VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANASANGAMA, BELAGAVI - 590018

Internship Report (21CSE182)

on

AI and ML Based Fabric Thread Count Measurement System Using Raspberry Pi and High-Resolution Camera

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by

1BG21CS058 Rashmi M Patil

Internship Guide
Mr. Sridhar Venkannachar

Internal Guide
Prof. Ashwini R Malipatil
Assistant Professor, Dept. of CSE
BNMIT, Bengaluru

Vidyayâmruthamashnuthe

B.N.M. Institute of Technology

An Autonomous Institution under VTU

Approved by AICTE, Accredited as grade A Institution by NAAC. All eligible branches – CSE, ECE, EEE, ISE & Mech. Engg. are Accredited by NBA for academic years 2018-19 to 2024-25 & valid upto 30.06.2025

URL: www.bnmit.org

Department of Computer Science and Engineering

2024 - 2025

B.N.M. Institute of Technology

An Autonomous Institution under VTU

Approved by AICTE, Accredited as grade A Institution by NAAC. All eligible branches – CSE, ECE, EEE, ISE & Mech. Engg. are Accredited by NBA for academic years 2018-19 to 2024-25 & valid upto 30.06.2025

URL: www.bnmit.org

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vidyayāmruthamashnuthe

CERTIFICATE

Certified that the internship work entitled **AI and ML Based Fabric Thread Count Measurement System Using Raspberry Pi and High-Resolution Camera** carried out by **Ms. Rashmi M Patil** (**1BG21CS058**), is a bonafide student of VIII Semester, BNM Institute of Technology in partial fulfillment for the award of Bachelor of Engineering in COMPUTER SCIENCE AND ENGINEERING of Visvesvaraya Technological University, Belagavi during the year 2024-25. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Internship Report has been approved as it satisfies the academic requirements in respect of internship work prescribed for the said Degree.

Prof. Ashwini R Malipatil	**Dr. Chayadevi M L**	**Dr. S Y Kulkarni**
Assistant Professor	**Professor and HOD**	**Additional Director and**
Department of CSE	**BNMIT, Bengaluru**	**Principal**
BNMIT, Bengaluru		**BNMIT, Bengaluru**

Examiner 1:

Examiner 2:

ACKNOWLEDGEMENT

I would like to place on record my sincere thanks and gratitude to the concerned people, whose suggestions and words of encouragement has been valuable.

I would like to thank **Shri. Narayan Rao R. Maanay**, Secretary, BNMIT, Bengaluru for providing excellent academic environment in the college.

I would like to sincerely thank **Prof. T. J. Rama Murthy**, Director, BNMIT, Bengaluru and **Dr. S. Y. Kulkarni**, Additional Director and Principal, BNMIT, Bengaluru for having extended his support and encouragement during the course of the work.

I would like to express my gratitude to **Prof. Eishwar N. Maanay**, Dean, BNMIT, Bengaluru for his relentless support, guidance, and assistance.

I would like to thank **Dr. Krishnamurthy G. N.**, Deputy Director, BNMIT, Bengaluru for his constant encouragement.

I would like to thank **Dr. Chayadevi M. L.**, Professor and Head of the Department of Computer Science and Engineering, who has shared her opinions and thoughts and extended her undivided support which helped me in giving my presentation successfully.

I would like to thank **Mr. Sridhar Venkannachar,** Internship Guide, for his guidance, feedback, and motivation throughout the internship, making this project successful.

I would like to thank **Prof. Ashwini R Malipatil**, Assistant Professor, for being the guiding force towards the successful completion of the project.

I would like to thank **Dr. Kavitha Jayaram**, Associate Professor and Internship Coordinator, Department of Computer Science and Engineering, for supporting and encouraging me during internship work.

Finally, I take this opportunity to extend my earnest gratitude and respect to my parents, family members, teaching & non-teaching staffs of the Department and all my friends, for always giving me valuable advice and support in all possible ways.

Rashmi M Patil
1BG21CS058

ABSTRACT

Fabric quality assessment plays a pivotal role in the textile industry, ensuring the production of consistent, durable, and defect-free materials. Traditional methods of fabric inspection rely heavily on human expertise, which, despite its advantages, is time-consuming, labor-intensive, and susceptible to errors. To address these challenges, this study introduces an automated thread detection system implemented on a Raspberry Pi, leveraging computer vision and image processing techniques to analyze the fabric's structural composition with high precision.

The system is designed to capture high-resolution images of fabric samples using the PiCamera module, ensuring detailed visualization of the weave patterns. Once an image is acquired, it undergoes preprocessing and enhancement using techniques such as grayscale conversion, contrast adjustment, and noise reduction to prepare the data for analysis. The Otsu's thresholding method is then applied to convert the image into a binary format, effectively distinguishing the fabric's thread structures from the background. This binarization process is crucial for accurately identifying the dark and light regions corresponding to the woven threads.

To detect and count the individual horizontal and vertical threads, the system employs edge detection algorithms and frequency analysis techniques. The edge detection process highlights the boundaries of threads, allowing for a structured analysis of their placement and orientation. Fourier transform-based frequency analysis is used to determine the periodicity of the weave pattern, ensuring precise thread count estimation. By segmenting a 1x1 inch cropped section of fabric, the system extracts essential parameters such as thread density, alignment, and irregularities, which serve as indicators of the fabric's overall quality.

