Christmas Light Placer

Micah Fierro and David Bair

December 2, 2021

Edge Detection

For the primary edge detection, we chose to use a modified Sobel filter. We used a single filter focused on horizontal edges. The filter looked for strong responses in the horizontal or x direction while not responding to changes in the vertical or y direction. We also made it so the overall sum of the filter was less than zero intentionally to remove any weak responses from the image. The result was a near binary image where only the strongest horizontal responses were left.

Originally, we tried to only detect perfectly horizontal edges, however, even after trying several variations of the Sobel filter, edges up to 45 degrees continued to appear in the image.

Selecting the best edges

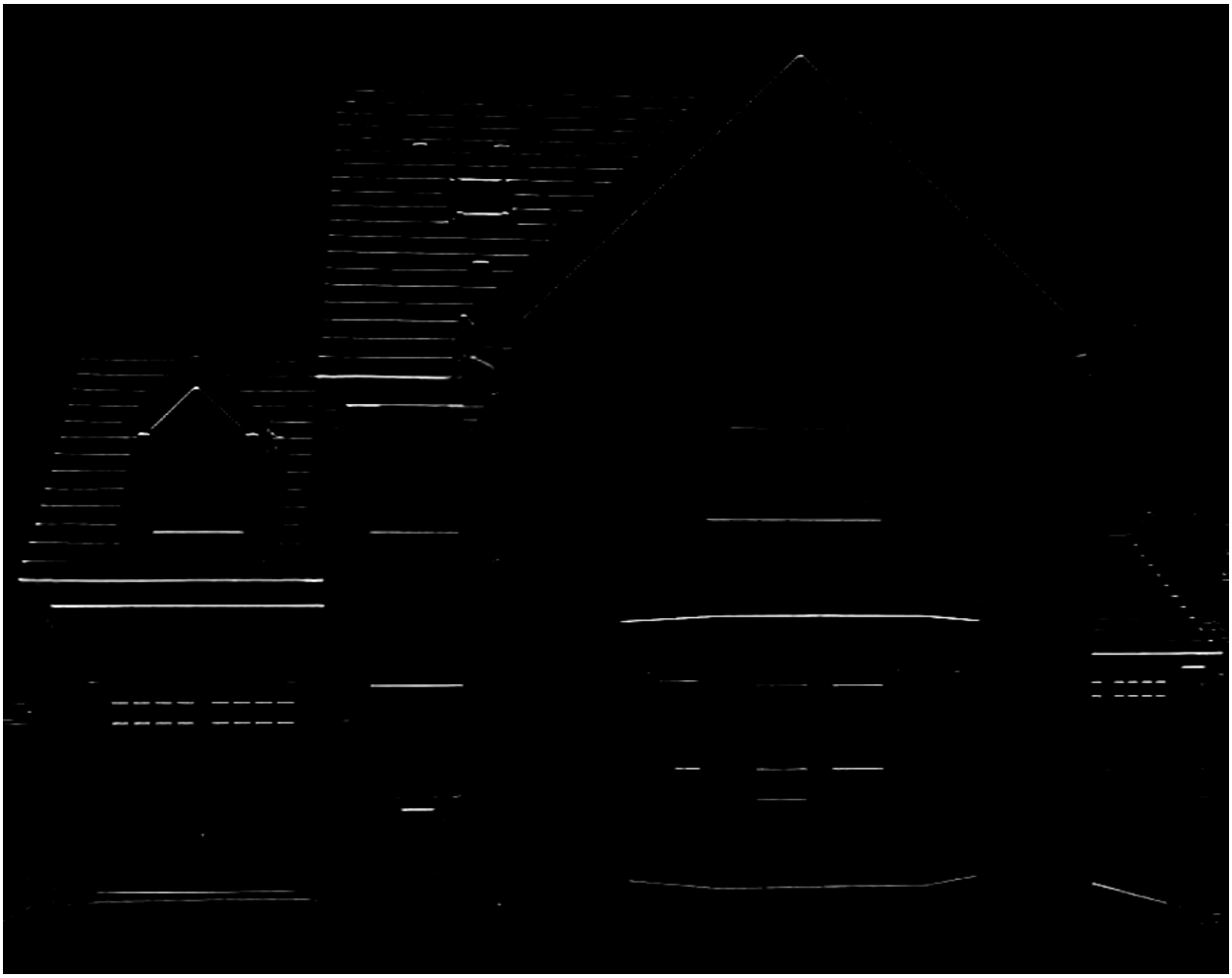
We then passed this image to the next function to determine the best edges to apply Christmas lights. To determine the best edges, we need to be able to determine what pixel went to which edges. For this task we used connected components technique. Using 8 bit connectivity we were able to generate a label mask that mapped each pixel to an edge. Using this we then only looked at the top ten percent of edges comparing their length. This allowed us to determine the best edges and the edges that most often correspond to edges along the roof of the house. We then saved the list of pixel locations for each edge.

