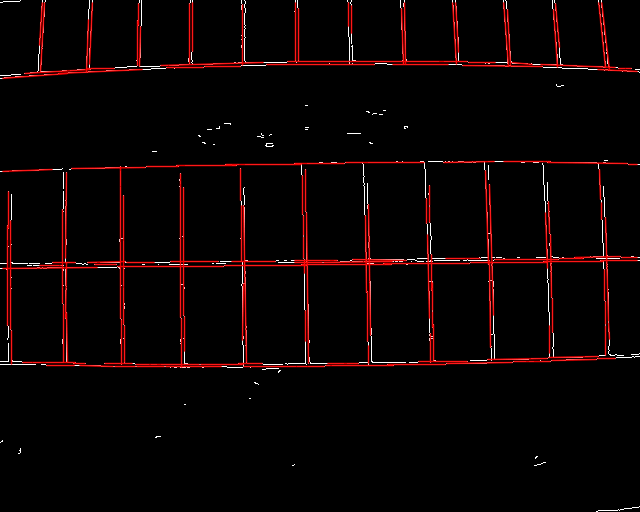
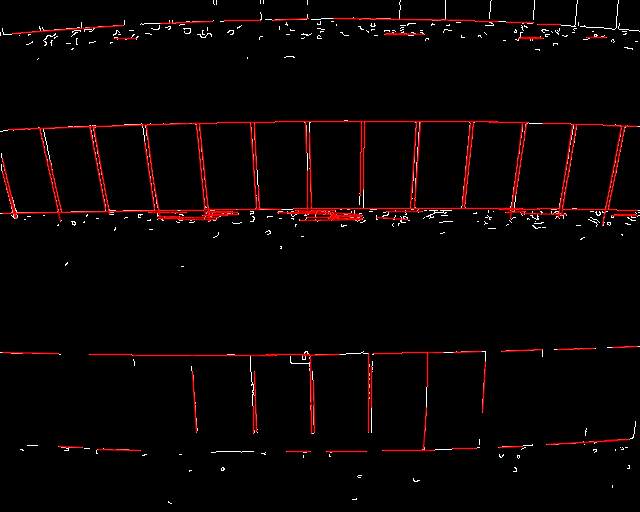
***FIRST ITERATION:***

***ALGORITHM:***

1. Apply canny edge detector on given grayscale test image.
2. Define a multidimensional vector to store detected line segments.
3. Apply probabilistic hough transform using opencv HoughP function.
4. Convert the image on which HoughP is performed to BGR, this is done so that opencv’s drawing functions work.
5. Draw the detected line segments on the converted BGR image.

***RESULTS:***



***SECOND ITERATION:***

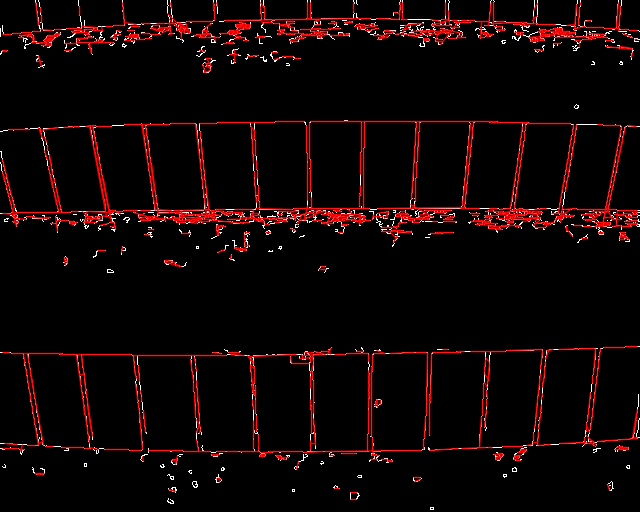
Lot of lines are still missing from the outputs, following changes were made:

Min and max values for canny edge detector was changed, these values are used for hysteresis thresholding, smoothing kernel size was reduced to 3 from 5 and L2 gradient was used in place of L1 norm to calculate pixel gradients . L2 grad gives more accurate results.