The research contributes significantly to the automation of textile inspection processes, reducing human error and enhancing the efficiency of fabric quality assessment. Future advancements could include deep learning-based pattern recognition to classify different fabric types and defects with higher precision, further optimizing textile manufacturing workflows.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

ABOUT THE ORGANIZATION

CHAPTER - 1

ABOUT THE ORGANIZATION

1.1 Overview of the organization

AutoTEC Systems Private Limited, established in April 2000, is a Bengaluru-based technology company specializing in providing products and customized solutions for defense applications**.**AutoTEC specialises in Design & Development, prototyping, qualification and manufacturing to Military standards, with focus on On-board Systems, Automated Testing and Ground Systems for Avionics, UAV Sub-systems and other defence applications.

1.1.1 AutoTEC Systems Leadership Team

Satish Kumar (Managing Director) – Electronics engineer with 34+ years of experience in avionics, defense systems, and business development. Co-founded AutoTEC in 2000, leading production, planning, and quality implementation.

C S Srikanth (Director – Technology) – Electronics engineer with 28+ years of experience in analog and digital hardware design. Leads mission-critical avionics system development for military aircraft like LCA, SU-30, Jaguar, and MIG-27.

B S Shashi Kumar (Vice President – Finance) – IT graduate with 24+ years of experience in project planning, corporate banking, and institutional finance. Heads finance and corporate management at AutoTEC.

A Srihari (Vice President – HR & Operations) – MCA graduate with expertise in software development for DRDO projects. Leads AutoTEC's software development team and oversees HR and operations.

1.1.2 Certifications and Approvals

- **AS9100D and ISO 9001:2015 Certified:** These certifications underscore AutoTEC's commitment to maintaining high-quality standards in design, development, and manufacturing processes for defense and aerospace applications.

- **CEMILAC Approved Design House:** This approval signifies the company's recognized capability in defense electronics systems design.

1.1.3 Product Portfolio

AutoTEC has developed a range of on-board avionics systems and products, including:

- Avionics Computers: Advanced computing systems designed for aircraft applications.
- Data Transfer and Recording Systems: Solutions for efficient data management and retrieval.
- On-Board Electronics: Customized electronics tailored for specific defense applications.
- UAV Electronics: Specialized systems for unmanned aerial vehicles, such as data communication modules, telemetry systems, digital engine control units, and pilot console systems.

1.1.4 Vision and Mission

Our vision is to be amongst premier Defence and Aerospace systems and solution providers of the country through reliable, quality and timely services ensuring customer satisfaction.

Our mission is to achieve profitable growth by providing indegenous, reliable and quality solutions through continuous product development efforts coupled with enhanced quality systems and processes to achieve timely and superior services ensuring customer satisfaction and build value for all our stakeholders.

CHAPTER 2

ABOUT THE DEPARTMENT

CHAPTER - 2

ABOUT THE DEPARTMENT

2.1 Overview of Embedded Software and IoT Services in Automation

Embedded software and IoT (Internet of Things) services are transforming industries by enabling real-time data processing, automation, and intelligent decision-making. Embedded systems consist of dedicated software running on microcontrollers or specialized hardware, allowing devices to function efficiently with minimal resources. IoT services further enhance this by enabling connectivity between devices, making it possible to monitor and control systems remotely. These technologies are widely used in industrial automation, smart manufacturing, and quality control, where precision and reliability are essential.

In fabric inspection and textile quality control, automated systems play a crucial role in detecting defects, analyzing thread density, and ensuring product consistency. Traditional methods of fabric inspection are labor-intensive and prone to errors, but modern embedded solution**s** help overcome these limitations by providing faster and more accurate measurement**s**. By integrating smart cameras and real-time processing, these systems can identify irregularities in weave patterns and assist manufacturers in maintaining high-quality standards. Such solutions can be deployed in textile factories, where they operate as standalone inspection units or as part of a larger automated production line.

By leveraging real-time image analysis and pattern detection, embedded systems can bring significant improvements to industries that rely on precision measurements. While this project does not involve cloud integration or IoT connectivity, it still falls under embedded software applications because it runs on hardware (such as a Raspberry Pi) with on-device processing. Future enhancements could include wireless data transmission, remote monitoring, or integration with industrial textile machines for fully automated quality control. As embedded systems advance, they continue to shape modern manufacturing by offering fast, accurate, and automated solutions tailored to specific industrial needs.

CHAPTER 3

TASK PERFORMED

CHAPTER 3

TASKS PERFORMED

3.1 Overview of the project

3.1.1 Understanding Thread Count

Thread count refers to the total number of threads woven together in one square inch of fabric, including both horizontal (weft) and vertical (warp) threads. It serves as a key indicator of fabric quality, influencing properties such as softness, durability, and breathability. A higher thread count generally suggests a finer and smoother texture, often seen in high-end cotton and satin fabrics used in bedding, clothing, and upholstery. However, an excessively high thread count does not always translate to better quality, as factors like fiber type, weaving method, and finishing treatments also play a crucial role in fabric performance.

Thread count is determined by counting the number of individual warp and weft threads in a given square inch of fabric. For example, a fabric with 150 warp threads and 150 weft threads per square inch has a thread count of 300. The density of the weave directly impacts how the fabric feels and performs over time. In loosely woven fabrics, lower thread counts result in more breathable and lightweight materials, while tightly woven fabrics with higher thread counts offer a softer and smoother finish.