Originally since we anticipated only dealing with horizontal edges with a variance of only a few degrees we attempted to only save the left-most pixel location and the right-most pixel location. These could then be passed on to generate the Christmas lights. However, we ran into the

previously mentioned issue of 45-degree edges showing up in the image. This was common due to the construction of triangle points on a house. To solve this issue, we attempted to pass the left-most pixel location, the right-most pixel location, and the pixel location with the greatest y value or highest location. This would accommodate for any triangles. This however, also caused an issue for any double triangle roof edges. In the end we passed a single list containing a list for each edge with each pixel location in the edge and processed the light locations in a separate function.

Reducing Image Sizes

It was noticed while using the edge detection algorithm on large images that many noisy edges were detected on the roofs of houses due to shingles. First, an attempt was made to use gaussian smoothing before using the modified sobel filter. This however resulted in the loss of some important edges during the detection step. No such issue was found on smaller images. To solve the problem, larger images were first reduced in size then the modified sobel algorithm was used. After the edges were detected, they were then scaled back up to the size of the original image.



Shingles create edges when the image is not scaled down for edge detection.

Removing Lower Fourth of Edges

Many houses have features on the bottom of the house that the edge detector would invariably detect. These were often the bottom of doors, garages, bushes, ends of sidewalks, and some driveways. Since most people do not place lights on the ground or around bushes, although some do, the edges in the bottom 4th of the image were ignored when placing lights. Several values were tried, but the lower fourth of the image was eventually settled upon.

Lights on the ground when not ignoring the lower fourth of the image.

Lights no longer on the ground when ground when removing edges in lower fourth of the image

Creating a Strand of Christmas Lights

An image of a single light was used to create an entire strand. First, the light was cropped from an image of Christmas lights. Then, the background was replaced with an alpha layer and the image was desaturated. The alpha layer caused a lot of problems due to unfamiliarity with it. And sometimes images would become transparent due to not accounting for the alpha layer when averaging was done. After the light was desaturated, it was colored using a color mask that will be described in the following section. Then, the light was added to a strand. This process of coloring the light and adding it to the strand was continued until it was a large enough strand for its corresponding edge.

