

Task Progress Update Report

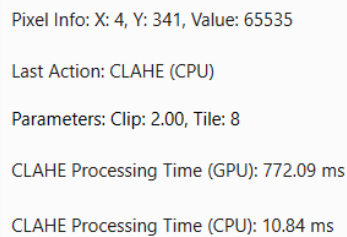
Name: LIM SHI KAI (Sky)

Update Date: 30-10-2024

1. Overview of Assigned Tasks and Recent Completion

Task 1: CLAHE in CPU Mode and Processing Time Display

- **Objective:** Add a function to apply CLAHE in CPU mode with a comparison display of processing time against the GPU implementation.
- **Current Status:** Completed
- **Details:** The CLAHE function is now in GPU and CPU modes. Processing times for each mode are displayed, enabling direct comparison. This allows the assessment of computational benefits when using GPU acceleration over CPU for image enhancement.
- **Image:**



Pixel Info: X: 4, Y: 341, Value: 65535
Last Action: CLAHE (CPU)
Parameters: Clip: 2.00, Tile: 8
CLAHE Processing Time (GPU): 772.09 ms
CLAHE Processing Time (CPU): 10.84 ms

Figure 1.1 Info label for CLAHE Processing Time

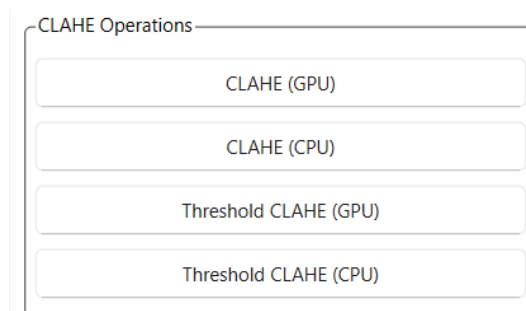


Figure 1.2 Control Panel for CLAHE Operation

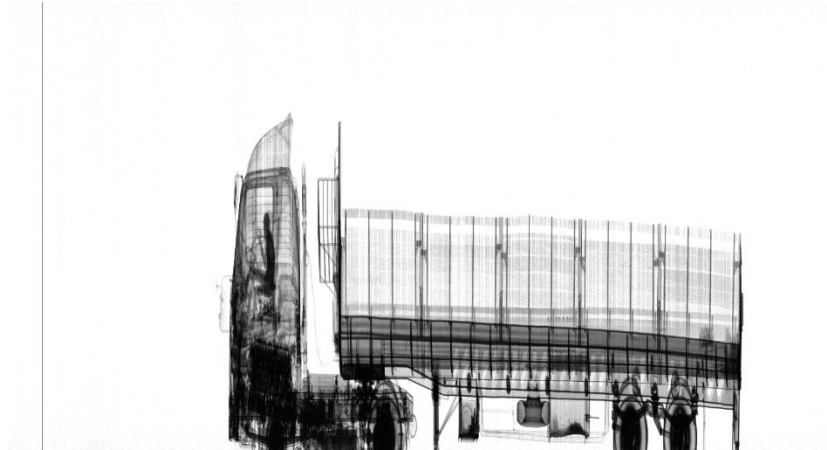


Figure 1.3 Output for CLAHE CPU

Task 2: Threshold-Based CLAHE with GPU/CPU

- **Objective:** Implement selective CLAHE using GPU and CPU modes with threshold-based vector segmentation, incorporating advanced enhancements for image quality.
- **Current Status:** Completed
- **Details:** This implementation utilizes vector-based thresholding for dark regions, enhanced by image sharpening, local contrast adjustments, and multipass Gaussian blurring. Gradual intensity-based adaptation helps produce a smooth, visually consistent output without artifacts, applying CLAHE only where needed.
- **Image:**

