1.

The code for this part is contained in process\_image\_p1.c

First I define all the Sobel templates in a list and use an index to reference them. I made a function called apply\_sobel\_template which uses a given template, the image, and an image x and y location. This function is used to calculate values for an intermediate temp\_image. Each template is applied by multiplying the values of the template with the values of the image and summing everything up.

After the raw values for all the pixels are calculated, the values are normalized linearly between the min and max values. The values have the range of 0-255. These values are then put into the proc\_image.

Finally, the size of the image is changed but this causes graphical artifacting due to a change in size. I talked to the professor and he said this is ok.

Vert:

A white cubes with a cylinder

Description automatically generatedA greyscale shot of a building

Description automatically generated

Hor:

A close-up of a grid

Description automatically generatedA greyscale shot of a building

Description automatically generated

Maj. Diag:

A greyscale shot of a cylinder

Description automatically generatedA building with trees and a circular walkway

Description automatically generated with medium confidence

Min. diag:

A 3d model of a cylinder

Description automatically generatedA building with trees and a circular fountain

Description automatically generated with medium confidence

2.

The code for this part is contained in process\_image\_p2.c

Part two works very similarly as part one. Now instead of applying a sobel template, the code has changed to handle different sized templates.

A temp\_image is used to hold the raw image data. This data is retrieved by applying the apply\_selected\_template function to each applicable pixel. The selected template is normalized by subtracting the mean of the template, this way the total of all the values in the template are equal to 0. Additionally I scaled by the standard deviation of the template to standardize different templates.

Same as before, the temp\_image data is then used and resiszed to be within the range of 0-255. This is done by scaling linearly between the min and max values in the temp image

Examples:

Selected A:

A computer generated image of a cylinder

Description automatically generated

You can see that the A letter has a center white dot along with ghosting lines that look like an x. This is because the template and the letter line up along the left and right angled supports (like / and \)

Selected B:

A transparent object with a cylinder on it

Description automatically generated with medium confidence

You can see the tripple B ghosting due to the template being able to match with the letter 3 times. This is because B can match with itself in the top half, bottom half, and in both halves.

Selected Window:

A blurry image of a building

Description automatically generated