Detection and Removal of Scratches in Old Films

Objective

Old movie films often get deteriorated with artifacts such as blotches and scratches, through chemical change and mechanical contact with the film projector. These artifacts are to be removed, because they are annoying in sight and degrade the value of the historical record.

Implementation

Detection - Canny Method

This is an edge detection method that uses a multi- stage algorithm to detect a wide range of edges in an image. There are 5 steps to follow for this algorithm to work.

1. Noise reduction
2. Gradient calculation
3. Non-maximum suppression
4. Double threshold detection
5. Edge Tracking by Hysteresis

When these steps implemented following the order above, we are able to detect edges in an image

Noise Reduction

Edge detection involves calculation of gradients and the resulting edges detected are highly sensitive to image noise. One way to get rid of the noise on the image is to apply Gaussian blur to smoothen the image.
In this project the Gaussian filter used kernel size 5. The equation below shows the kernel equation.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

Figure1: Gaussian filter Kernel Equation

Gradient Calculation

The gradient calculation step detects the edge intensity and direction of the edges by calculating the gradient using the edge operators.

Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y). The Sober filters used to calculate the derivatives in the horizontal and vertical is as shown below.

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

Figure # : Sobel filters for both vertical and horizontal directions.

The magnitude and the slope of the gradient are calculated as follow:

$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x, y) = arctan\left(\frac{I_y}{I_x}\right)$$

However, the gradient intensity level is between 0 and 255, which is not uniform. The edges on the final result should have the same intensity with white being 255 and black being 0.

Non-Maximum Suppression

For the final image to have thin edges, non-maximum suppression has to be performed. In this step, the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions. The result is the same image but with thinner edges. But still there the result is not quite what we wanted since some pixels seem to be brighter than others. This problem is resolved in the next two final steps.

Double threshold Detection

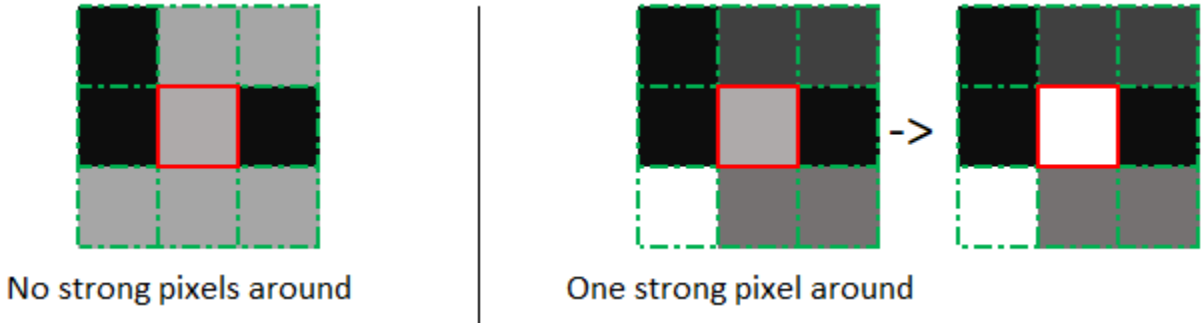
This step aims at identifying three kinds of pixels, strong, weak and non-relevant. Strong pixels are pixels that have an intensity so high that we are sure they contribute to the final edge. Weak pixels are pixels that have an intensity value that is not enough to be considered as strong ones, but yet not small enough to be considered as non-relevant for the edge detection and Other pixels are considered as non-relevant for the edge.

Now you can see what the double threshold holds for, a high threshold is used to identify the strong pixels (intensity higher than the high threshold). Low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold). All pixels having intensity between both thresholds are flagged as weak and the Hysteresis mechanism (next step) will help us identify the ones that could be considered as strong and the ones that are considered as non-relevant.

The result from after this step is an image with only 2 pixels intensity values; strong and weak.

Edge Tracking by Hysteresis

Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one, as described below



This is the final step of edge detection. After all the edges have been detected both scratches and image edges, Hough transform is then performed to only detect the scratches now.

The Hough Transform

The four concepts of the method are the edge image, the hough space and the mapping of the edges points onto the Hough Space, which is an alternate way to represent a line, and how lines are detected.

The Hough Transform Algorithm

1. Decide on the range of ρ and θ. Often, the range of θ is [0, 180] degrees and ρ is [$-d$, d] where d is the length of the edge image's diagonal. It is important to quantize the range of ρ and θ meaning there should be a finite number of possible values.
2. Create a 2D array called the accumulator representing the Hough Space with dimension (*num_rhos*, *num_thetas*) and initialize all its values to zero.
3. Perform edge detection on the original image. This can be done with any edge detection algorithm of your choice.
4. For every pixel on the edge image, check whether the pixel is an edge pixel. If it is an edge pixel, loop through all possible values of θ, calculate the corresponding ρ, find the θ and ρ index in the accumulator, and increment the accumulator base on those index pairs.

5. Loop through all the values in the accumulator. If the value is larger than a certain threshold, get the ρ and θ index, get the value of ρ and θ from the index pair which can then be converted back to the form of $y = ax + b$.