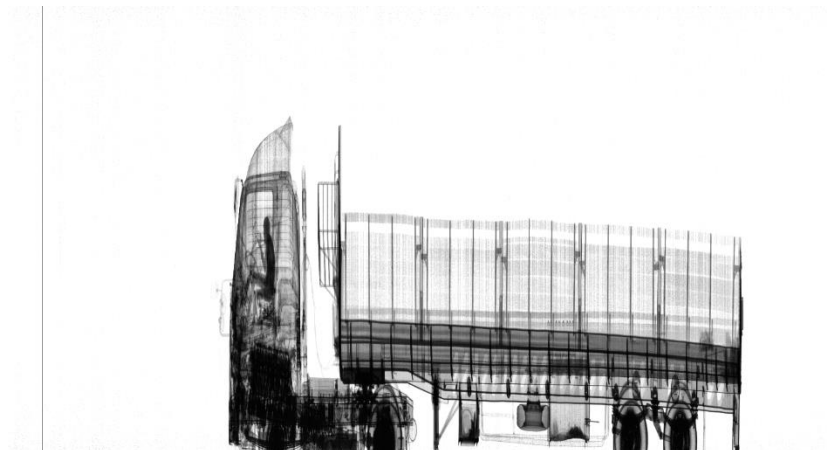


Figure 1.4 Output for Threshold CLAHE (GPU) in 5000 threshold, 2 clipLimit, 80 tiles

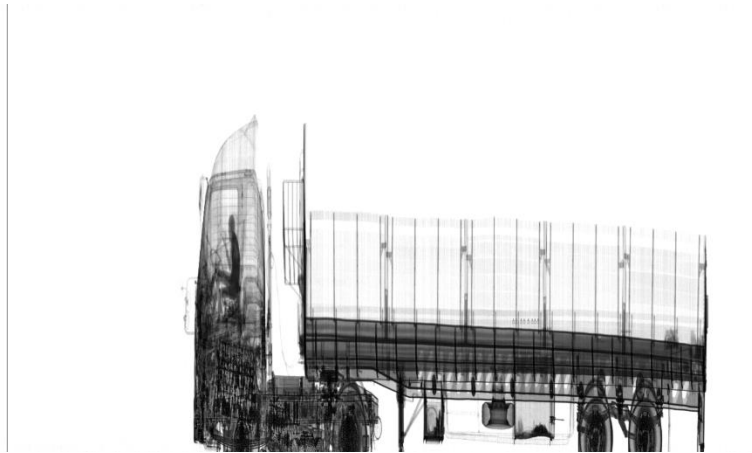


Figure 1.5 Output for Threshold CLAHE (CPU) in 5000 threshold, 2 clipLimit, 80 tiles

CLAHE Processing Time (GPU): 997.26 ms

CLAHE Processing Time (CPU): 98.77 ms

Figure 1.6 Processing Time comparison for Threshold CLAHE

Task 3: Thread Processing Display in Debug Mode

- **Objective:** Display detailed thread processing in debug mode for adjustment functions, including gamma, contrast, and sharpening, and for region-specific operations.
- **Current Status:** Completed
- **Details:** Thread processing for various adjustments (e.g., gamma, contrast) now includes debug outputs to trace and optimize threaded operations. This feature assists with debugging issues by pinpointing artifacts from specific thread regions, especially when functions are applied selectively to image areas.
- **Debug Output Image:**

```
Starting gamma adjustment with 8 threads
Thread 0 processing gamma correction for rows 0 to 104
Thread 1 processing gamma correction for rows 104 to 208
Thread 2 processing gamma correction for rows 208 to 312
Thread 3 processing gamma correction for rows 312 to 416
Thread 4 processing gamma correction for rows 416 to 520
Thread 5 processing gamma correction for rows 520 to 624
Thread 6 processing gamma correction for rows 624 to 728
Thread 7 processing gamma correction for rows 728 to 832
Thread 1 completed gamma correction
Thread 4 completed gamma correction
Thread 2 completed gamma correction
Thread 3 completed gamma correction
Thread 6 completed gamma correction
Thread 5 completed gamma correction
Thread 7 completed gamma correction
Thread 0 completed gamma correction
All threads completed gamma adjustment
```