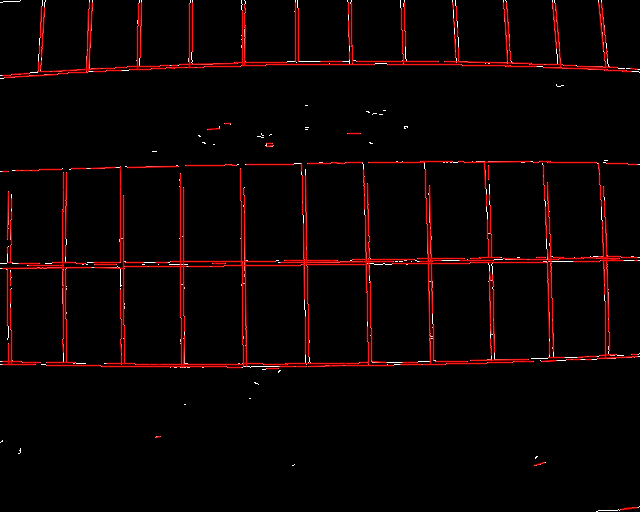
Houghp parameters like minimum line length and maximum allowed gap between points that are on the same line segment were also reduced.

Algorithm is still the same.

**RESULTS:**



**921 lines are detected**



**313 lines were detected**

**THIRD ITERATION:**

As evident, there are still many small line segments that are not being detected so the following changes were made:

Instead of directly applying edge detection and HoughP afterwards, couple of pre processing steps were added in order to remove the noise i.e. white edges being detected that are not part of solar panels.

So we have four methods that are done in different orders:

1. **Adaptive thresholding** **:** converts a noisy grayscale image to binary image using a kernel size of 3X3, thresholding method was THRESH\_BINARY\_INV which means any pixel with value above the value obtained by thresholding in neighborhood of 3X3 is zero.

**2) Edge detection :** As defined above, this detector keeps the edges across the pixel gradients are above the maxThreshold parameter and rejects edges that are below the minThreshold parameter, edges that are between are considered if they’re connected to any edge which satisfies the above condition.

**3)**  **Erode :** Included this method to clean out the borders of the image so that the lines are properly segmented from the background. Doesn’t make a lot of difference but a good practice to apply during the process.

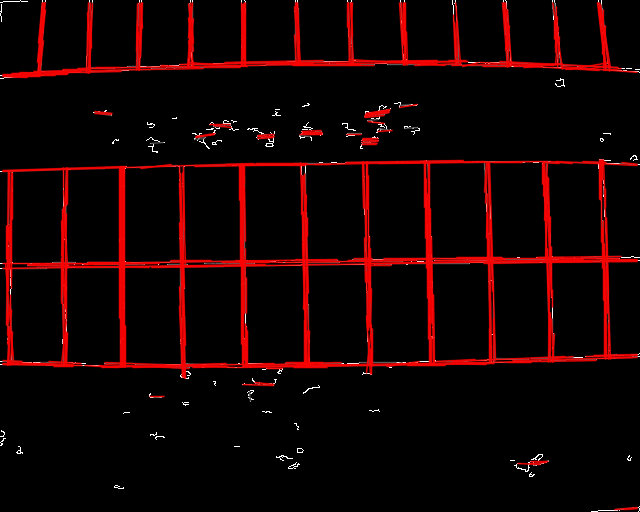
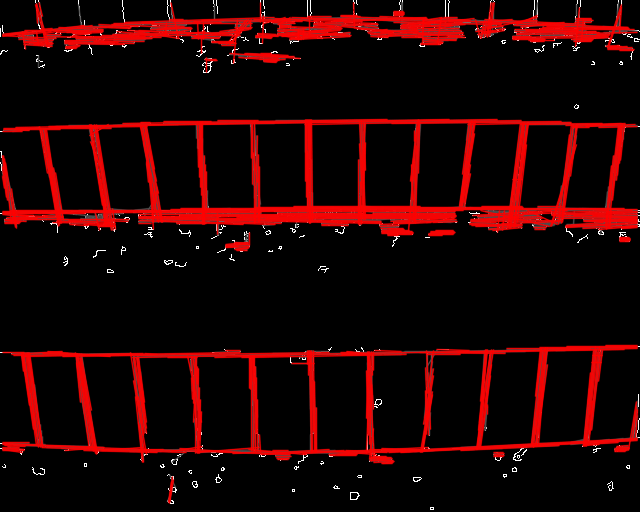
**4) HoughP :** Actual houghP algorithm, the parameters are almost the same as they’re in second iteration.

