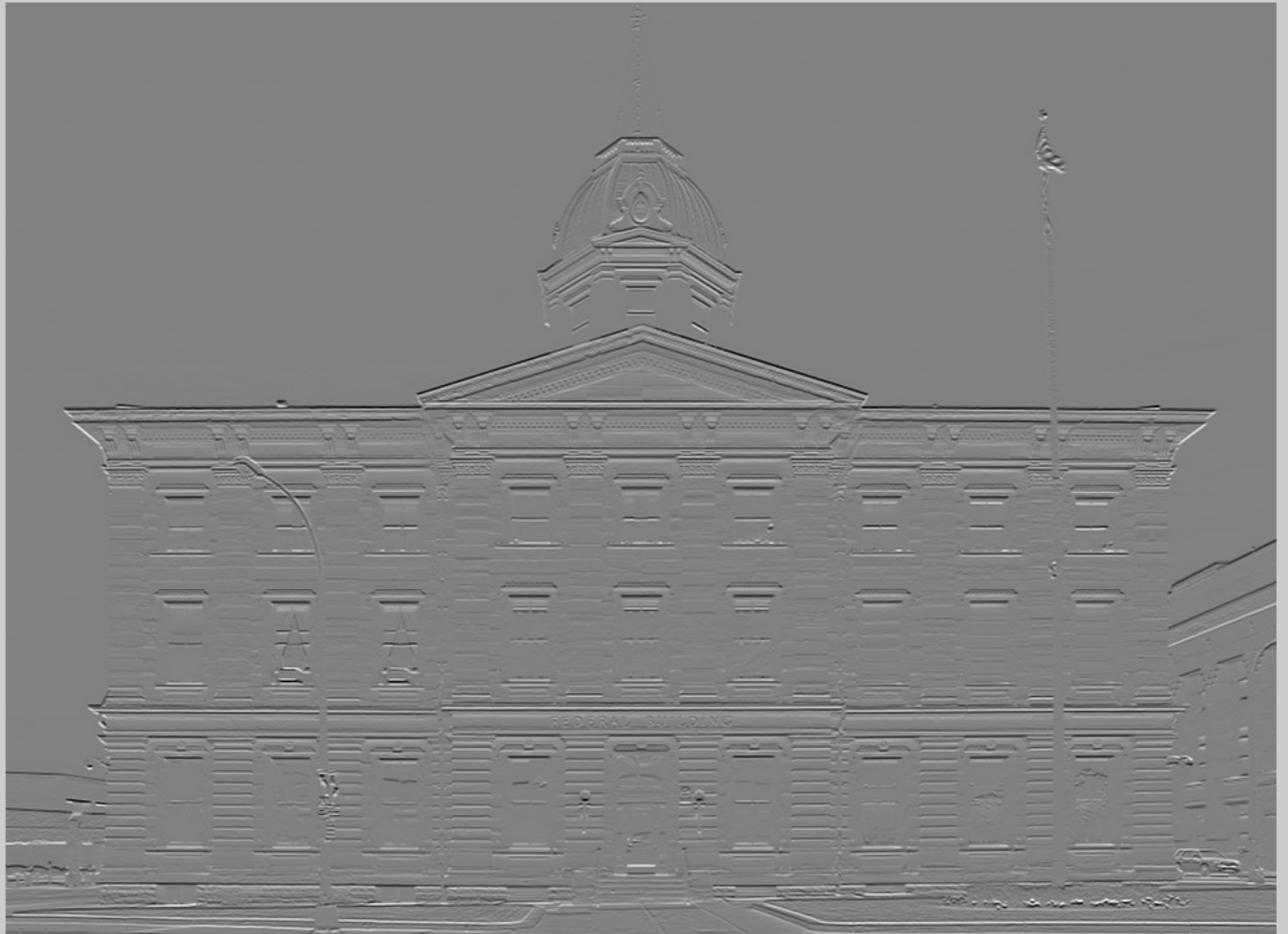
```

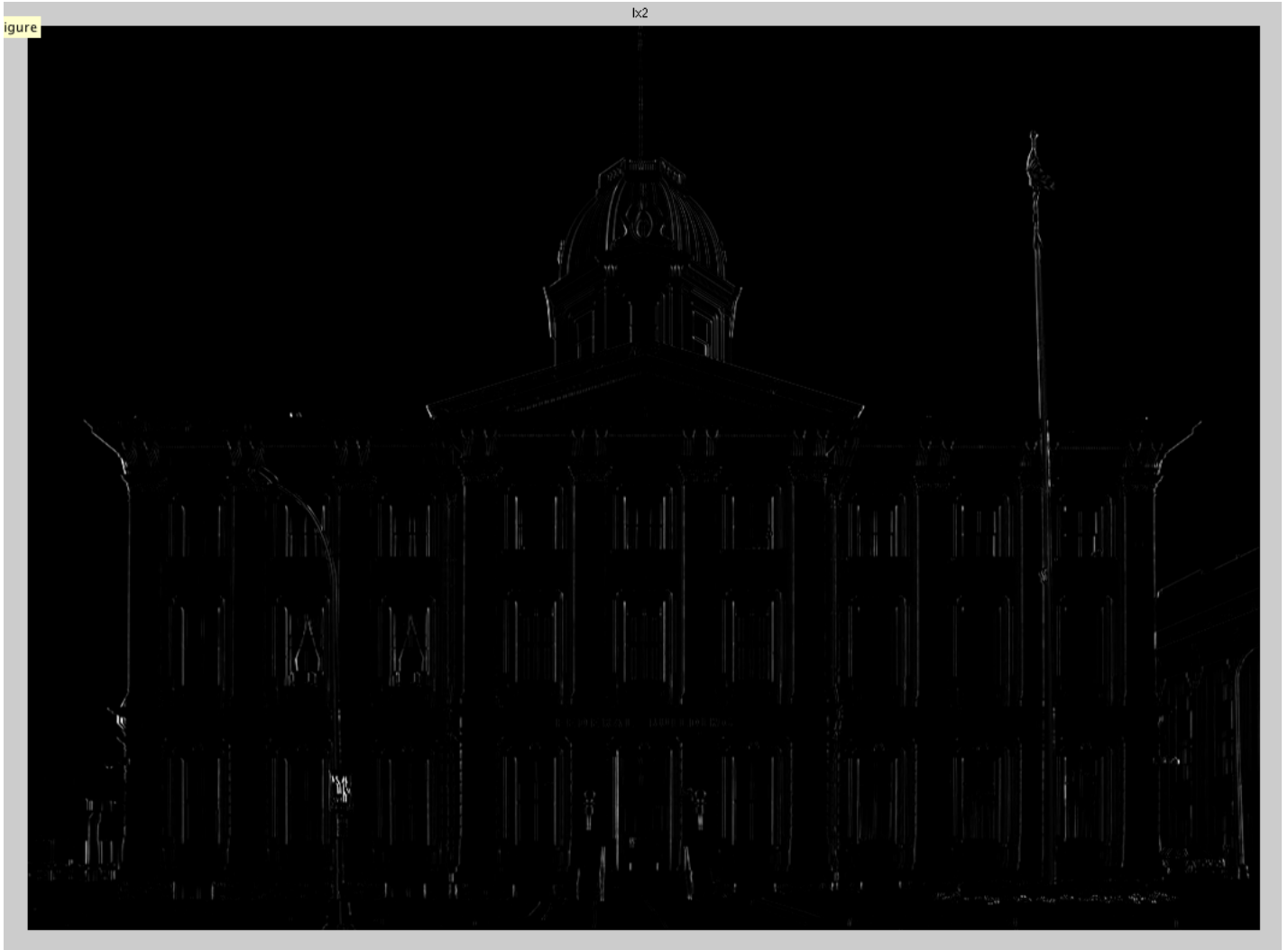
```

5 threshold = max(max(R)) * 0.03; % R threshold
6