Figure 1.7 Thread Processing Step Debug

Task 4: Black Line Detection and Removal

- **Objective:** Detect and remove black lines while preserving those associated with objects, using multi-threaded detection and adaptive replacement.
- **Current Status:** Completed
- **Details:**
 - **Detection:** The `detectBlackLines` function leverages multi-threading to identify black lines vertically and horizontally across the image. Key parameters such as `BLACK_THRESHOLD`, `NOISE_TOLERANCE`, and `WHITE_THRESHOLD` control line detection accuracy. Threads process designated columns and rows, identifying black pixels within a tolerance limit.
 - **Object Proximity Check:** Using a lambda function, `isInObject` determines if lines are near objects. Each line is assessed based on surrounding pixels (up to a defined range) to verify if they form part of an object.
 - **Removal:** The `removeBlackLines` function selectively removes isolated black lines by replacing black pixels with median brightness values from nearby pixels. Vertical or horizontal checks within a `SEARCH_RADIUS` provide valid replacement values, maintaining image integrity without affecting object boundaries.
 - **Visualisation:** The `visualizeBlackLines` function presents detected lines with detailed coordinates and dimensions, aiding verification of the line detection accuracy before removal.
- **Process Image:**

For thin line:

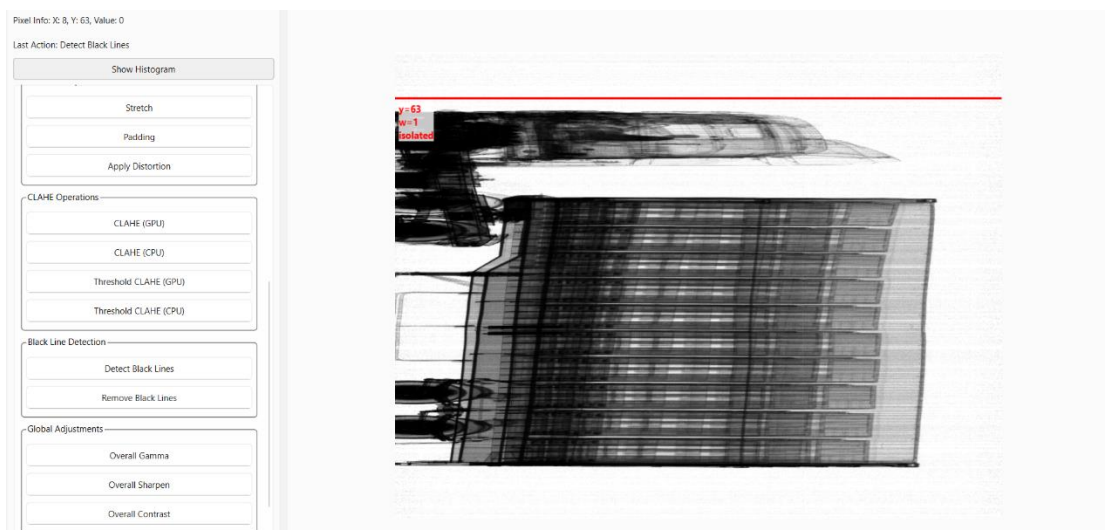


Figure 1.8 Red Line will be shown if the line is not in the vehicle region

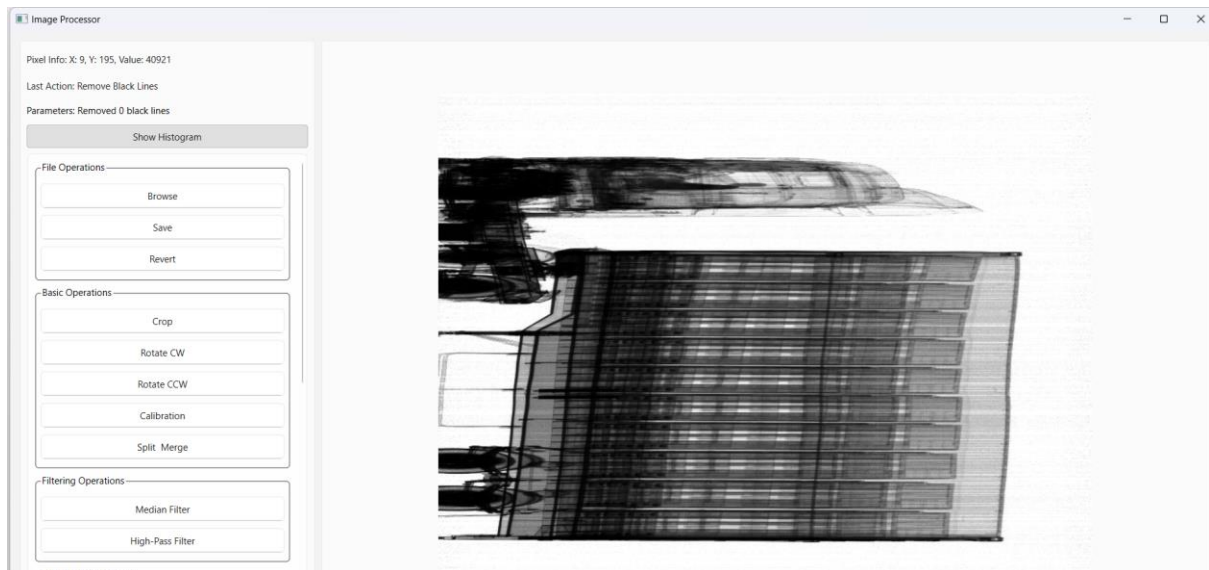


Figure 1.9 Line will be removed and the red line will be auto-hide

```