High thread counts are often marketed as a sign of luxurious, comfortable bedding. However, optimal comfort is achieved through a balance of fiber quality, weave structure, and breathability. Soft, durable, and wrinkle-resistant fabrics with medium to high thread counts are preferred for premium-quality shirts, suits, and dresses. Understanding thread count beyond numerical values is crucial for both consumers and manufacturers. While a high thread count can indicate superior quality, true fabric performance depends on a combination of fiber composition, weave structure, and finishing techniqu

3.1.2 Why is Thread Count Important?

1. **Fabric Quality Assessment:** Thread count helps determine the density, strength, and durability of a fabric, directly impacting its overall quality. Higher thread counts generally indicate a tighter weave, though breathability might be reduced in very high thread count materials.
2. **Manufacturing Standards:** Textile industries rely on thread count measurements to maintain consistency in fabric production, ensuring uniform quality across batches. Standardized thread count evaluations allow manufacturers to meet industry guidelines and improve product quality control.
3. **Product Categorization:** Different thread counts classify fabrics into various grades, helping distinguish between premium and standard textile materials. For instance, fabrics with thread counts below 200 are often considered lower-grade, whereas high-end sheets and garments typically feature counts of 300 or more.
4. **Consumer Awareness:** Consumers use thread count information to make informed purchasing decisions, especially when selecting fabrics for home textiles, clothing, and luxury items. Marketing strategies often emphasize high thread counts as an indicator of superior fabric, though other factors such as fiber quality and weave type must also be considered.
5. **Economic Implications:** High thread count fabrics often cost more due to the intricate weaving process and finer threads used, making it a key marketing and pricing factor in the textile industry. Luxury brands frequently promote high thread count products to justify premium pricing, though it is essential to balance quality with affordability to cater to diverse market segments.

3.2 Objective

1. Develop an automated system to accurately count horizontal (weft) and vertical (warp) threads in a 1-inch by 1-inch fabric sample, focusing on cotton and satin materials.
2. Replace traditional manual counting methods with an automated approach to ensure precision and consistency in textile quality assessment.
3. Utilize contrast enhancement, filtering, and pattern detection algorithms to accurately differentiate fabric threads from background noise.

4. Account for differences in fabric texture, weave complexity, and lighting conditions to ensure accurate thread count analysis.
5. Improve textile inspection accuracy, allowing manufacturers to maintain consistent fabric quality and detect defects efficiently.
6. Enable seamless integration into textile production lines, supporting both large-scale and small-scale fabric manufacturers.
7. Reduce labor-intensive manual inspections and speed up the thread-counting process, making it more cost-effective and efficient.
8. Improve adaptability by incorporating machine learning techniques to enhance analysis accuracy across different fabric types and thread densities.
9. Design an accessible and easy-to-use system suitable for textile manufacturers, researchers, and quality control professionals.

3.3 System Requirements

Development Environment

- **Hardware:** Raspberry Pi 3 B+ Model
- **Camera:** OV7670 Camera Module (640x480 VGA CMOS Sensor)
- **Operating System**: Raspberry Pi OS (Raspbian)
- **Programming Language:** Python 3

Required Software and Libraries

- sudo apt update && sudo apt upgrade –y
- pip install opencv-python numpy matplotlib scipy picamera

Library Purpose

- opencv-python Image processing and analysis
- numpy Numerical computations
- matplotlib Visualization of results
- scipy.signal Peak detection for thread counting
- picamera Capturing images with the Raspberry Pi camera module

Hardware Configuration

1. **Raspberry Pi 3 B:** This model was selected due to its processing power and connectivity options. The Pi 3 B runs a Linux-based operating system, making it compatible with various image processing libraries and software tools.
2. **PiCamera Module:** The camera was attached to the Raspberry Pi using the Camera Serial Interface (CSI). The camera was configured for optimal image capture, with attention to lighting conditions and focus settings to ensure clear images.
3. **Power Supply**: A stable 5V 2.5A power supply was used to power the Raspberry Pi and connected peripherals, ensuring uninterrupted operation during image capture.

3.4 Hardware Setup

Part 1: Install Raspberry Pi OS

1. Download Raspberry Pi Imager: Visit the Raspberry Pi website and download the Raspberry Pi Imager for your operating system (Windows, macOS, or Linux).
2. Prepare the microSD Card: Insert a microSD card (minimum 8 GB recommended) into your laptop using an adapter if needed.
3. Install Raspberry Pi OS:
   - Open Raspberry Pi Imager.
   - Click "Choose OS" and select "Raspberry Pi OS (32-bit)" or another version based on your needs.
   - Click "Choose Storage" and select your microSD card.
   - Click "Write" to flash the OS onto the microSD card. Wait for the process to complete.
4. Enable SSH:
   - After flashing, remove and reinsert the microSD card into your laptop to access its boot partition.
   - Create a blank file named ssh (no extension) in the boot directory to enable SSH.
5. Boot the Raspberry Pi:
   - Insert the microSD card into the Raspberry Pi.
   - Power on the Raspberry Pi. It will connect to your Wi-Fi network.

Part 2: Connect to the Raspberry Pi via PuTTY

1. Find the Raspberry Pi's IP Address:
   - Use a network scanner like Advanced IP Scanner or check your router's admin panel for connected devices.
   - Look for a device named raspberrypi.
2. Download and Install PuTTY
3. Connect to Raspberry Pi:
   - Open PuTTY and enter the Raspberry Pi's IP address in the Host Name (or IP address) field.
   - Set Port to 22 and Connection type to SSH.
   - Click Open.
   - When prompted, log in with the default credentials:
     - Username: pi
     - Password: raspberry (you may be prompted to change this on the first login).

Part 3: Set Up RealVNC for GUI Access

1. Enable VNC on Raspberry Pi:
   - In the PuTTY session, run the following commands:

     sudo raspi-config
   - Navigate to Interfacing Options > VNC and enable it.
   - Exit and reboot if prompted:

     sudo reboot
2. Download and Install RealVNC Viewer
3. Connect to the Raspberry Pi via RealVNC:
   - Open RealVNC Viewer on your laptop.
   - Enter the Raspberry Pi's IP address and press Enter.
   - Log in using the same credentials as for SSH (pi and the password you set).