After the line of scratches have been detected, we then go to the final step, which is removal of the scratches.

Removal of Scratches - Adaptive Median Filtering

Adaptive Median Filter performs spatial processing to preserve detail and smooth non-impulsive noise. A prime benefit to this adaptive approach to median filtering is that repeated applications of this Adaptive Median Filter do not erode away edges or other small structure in the image.

Following the steps outlined in the lecture note, I was able to write an algorithm that removes the scratches from the images.

- z_{min}: minimum gray level in S_{xy}
- z_{max}: maximum gray level in S_{xy}
- z_{med}: median gray level of S_{xy}
- z_{xy}: gray level at coordinate (x,y)
- S_{max}: maximum allowed size of S_{xy}

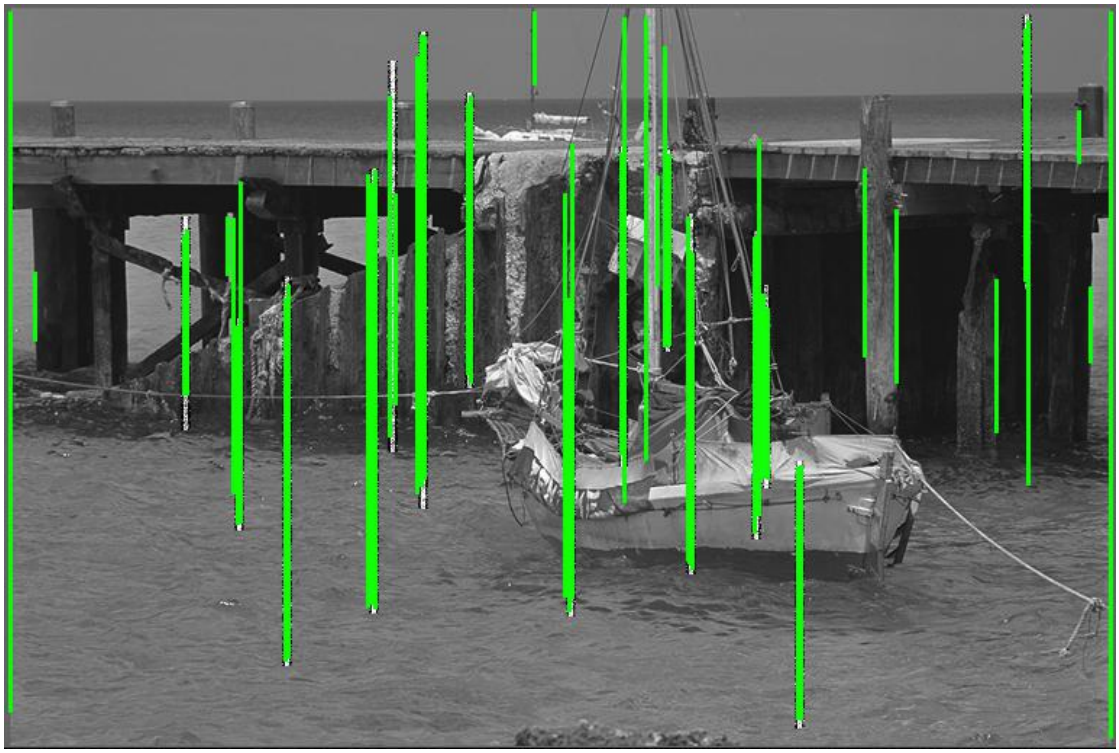
- A
  - $A1=z_{med}-z_{min}$
  - $A2=z_{med}-z_{max}$
  - If $A1>0$ and $A2<0$ go to B else increase the window size
  - If window size$<S_{max}$ repeat A
  - Else output z_{xy}
- B
  - $B1=z_{xy}-z_{min}$
  - $B2=z_{xy}-z_{max}$
  - If $B1>0$ and $B2<0$ output z_{xy}
  - Else output z_{med}

Steps followed to write the adaptive median filtering algorithm.

Experimental Result

For the scratch detection, the algorithm is doing really well since all the scratches in the image are detected. Below are the input image and the masted image with scratches detected.

Input image



Mask

Image with Scratches Detected

The output is as shown below.

Limitation of the Algorithm

This algorithm is not the best but it tries to do what it is designed for. First in the edge detection, the vertical edges of the image are also detected as a scratch. Secondly, the final restored image is not perfectly a uniform image. As you can see from above, there is still a little distortion in the image.

Conclusion

The implementation of this project was a step by step process. After researching from paper to paper, I was able to combine different techniques to do what the problem statement was. First, we start off from edge detection which include detecting the edges of both the image and scratches. After we do the Hough Transform step that differentiate between the edges of the image and the scratches. And then finally the removal of the scratches.

References

1. A method of Scratch Removal from Old Movie Film Using Variant Window by Hough Transform
   https://ieeexplore-ieee-org.libaccess.lib.mcmaster.ca/document/5341031

2. Canny Edge Detection
   https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123
3. Line Detection with Hough Transform
   https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549