Image loaded successfully from: "C:/internship/2_Raw File/remove line lost/0101202407020970/010120240702142147_raw.txt"
Starting vertical line detection with 8 threads
Thread 0 processing columns 0 to 104
Thread 0 completed vertical line detection
Thread 1 processing columns 104 to 208
Thread 1 completed vertical line detection
Thread 2 processing columns 208 to 312
Thread 2 completed vertical line detection
Thread 5 processing columns 520 to 624
Thread 5 completed vertical line detection
Thread 6 processing columns 624 to 728
Thread 6 completed vertical line detection
Thread 7 processing columns 728 to 832
Thread 3 processing columns 312 to 416
Thread 7 completed vertical line detection
Thread 3 completed vertical line detection
Thread 4 processing columns 416 to 520
Thread 4 completed vertical line detection
Starting horizontal line detection with 8 threads
Thread 2 processing rows 160 to 240
Thread 2 completed horizontal line detection
Thread 0 processing rows 0 to 80
Thread 0 completed horizontal line detection
Thread 3 processing rows 240 to 320
Thread 3 completed horizontal line detection
Thread 1 processing rows 80 to 160
Thread 1 completed horizontal line detection
Thread 4 processing rows 320 to 400
Thread 4 completed horizontal line detection
Thread 5 processing rows 400 to 480
Thread 5 completed horizontal line detection
Thread 7 processing rows 560 to 640
Thread 7 completed horizontal line detection
Thread 6 processing rows 480 to 560
Thread 6 completed horizontal line detection
Thread 0 checking horizontal line at y=63
Thread 0 - Analyzing horizontal line at y= 63 with width= 1
Black line detection completed. Found 1 lines.
```

Figure 1.10 Thread Processing

For thick line:



Figure 1.11.1 Thick Line detected with 40 width

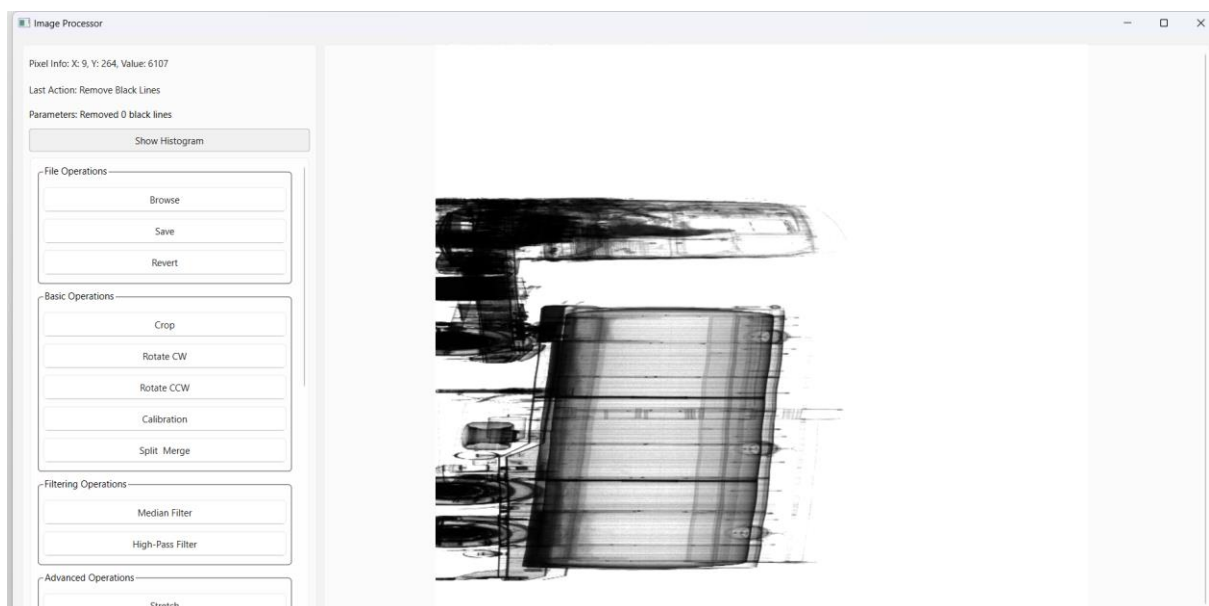


Figure 1.11.2 Thick Line removed

```
Image loaded successfully from: "C:/internship/2_Raw File/remove line lost/0101202407020991/010120240702143911_raw.txt"
Starting vertical line detection with 8 threads
Thread 0 processing columns 0 to 104
Thread 6 processing columns 624 to 728
Thread 0 completed vertical line detection
Thread 3 processing columns 312 to 416
Thread 6 completed vertical line detection
Thread 4 processing columns 416 to 520
Thread 3 completed vertical line detection
Thread 5 processing columns 520 to 624
Thread 4 completed vertical line detection
Thread 1 processing columns 104 to 208
Thread 5 completed vertical line detection
Thread 2 processing columns 208 to 312
Thread 1 completed vertical line detection
Thread 7 processing columns 728 to 832
Thread 2 completed vertical line detection
Thread 7 completed vertical line detection
Starting horizontal line detection with 8 threads
Thread 0 processing rows 0 to 104
Thread 5 processing rows 520 to 624
Thread 0 completed horizontal line detection
Thread 5 completed horizontal line detection
Thread 1 processing rows 104 to 208
Thread 1 completed horizontal line detection
Thread 4 processing rows 416 to 520
Thread 4 completed horizontal line detection
Thread 2 processing rows 208 to 312
Thread 2 completed horizontal line detection
Thread 7 processing rows 728 to 832
Thread 7 completed horizontal line detection
Thread 6 processing rows 624 to 728
Thread 6 completed horizontal line detection
Thread 3 processing rows 312 to 416
Thread 3 completed horizontal line detection
Thread 0 checking horizontal line at y=24

Thread 0 - Analyzing horizontal line at y= 24  with width= 40
Black line detection completed. Found 1 lines.
```

Figure 1.11.3 Thread Processing

For Line around Vehicle:



Figure 1.11.4 The line detected in Object will show in Blue



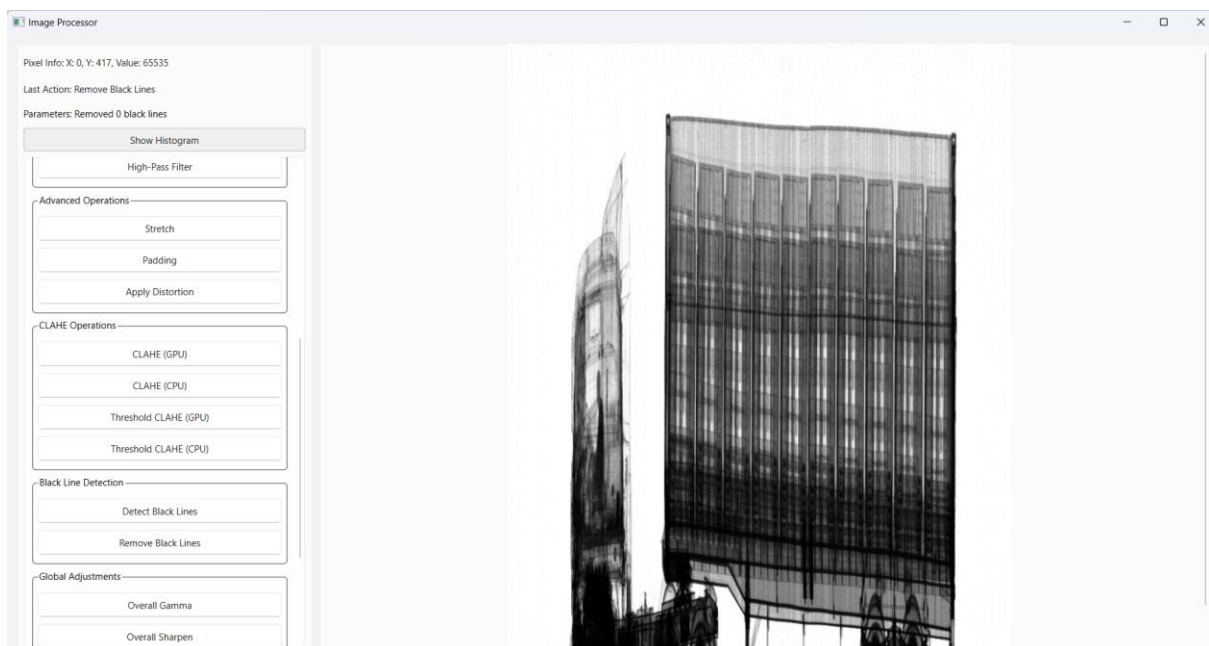
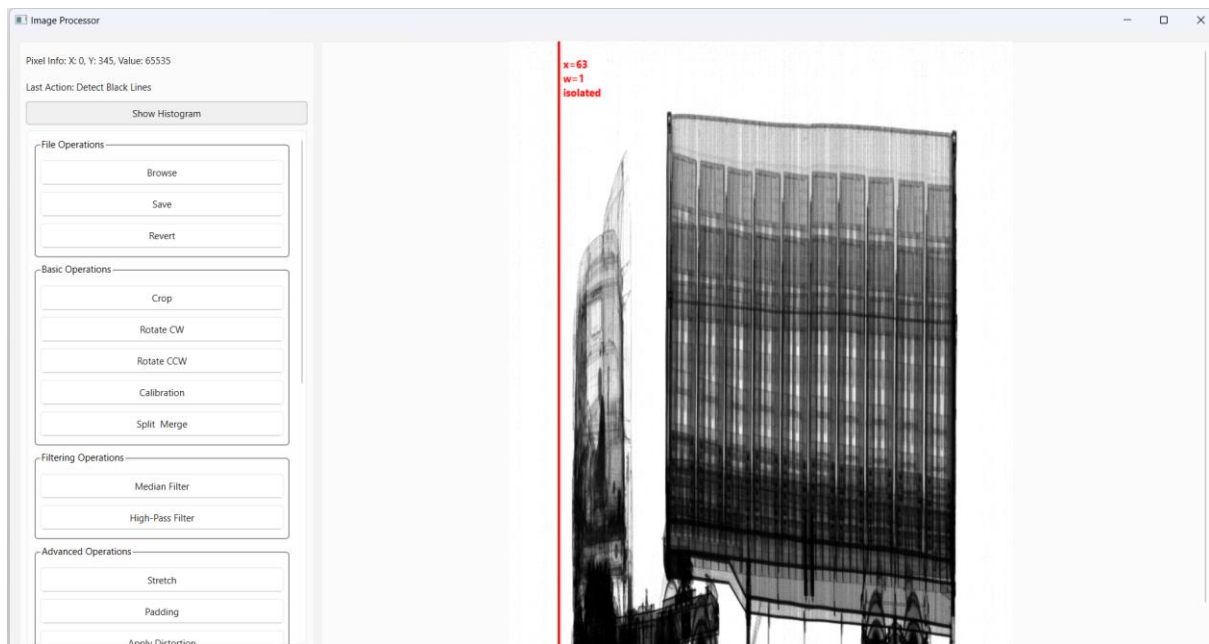
Figure 1.11.6 Line remained even the remove function called

```
Image loaded successfully from: "C:/internship/2_Raw File/remove line lost/0102202401310632/010220240131152514_raw.txt"
Starting vertical line detection with 8 threads
Thread 0 processing columns 0 to 104
Thread 0 completed vertical line detection
Thread 1 processing columns 104 to 208
Thread 1 completed vertical line detection
Thread 2 processing columns 208 to 312
Thread 2 completed vertical line detection
Thread 5 processing columns 520 to 624
Thread 5 completed vertical line detection
Thread 3 processing columns 312 to 416
Thread 3 completed vertical line detection
Thread 7 processing columns 728 to 832
Thread 7 completed vertical line detection
Thread 4 processing columns 416 to 520
Thread 4 completed vertical line detection
Thread 6 processing columns 624 to 728
Thread 6 completed vertical line detection
Starting horizontal line detection with 8 threads
Thread 0 processing rows 0 to 72
Thread 0 completed horizontal line detection
Thread 1 processing rows 72 to 144
Thread 1 completed horizontal line detection
Thread 3 processing rows 216 to 288
Thread 3 completed horizontal line detection
Thread 2 processing rows 144 to 216
Thread 2 completed horizontal line detection
Thread 4 processing rows 288 to 360
Thread 4 completed horizontal line detection
Thread 5 processing rows 360 to 432
Thread 5 completed horizontal line detection
Thread 6 processing rows 432 to 504
Thread 6 completed horizontal line detection
Thread 7 processing rows 504 to 576
Thread 7 completed horizontal line detection
Thread 0 checking horizontal line at y=63
Black line detection completed. Found 1 lines.

Thread 0 - Analyzing horizontal line at y= 63 with width= 1
Thread 0 - Found non-white pixel above line at x= 0
```