Steps to enable and use the camera on a Raspberry Pi:

Step 1: Connect the Camera Module

1. Attach the Camera Module:
   - Turn off the Raspberry Pi.
   - Locate the Camera Serial Interface (CSI) port on the Raspberry Pi.
   - Insert the ribbon cable from the camera into the CSI port with the shiny contacts facing the Raspberry Pi's circuit board.
   - Secure the ribbon cable by gently pressing down the connector clip.

Step 2: Enable the Camera in Raspberry Pi OS

1. Boot the Raspberry Pi: Power on the Raspberry Pi.
2. Open the Raspberry Pi Configuration Tool: Open a terminal or access the desktop menu and navigate to Preferences > Raspberry Pi Configuration.
3. Enable the Camera: In the configuration tool:
   - Go to the Interfaces tab.
   - Find Camera and toggle it to Enable.
   - Click OK and reboot the Raspberry Pi.
   - Alternatively, enable the camera via the terminal: sudo raspi-config
   - Navigate to Interfacing Options > Camera.
   - Enable the camera and reboot when prompted.

3.5 Methodology

3.5.1 Image Acquisition Using Raspberry Pi Camera

The first step is to capture a high-resolution image of the fabric using the Raspberry Pi Camera Module. A resolution of 1920×1080 pixels is selected to ensure fine details of the fabric's weave pattern are clearly visible. Before capturing the image, a 2-second delay is introduced to allow the camera to stabilize and adjust to the lighting conditions, which helps in minimizing noise and motion blur. The camera is set to capture continuously so that multiple images can be processed dynamically, allowing for the best possible image selection.

This step is crucial because a sharp and well-exposed image is essential for accurately detecting fabric threads. A high-resolution image ensures that even the smallest details of the weave structure are visible, preventing any loss of information due to pixelation. The delay before capturing allows the camera to adjust its exposure settings properly, reducing the chances of capturing an overexposed or underexposed image. Additionally, continuous capture mode helps in adjusting parameters dynamically, which is useful in real-time applications where lighting and fabric texture may vary.

3.5.2 Cropping the Image to a 1x1 Inch Square

Once the image is captured, it is cropped to a 1x1 inch area to standardize the thread count measurement. The cropping is done around the center of the image to ensure that the analyzed section represents a uniform portion of the fabric, avoiding distortions or irregularities that might appear at the edges. The resolution is adjusted to 300 DPI (dots per inch) so that the measurement remains accurate, and the pixel-per-inch ratio is calculated accordingly. This step is important because standardization is necessary for accurate and consistent thread count calculation. If the fabric is analyzed over varying areas, the results might not be comparable across different samples. Cropping the image at the center helps in selecting a representative section that is free from distortions caused by fabric edges or folds. Furthermore, maintaining a fixed resolution of 300 DPI ensures that the thread density measurement remains consistent across different images, allowing for precise comparisons between various fabric samples.

3.5.3 Conversion to Grayscale and Contrast Enhancement

The captured image is then converted to grayscale, removing color information and retaining only intensity values. This simplifies image processing, as the focus is purely on detecting fabric threads rather than dealing with unnecessary color variations. After conversion, Contrast Limited Adaptive Histogram Equalization (CLAHE) is applied to enhance the contrast locally, making the weave structure more prominent. Additionally, multi-scale enhancement techniques are used to highlight both fine and thick threads in the fabric pattern.

This step is essential because grayscale conversion reduces complexity, allowing the algorithm to focus solely on thread detection without interference from color variations. Applying contrast enhancement techniques like CLAHE ensures that even faint thread structures become visible, which is particularly useful when analysing low- contrast fabrics. Multi-scale enhancement further refines the visibility of different thread types, ensuring that both fine and coarse weaves are detected accurately, leading to a more reliable thread count measurement.

3.5.4 Binary Thresholding for Dark Region Detection

To separate fabric threads from the background, a binary thresholding technique is applied, converting the grayscale image into a high-contrast black and white format. Black represents the thread regions, while white represents the gaps between them. To accommodate variations in lighting and fabric texture, Adaptive Gaussian Thresholding is used, dynamically adjusting the threshold values across different parts of the image. The resulting binary image is then inverted, making the thread gaps appear as white regions, which is necessary for the next stage of analysis.

Thresholding plays a critical role because it isolates the fabric's weave structure from the background, making it easier to count individual threads. Using a simple fixed threshold might not work well due to variations in lighting and fabric texture, which is why an adaptive approach is necessary. Adaptive Gaussian Thresholding ensures that different regions of the fabric are processed accurately, preventing overexposed or underexposed areas from affecting the analysis. Additionally, inverting the binary image helps in better edge detection, improving the precision of thread counting.

3.5.5 Projection and Peak Detection for Thread Identification

Once thresholding is complete, the density of threads along both the horizontal and vertical directions is analyzed. This is done by summing the pixel values along rows and columns to generate projection profiles, which represent the distribution of threads in the fabric. These projections are then smoothed using a Gaussian filter to eliminate noise and improve accuracy. Peak detection techniques are applied to identify the gaps between threads, with the number of peaks corresponding to the number of threads in each direction.

This step is crucial because projection profiles provide a structured way to measure thread density numerically, rather than relying on visual estimation. Smoothing the projection profiles ensures that small irregularities or distortions in the image do not affect the accuracy of the thread count. Peak detection helps in identifying the precise location of each thread, ensuring that the measurement is

not influenced by noise or artifacts in the image. By analyzing both horizontal and vertical projections, an accurate count of warp and weft threads is obtained, making the process highly reliable.

3.5.6 Visualization and Verification

To verify the accuracy of the detected thread count, the identified warp and weft threads are superimposed onto the original image using colored lines. Green lines indicate vertical (warp) threads, while blue lines indicate horizontal (weft) threads. In addition to this, the processed binary image and projection graphs are displayed to provide a graphical representation of the detected peaks. These graphs help in further validation by showing the detected thread gaps as peaks, allowing users to cross-check the accuracy of the measurement.