7 % Perform non-maximum suppression by checking surrounding R values for a
8 % local maxima with a radius of 1
9 for i = 2:height - 1
10     for j = 2:width - 1
11         if R(i, j) > threshold && R(i, j) > R(i - 1, j + 1) && R(i, j) > R(i, j + 1) &&
12             R(i, j) > R(i + 1, j + 1) && R(i, j) > R(i - 1, j) && R(i, j) > R(i + 1, j) &&
13             R(i, j) > R(i - 1, j - 1) && R(i, j) > R(i, j - 1) && R(i, j) > R(i - 1, j + 1)
14             X(end+1) = j;
15             Y(end+1) = i;
16         end
17     end
18 end
19
20 figure, imshow(img), title('corners');
21 hold on;
22 plot(X, Y, 'r+');

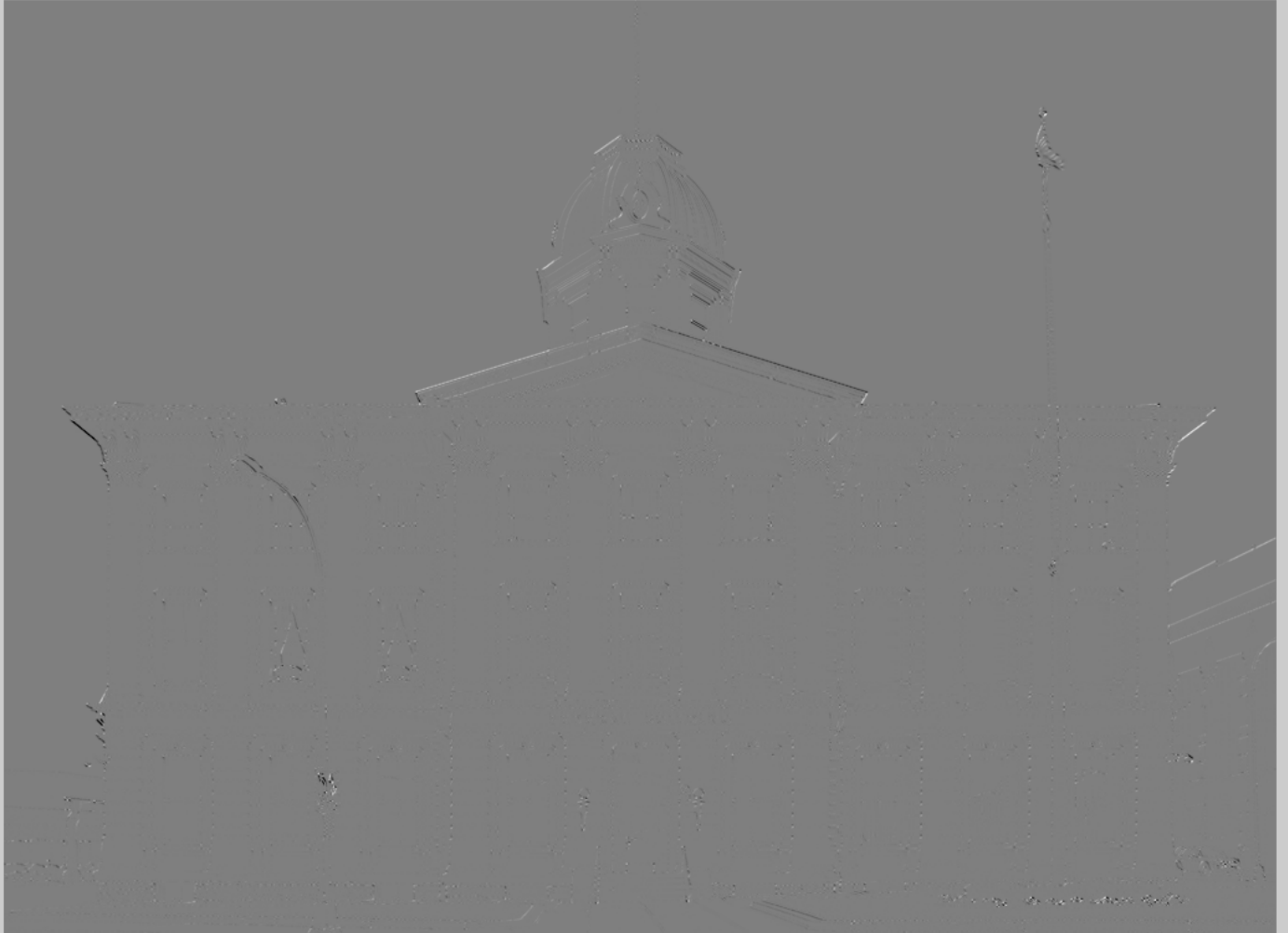