**ALGORITHM:** In final iteration, above four methods are applied in different order to obtain the desired result. One such order is mentioned below:

1. Adaptive thresholding was applied to the given grayscale test image with neighborhood size of 3X3.
2. Edge detection was done for both images, threshold values minThreshold and maxThreshold mentioned above for first test image was kept at 110 and 220 respectively while the same for second test image was between 100 and 255.
3. HoughP was applied for the first time with threshold votes = 20, minimum line length = 15, max allowed gap between points on a line = 7.
4. This step is optional, erosion is applied to BGR image to clean out the output.
5. HoughP is again applied to the image of previous step with same parameters ( could be different also) to detect every possible line segment.

During the above process, the image was converted back and forth between single channel and three channel BGR type to enable drawing functions and fulfill input requirements for HoughP which requires a single channel 8 bit image but drawing lines on an image requires it to be in BGR format.

**RESULTS:**



**USAGE:**

To perform the above algorithm on a real world image, opencv’s imread is used to read the image in Matrix form using this line,

**imgsrc=imread("/home/learn/Documents/cv/practice/projects/HoughP/0002.jpeg")**;

Compilation command line arguments:

FLAGS="$(pkg-config --cflags --libs opencv4)"

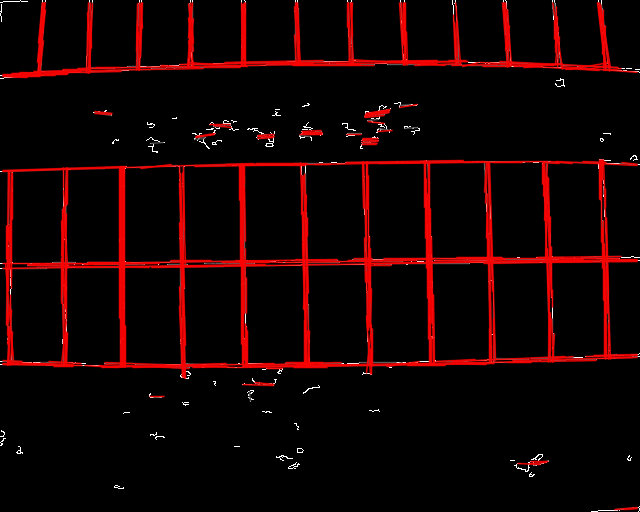
g++ houghpiterativesecond.cpp -o cv $FLAGS

./cv

FLAGS is an environment variable specifying the lib version and path variables. The compiled file is cv that is then executed.

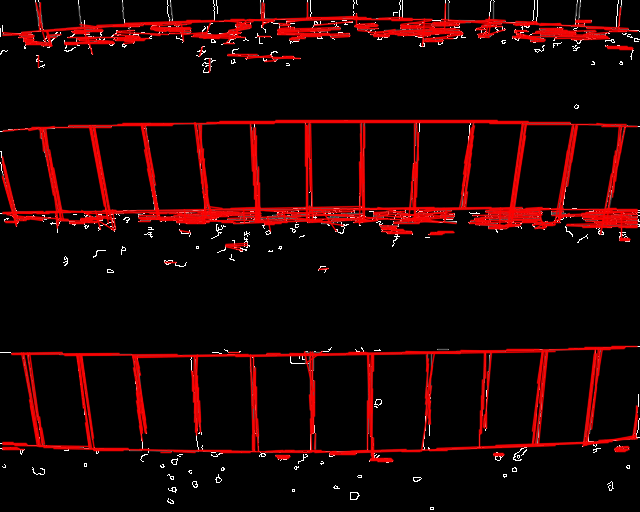
***Number of lines detected and grouped together = A***

***FOR FIRST TEST IMAGE :***



***A = 667***

**FOR SECOND TEST IMAGE :**



**A = 358**

FO