Visualization is an important step because it provides a clear confirmation of the detection process, ensuring that the measured thread count aligns with the actual weave structure. Superimposing detected threads onto the original image helps in identifying any errors in detection, allowing adjustments if necessary. The projection graphs offer a secondary layer of verification, showing the numerical representation of thread density, which can be compared with expected values. By combining visual confirmation with data validation, this step ensures that the final thread count is both accurate and reliable.

3.5.7 Calculation and Output of Thread Count

The final thread count is calculated by counting the number of detected peaks in both the horizontal and vertical projections. Since peaks represent the gaps between threads, an incremental correction factor is applied to ensure that all threads are counted correctly. The final output includes:

- The number of horizontal (warp) threads
- The number of vertical (weft) threads
- The total thread count (warp + weft)

This step is essential because it provides an objective and precise measurement of fabric thread density, which is a key factor in textile quality assessment. Counting both warp and weft threads ensures that the full structure of the fabric is analyzed, leading to an accurate representation of the weave pattern. The use of an incremental correction factor prevents miscalculations due to missing or overlapping thread detections. The final output allows manufacturers, researchers, and quality controllers to assess fabric quality systematically, ensuring consistency across different textile samples.

3.6 Implementation

This section describes the step-by-step implementation of the thread detection system using a Raspberry Pi Camera and OpenCV for image processing. The implementation follows the methodology outlined in the previous section, ensuring precise detection of warp and weft threads.

System Setup and Image Acquisition

The system utilizes a Raspberry Pi Camera Module to capture high-resolution images of fabric. The camera is initialized with a resolution of 1920×1080 pixels, and a 2-second delay is introduced to stabilize the exposure settings before capturing an image. The camera operates in a continuous mode to ensure multiple images can be captured dynamically, improving accuracy.

```python
from picamera import PiCamera
from picamera.array import PiRGBArray
from time import sleep
import cv2
import numpy as np

camera = PiCamera()
camera.resolution = (1920, 1080)
raw_capture = PiRGBArray(camera)
sleep(2)  # Allow camera to adjust to lighting conditions
camera.capture(raw_capture, format="bgr")
image = raw_capture.array
```

Figure 3.7.1 Image Acquisition

Image Preprocessing

After capturing the image, it is cropped to a 1×1 inch square from the center to ensure uniform thread count analysis. A practical Pixels Per Inch (PPI) value of 300 DPI is used to maintain accuracy. The cropped image is resized if necessary to ensure consistency across different fabric samples.

```python
def crop_1x1_inch(image):
    """Crop a 1x1 inch square from the center."""
    ppi = calculate_ppi(image.shape[1])
    height, width = image.shape[:2]
    center_x, center_y = width // 2, height // 2
    half_size = ppi // 2

    x1 = max(center_x - half_size, 0)
    y1 = max(center_y - half_size, 0)
    x2 = min(center_x + half_size, width)
    y2 = min(center_y + half_size, height)

    cropped_image = image[y1:y2, x1:x2]

    if cropped_image.shape[0] != ppi or cropped_image.shape[1] != ppi:
        cropped_image = cv2.resize(cropped_image, (ppi, ppi))

    return cropped_image, ppi
```

Figure 3.7.2 Cropping of image

Grayscale Conversion and Contrast Enhancement

To simplify thread detection, the image is converted to grayscale, and Contrast Limited Adaptive Histogram Equalization (CLAHE) is applied for contrast enhancement. Multi-scale enhancement is used to highlight both fine and coarse threads

```python
def enhance_contrast(image):
    """Convert to grayscale and apply CLAHE for enhancement."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    denoised = cv2.fastNlMeansDenoising(gray, None, h=10, searchWindowSize=21)

    # Multi-scale enhancement
    blurred1 = cv2.GaussianBlur(denoised, (3, 3), 0)
    clahe1 = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(4, 4))
    enhanced1 = clahe1.apply(blurred1)

    blurred2 = cv2.GaussianBlur(denoised, (5, 5), 0)
    clahe2 = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    enhanced2 = clahe2.apply(blurred2)

    enhanced = cv2.addWeighted(enhanced1, 0.6, enhanced2, 0.4, 0)
    return enhanced
```

Figure 3.7.3 Conversion of the image to Grayscale

Binary Thresholding for Thread Segmentation

To separate the threads from the background, Adaptive Gaussian Thresholding is applied, ensuring robust detection despite variations in lighting and fabric texture. The binary image is inverted so that thread gaps appear as white regions, making thread counting easier.

Thread Density Estimation

A method is implemented to classify the fabric as densely woven or loosely woven by calculating the ratio of white pixels in the binary image.

```python
def estimate_fabric_density(binary_image):
    """Estimate if the fabric is densely or loosely woven."""
    white_pixel_ratio = np.sum(binary_image > 0) / binary_image.size
    return white_pixel_ratio > 0.3  # Empirical threshold
```

Figure 3.7.4 Estimation of the thread density

Thread Projection and Peak Detection

The thread density is measured by computing horizontal and vertical projections, where pixel values are summed along each axis. The projection profiles are smoothed using a Gaussian filter, and peak detection is applied to identify thread gaps.

Visualization and Output

The detected warp and weft threads are overlaid on the original image using colored lines, with green for warp (vertical) threads and blue for weft (horizontal) threads.