```

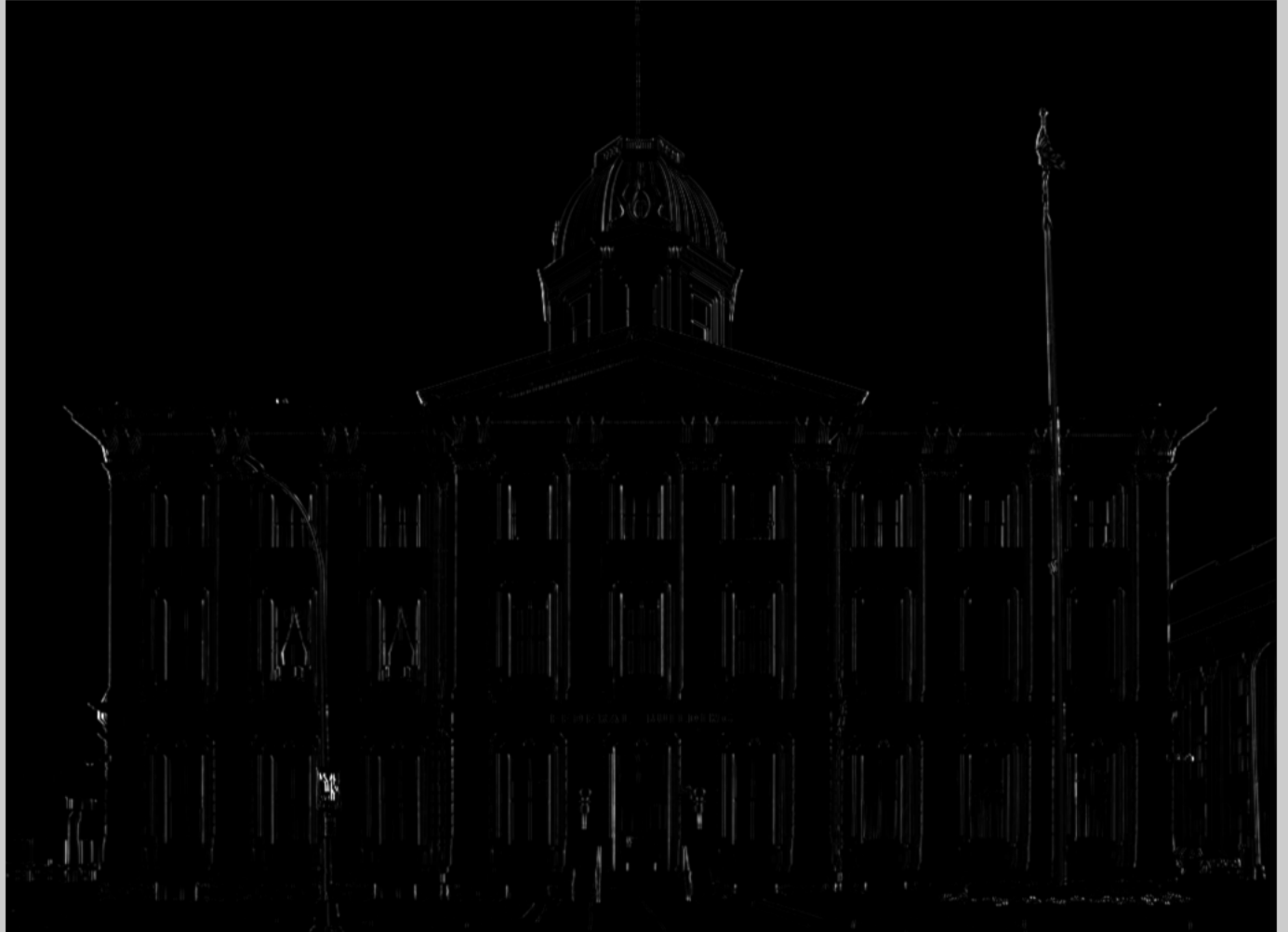




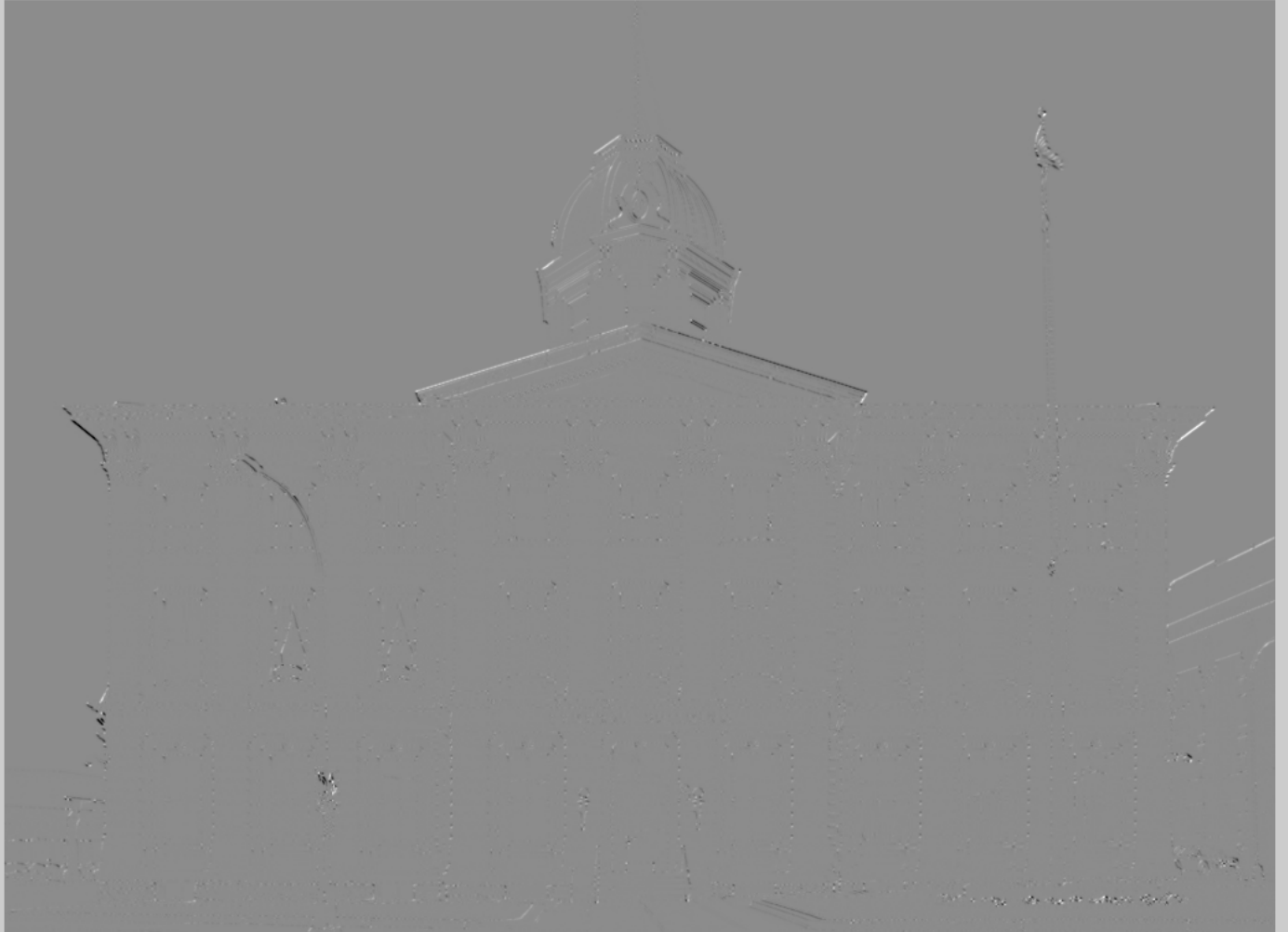


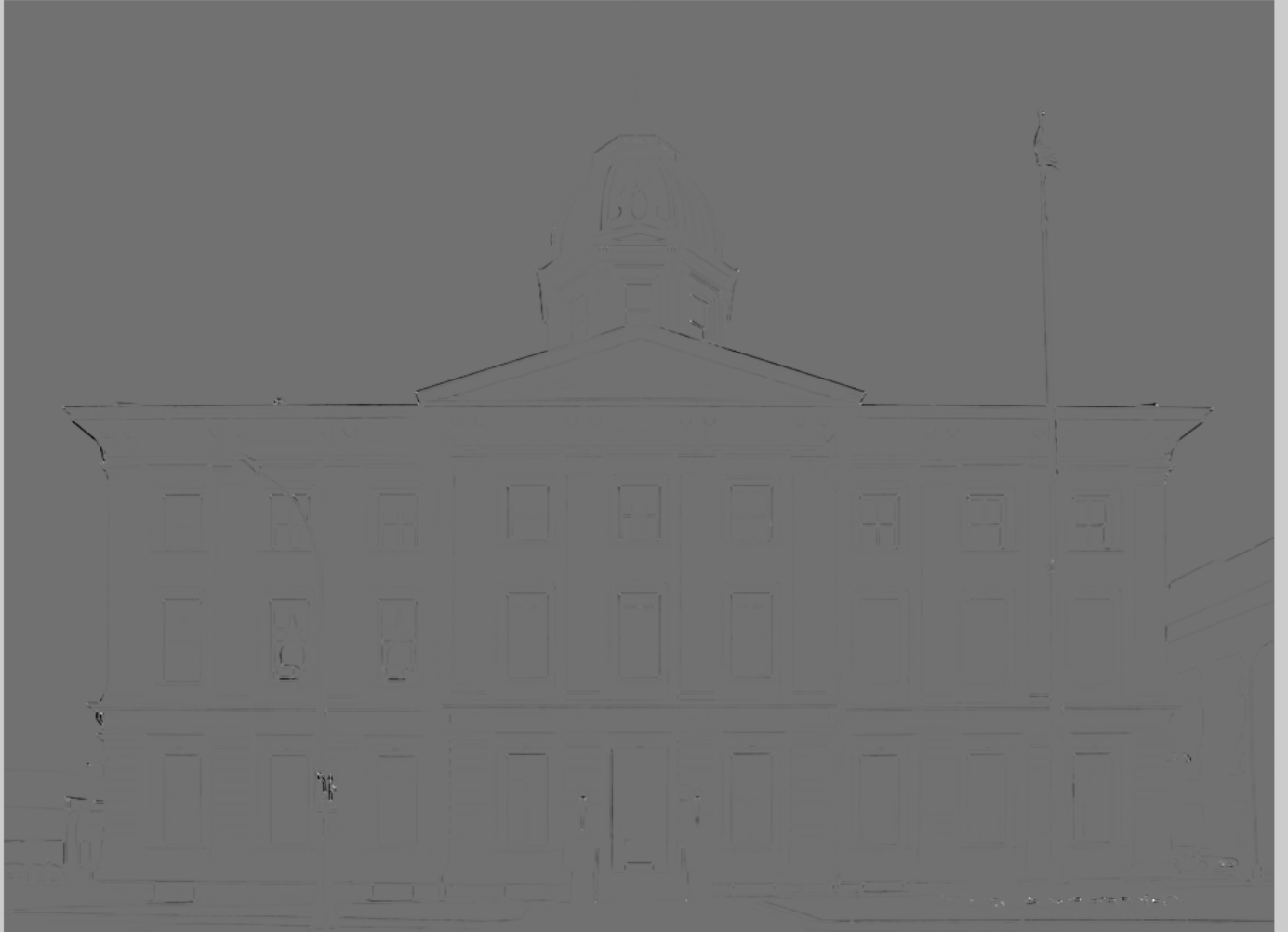














2