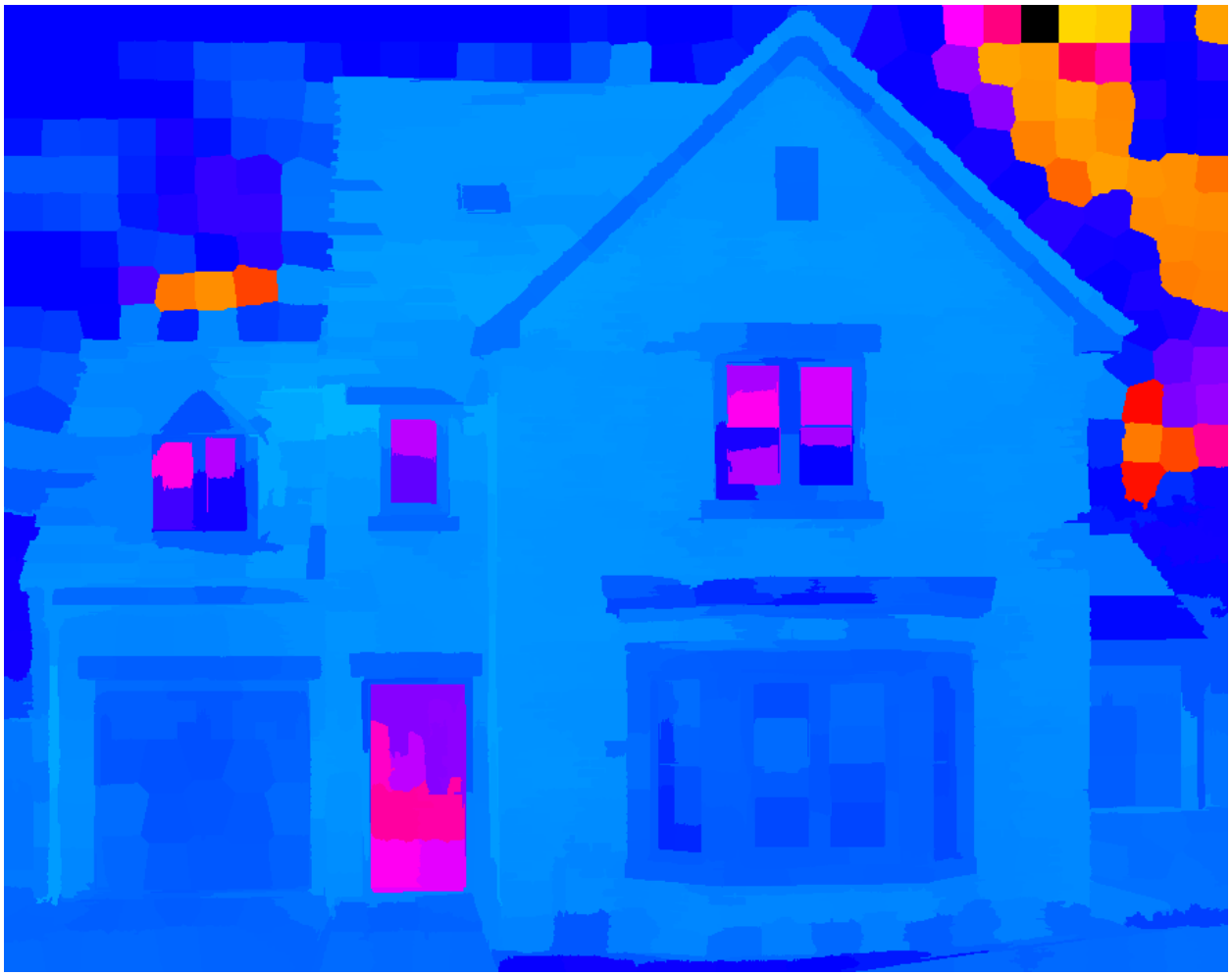
Single desaturated Christmas light

Light strand formed from several color masked lights

Generating the Light Color Mask

Utilizing a k-means algorithm on color and distance, with k equal to the square root of the number of pixels in the image, a set of ‘super pixels’ were generated. The average color of each of these super pixels was then calculated. Then a color mask for each color channel was generated by subtracting the lowest channel value from each channel and normalizing the resulting color vector. The resulting vector was then subtracted from a (1, 1, 1) vector. This created a vibrant color with a high contrast with the image at each point. When a light was being placed, it was colored by multiplying each color value in the pixel by the mask value corresponding to the upper left of the light’s placement.

Other color masks that were tried include the unmodified average color, the same modification described above without subtracting the generated vector from (1, 1, 1) and just using the resulting vector, and a scaled average color to maximize the brightness of the color while maintaining the color ratios. The pure average color was very difficult to see and not interesting to look at. The generated vector without the subtraction from (1, 1, 1) step was workable, but was discarded in favor of increased contrast. The scaled average color was discarded due to a lack of contrast, but it was better than a pure average color.



Color mask for lights

Placing A Light Strand On the Image

Light strands were then rotated to match the correct angle to go between two points. The rotated image was then placed so that the upper left of the image started in the upper left. This resulted in the strand being offset from the correct point when the strand was rotated due to the strand image size changing. An offset to account for the change to the rotated image's shape was added by adding the height of the image times the sin of the rotated angle when the image was rotated downward. A different offset was used when the image was rotated upward which was the change in height of the current image from the original image.

Since, lights have a quantized length, there was often some leftover space or gaps between the start of one strand and the end of its predecessor. To correct for this, the light's image was

resized to reach the correct end point. This results in some artifacts wherein some lights are larger than their neighbors. These artifacts could be removed by an algorithm that looks ahead and better spaces the points along the edges, but the artifacts were accepted as a part of the algorithm due to the artifact being discovered very recently.

Summary Of Work

Micah Fierro wrote the report and code for the following sections:

- Edge Detector
- Selecting the Best Edges

David Bair wrote the report and code for the following sections:

- Reducing Image Sizes
- Removing Lower Fourth of Edges
- Creating Light Strands
- Generating the Light Color Mask
- Placing a Light Strand On the Image