Final Thread Count Calculation

The final thread count per inch (TPI) is calculated based on the number of detected peaks in the warp and weft directions.

```
def calculate_thread_count(peaks_x, peaks_y):
    """Calculate final thread count based on detected peaks."""
    total_threads = peaks_x + peaks_y
    print(f"Warp Threads: {peaks_x}, Weft Threads: {peaks_y}, Total Thread Count: {total_threa
```

Figure 3.7.5 Thread count calculation

3.7 Results

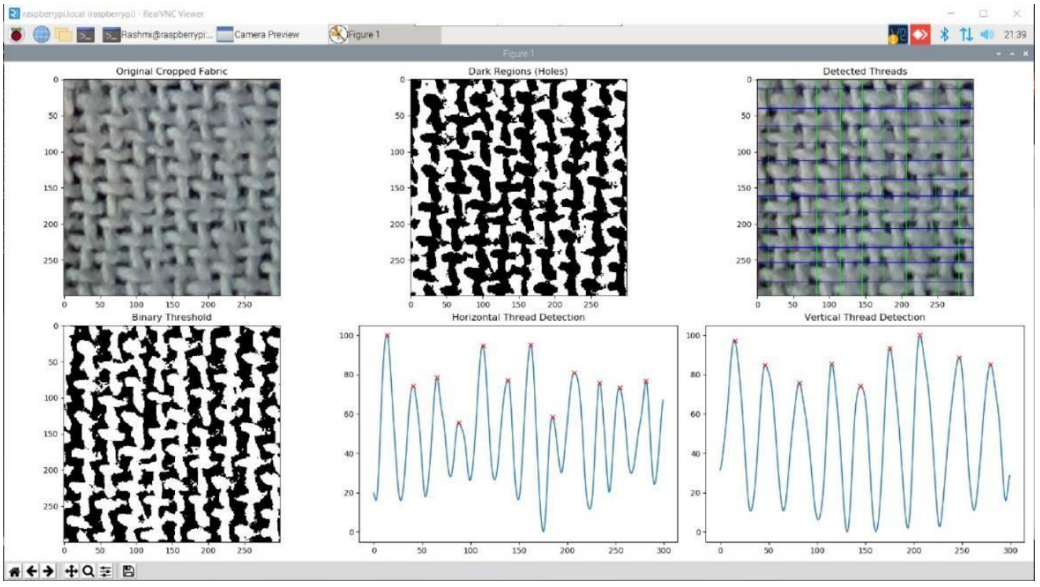

Figure 3.8.1 Detection and visualization of the detected thread

The image showcases a fabric thread detection system implemented using image processing techniques on a Raspberry Pi. It consists of multiple subplots, each representing a different stage in detecting the woven pattern of the fabric. The top-left subplot displays the original grayscale image of a cropped fabric sample, revealing the weave structure with visible horizontal and vertical threads. Moving to the top-middle subplot, a binary thresholding operation has been applied to highlight dark regions or

holes in the weave, where black regions represent gaps, and white regions correspond to fabric threads. The top- right subplot presents the detected threads overlaid with a grid, where blue and green lines indicate thread positions, allowing for a structured visualization of the weave pattern.

In the bottom-left subplot, another binary thresholding technique is used to further segment the image, simplifying thread detection by distinguishing threads from gaps. The bottom-middle and bottom-right subplots display the results of frequency-based analysis for horizontal and vertical thread detection, respectively. Peaks in the plotted signals, marked with red ‘x’ symbols, represent detected thread positions in both directions. This approach ensures precise identification of the fabric's woven structure.

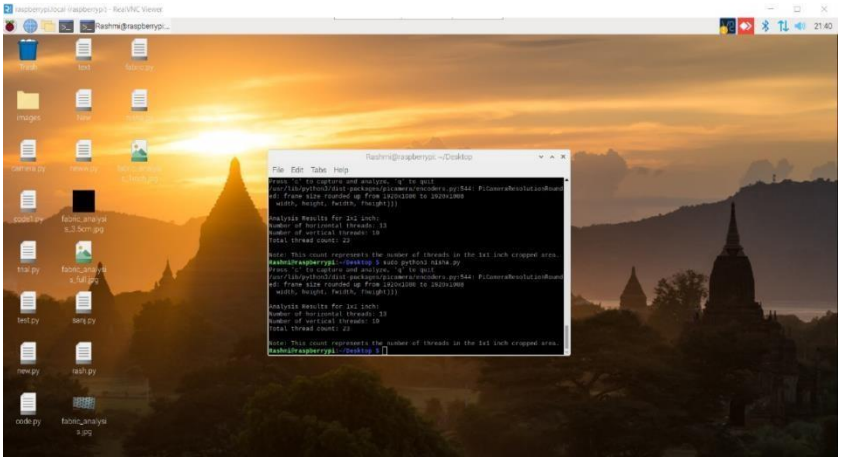

Figure 3.8.2 Output results

The analysis results are displayed in the terminal, showing the detected number of horizontal and vertical threads, along with the total thread count. The process is repeated in two separate executions, with slight variations in the detected vertical thread count, indicating possible variations in fabric structure or minor detection differences. The message at the end of each execution clarifies that the detected numbers represent the threads in the examined fabric section. The script also logs an informational message from the PiCamera module, adjusting the frame resolution from 1920x1200 to 1920x1088, ensuring compatibility. This terminal output suggests that the program is successfully detecting fabric threads, likely using image processing methods such as edge detection and thresholding, making it useful for fabric quality assessment or defect detection in textile industries.

CHAPTER 4

REFLECTION NOTES

CHAPTER 4

Reflection Notes

4.1 Project Assignment and Planning

Developing an automated fabric thread count detection system required a structured approach to tackle different phases of the project, from understanding fabric properties to building an accurate detection algorithm and implementing a working prototype. The process involved deep research, multiple iterations of algorithm development, hardware integration, and troubleshooting real-world challenges.