Figure 1.15 Thread Processing

For thin lines in vertical:



```
Starting horizontal line detection with 8 threads
Thread 0 processing rows 0 to 104
Thread 0 completed horizontal line detection
Thread 7 processing rows 728 to 832
Thread 7 completed horizontal line detection
Thread 2 processing rows 208 to 312
Thread 2 completed horizontal line detection
Thread 4 processing rows 416 to 520
Thread 4 completed horizontal line detection
Thread 5 processing rows 520 to 624
Thread 5 completed horizontal line detection
Thread 6 processing rows 624 to 728
Thread 6 completed horizontal line detection
Thread 1 processing rows 104 to 208
Thread 1 completed horizontal line detection
Thread 3 processing rows 312 to 416
Thread 3 completed horizontal line detection
Black line detection completed. Found 1 lines.
```

2. Roadblocks/Challenges

- **CLAHE with Threshold Blending:** While improvements have been made, tried to make the CLAHE with Threshold Blending able to apply after the normal CLAHE applied.
 - **Black Line Detection:** The Parameters or Information Label there needs to be updated with clear information, such as coordinates or lines.
-

3. Conclusion

- **Key Progress:**
 - CLAHE implementation for CPU and GPU modes, complete with comparing processing time.
 - Advanced CLAHE threshold-based enhancements, including vector-based segmentation and smoothing adjustments.
 - Multi-threaded thread processing displays for debugging adjustments.
 - Multi-threaded black line detection and selective removal, distinguishing isolated lines from those near objects.