This chapter provides a month-wise breakdown of the tasks performed, highlighting key challenges faced and the solutions implemented at each stage and the key learnings and takeaway

Month 1: Research and Understanding Thread Count

The first month was dedicated to understanding the importance of thread count, its role in fabric quality assessment, and the traditional methods used in textile industries. We started by researching why thread count is considered a key parameter and how it affects fabric strength, comfort, and breathability.

As we delved deeper, we realized that not all fabrics have the same weaving pattern. Some fabrics, like plain weave cotton, have clearly defined threads, while others, like satin weave, have a more complex structure where threads are harder to differentiate. This raised an important question—how can we develop a detection method that works reliably across different fabric types?

- Plain Weave – The simplest and most balanced weave, where one horizontal (weft) thread passes over one vertical (warp) thread alternately. This creates a tight, strong, and uniform texture, commonly found in cotton fabrics and linen.
- Twill Weave – Characterized by diagonal rib patterns, where every 2-3 weft threads pass over and under multiple warp threads in a staggered pattern. This makes the fabric more flexible and drapable, commonly used in denim, chinos, and workwear fabrics.
- Satin Weave – Known for its smooth and glossy surface, where every 3-4 weft threads pass over multiple warp threads before going under. This reduces the number of thread

intersections, giving it a luxurious shine, commonly seen in satin, silk, and high-end bedsheets.

Challenge: Understanding How Thread Count Translates into Fabric Quality

Initially, we questioned whether higher thread count truly meant better fabric quality or if it was just a marketing strategy. To understand this, we studied industry case studies and research papers, which revealed that thread count affects several key factors:

- Durability – Higher thread count fabrics are stronger and last longer.
- Softness & Comfort – More threads per inch create a smoother, silkier texture.
- Breathability – Lower thread count = more airflow (good for summer), while higher thread count traps heat (better for winter).
- Strength vs. Flexibility – Extremely high thread counts can make fabrics stiff and less breathable.
- Cost & Misleading Marketing – Some manufacturers inflate thread counts without improving fabric quality.

This research helped us understand why accurate thread counting is crucial for quality control and shaped our approach to building a reliable, automated measurement system.

Month 2: Algorithm Development and Preprocessing

With a strong foundation in place, we moved to **developing the detection algorithm**. Our initial plan was to use **Hough Line Transform (HLT)** to detect and count individual threads. This method works by identifying **straight lines** in an image, which seemed like a perfect match for our problem.

We structured our algorithm into key steps:

1. **Convert fabric images to grayscale** to remove color distractions.
2. **Apply noise reduction filters** to improve clarity.
3. **Detect edges using Canny Edge Detection** to highlight thread boundaries.
4. **Use Hough Line Transform** to count vertical (warp) and horizontal (weft) threads.

After implementing the first version of our algorithm, we ran tests on sample fabric images. **Unfortunately, the results were disappointing**—many threads were **missed, miscounted, or merged together**. The detection was **inconsistent**, and we quickly realized that **Hough Transform was struggling with fabric threads**, as they were **not always perfectly straight** and often had **breaks or overlaps**.

Challenges Faced & Solutions:

One major issue was that **our preprocessing wasn't enhancing the fabric structure properly**. If the threads were too thin or faint, they were **completely ignored** by the edge detection step. To fix this, we introduced **Contrast Limited Adaptive Histogram Equalization (CLAHE)** to improve contrast and **adaptive thresholding** to ensure all threads were visible.

Another issue was that **our Hough Transform parameters were not tuned for fabric textures**. The default settings were designed for detecting long, continuous lines, but fabric threads are often **short and closely packed**. We adjusted the parameters by:

- **Lowering minLineLength** to detect shorter thread segments.
- **Increasing maxLineGap** to allow small breaks in thread detection.

Even after these optimizations, **Hough Transform still struggled with curved or tilted threads**, leading to **inconsistent results**. This forced us to rethink our entire approach.

Month 3: Overcoming Hough Transform Failure &Developing an Alternative Approach

After multiple failed attempts, we took a step back and re-evaluated our detection method. We asked ourselves—instead of detecting the threads directly, what if we counted the gaps between them? This was a breakthrough moment! Instead of struggling to identify thin, overlapping threads, we focused on detecting the empty spaces (gaps) between them. The logic was simple:

- Number of threads per inch = Number of gaps + 1

Shifting our approach to gap detection made the process much simpler and more reliable. We modified our algorithm to:

- Identify thread gaps instead of threads themselves.
- Apply histogram-based projection analysis to detect patterns.
- Use Gaussian smoothing to remove noise and improve clarity.

Challenges Faced & Solutions:

One of the challenges was ensuring that gaps were accurately detected without being confused with random textures in the fabric. To solve this, we applied adaptive thresholding and tested multiple filtering techniques to refine the results.

Another challenge was validating our results against actual thread counts. We manually counted threads in different fabrics and compared them with our automated results. The accuracy was significantly higher than the Hough Transform method, confirming that gap detection was the right approach.

Month 4: Hardware Implementation & Final Prototype

With our software finalized, it was time to integrate it with hardware for real-world testing. We set up a Raspberry Pi with a Camera Module to capture fabric images, but this introduced a new set of challenges.One of the biggest issues was that the camera resolution wasn’t high enough to capture fine thread details. The fabric images appeared blurry, making detection difficult. After multiple failed attempts to enhance clarity, we came up with an effective solution:

Zoom into the fabric image by 8X and scale the results back to a 1-inch measurement.

By magnifying the image, we were able to capture finer thread details and get more precise results. We also had to troubleshoot multiple Raspberry Pi configuration issues, which required guidance from EC professors to fix.

Challenges Faced & Solutions:

1.Camera resolution was too low for fine thread detection.

Solution: Zoomed in 8X and scaled the results mathematically.

2. Raspberry Pi wasn't configured properly, leading to errors.

Solution: Debugged configuration settings and optimized camera parameters.

3.Real-time processing was slow.

Solution: Optimized the software pipeline to run efficiently on Raspberry Pi.

After resolving these challenges, we conducted extensive testing on different fabrics. The system was able to accurately measure thread count across various weaving patterns, proving that our gap-based approach was successful

4.2 Key Learnings and Takeaways

The development of the ML-Based Fabric Thread Count Measurement System provided valuable insights into image processing, hardware integration, and problem-solving in real-world applications. The key learnings and takeaways from this project include:

1. Importance of Preprocessing in Image Analysis

- Proper image preprocessing techniques (grayscale conversion, noise reduction, contrast enhancement) are crucial for accurate feature detection.
- LAB color space transformation significantly improves contrast and enhances thread visibility.
- Peak detection methods are highly effective for counting repetitive patterns like fabric threads.

2. Optimizing Algorithm Performance for Hardware Limitations

- Raspberry Pi has limited processing power, requiring optimized image resolutions and efficient algorithms to maintain real-time performance.
- Preprocessing images in smaller regions reduces computational load and speeds up thread detection.
- Balancing accuracy and efficiency is key to making ML-based systems practical for industrial applications.

3. Handling Real-World Challenges in Image Processing

- Lighting conditions and fabric texture variations affect image clarity and detection accuracy.
- Fine-tuning parameters (edge detection thresholds, Hough Transform settings) is essential for different fabric types.

- Alternative approaches like projection-based analysis can solve issues that traditional methods struggle with.

4. Adaptability and Scalability of Automated Systems

- The system can be adapted for various fabric textures by adjusting preprocessing and detection techniques.
- Automation enhances consistency and reduces human error in quality control.
- Future improvements could include deep learning models for fabric classification and enhanced thread analysis.

5. Industry Relevance and Practical Impact

- Automating fabric thread count measurement improves efficiency, accuracy, and productivity in the textile industry.
- The system offers a cost-effective solution compared to expensive commercial fabric analyzers.
- The knowledge gained can be applied to other image processing-based quality assessment systems in manufacturing.

The project demonstrated the power of automation in textile quality control, highlighting how image processing and machine learning techniques can replace traditional manual methods. The challenges encountered enhanced problem-solving skills, and the successful implementation proved that even resource-constrained hardware like Raspberry Pi can be leveraged effectively with the right optimizations.

Executive Summary

The ML-Based Fabric Thread Count Measurement System internship was a transformative experience that provided deep insights into image processing, hardware integration, and automation in textile quality control. The objective of the project was to develop an automated, accurate, and efficient thread counting system using Raspberry Pi and high-resolution imaging, addressing the limitations of traditional manual methods.

The project followed a structured development process, progressing through research, software development, problem-solving, and hardware integration. Key techniques such as grayscale conversion, noise reduction, edge detection, and peak detection were implemented to enhance thread visibility. The Hough Transform, initially used for line detection, presented challenges such as noise misidentification and broken thread detection, which were resolved through contrast enhancement (CLAHE), morphological operations, and projection-based analysis.

Throughout the internship, various challenges were encountered, including hardware limitations of Raspberry Pi, lighting inconsistencies, and computational efficiency. These were addressed by optimizing image resolution (1x1 inch at 300 PPI), implementing adaptive peak detection, and refining processing algorithms to balance accuracy and efficiency. The final system successfully automated thread counting with high precision, demonstrating its potential as a cost-effective and scalable solution for the textile industry.

This internship provided invaluable technical, analytical, and problem-solving experience, emphasizing the practical application of machine learning and image processing in industrial automation. The knowledge gained during this project not only enhanced the understanding of fabric analysis techniques but also highlighted the broader impact of computer vision in quality control processes. The developed system lays the foundation for future improvements, including deep learning-based fabric classification and further enhancements in detection accuracy.

REFERENCES

[1] **Harwood, J., & Salik, M. (2017).** "Automated fabric defect detection: A review of advancements and challenges." *Textile Research Journal, 87(5), 591-611.* [DOI: 10.1177/0040517516651102]

[2] **Kumar, A. (2008).** "Computer-vision-based fabric defect detection: A survey." *IEEE Transactions on Industrial Electronics, 55(1), 348-363.* [DOI: 10.1109/TIE.2007.909051]

[3] **Zhu, Y., & Huang, J. (2015).** "Weave pattern analysis and defect detection using image processing techniques." *Pattern Recognition Letters, 52, 142-151.* [DOI: 10.1016/j.patrec.2014.11.007]

[4] **Mok, P. Y., & Kwok, Y. L. (2014).** "Intelligent textile inspection using image processing and machine learning techniques." *Computers & Industrial Engineering, 76, 180-190.* [DOI: 10.1016/j.cie.2014.07.016]

[5] **Yuen, C. W. M., & Kan, C. W. (2009).** "Fabric quality assessment using digital image processing." *Fibers and Polymers, 10(2), 250-255.* [DOI: 10.1007/s12221-009-0250-7]

[6] **Jain, R. (2021).** *Introduction to Computer Vision and Image Processing.* Springer. ISBN: 978-3-030-78244-5

[7] **Fukuda, S. (2019).** *Machine Vision for Textile Applications.* Woodhead Publishing. ISBN: 978-0-08-101225-1

[8] **Parker, J. R. (2011).** *Algorithms for Image Processing and Computer Vision.* Wiley. ISBN: 978-0470643853

[9] **Raspberry Pi Foundation (2023).** "Using PiCamera for Computer Vision Applications." [Available at: https://www.raspberrypi.org/documentation/]