



---

# FABRIC INSPECTION BASED ON SHRINKAGE

---

GROUP 06



DECEMBER 15, 2023

## Group 06

Member Name	Registration Number
H.A. Sandali Kavindi	S/18/843
M.M.S. Devapriya	S/18/844
U. Jathunan	S/18/845
K.B.J.D. Wijerathna	S/18/846
W.N.M. De Silva	S/18/847
J.N. Jullion Mathukaran	S/18/848
Y. Shakiraj	S/18/850
K.P.K.J. Wimalaweera	S/18/SP/851

## **ACKNOWLEDMENT**

Our deepest gratitude is extended to Dr. Erunika O. Dayartna and Prof. W.B. Daundasekera, Dr. Niluka Rodrigo our project supervisors, for their steadfast advice, knowledge, and assistance throughout the project. Their helpful criticism and support have been priceless.

I would like to thank Brandix Company for their cooperation and provision of vital resources, which greatly aided in this project's success. We would like to thank Mr. Melvin who committed himself to providing the necessary information for this project.

I express my gratitude to all those who contributed to making this project feasible.

Please feel free to change the phrasing to better reflect the content and tone of your report, as well as the input from Brandix and your supervisor.

The collaboration with Brandix has improved our work quality and given us practical insights that have been essential to the project's success.

# Contents

ACKNOWLEDGMENT .....	2
ABSTRACT.....	5
CHAPTER 01 .....	6
1.1.    COMPANY OVERVIEW .....	6
1.2.    PRODUCTION PROCESS (Fabric Inspection Process).....	7
1.3.    PROBLEM STATEMENT .....	9
1.3.1.    MEASURING THE SHRINKAGE .....	9
1.3.2.    WHAT IS MEANT BY SHRINKAGE? .....	9
1.3.3.    MAJOR ISSUES IN BRANDIX'S CURRENT PROCEDURE.....	10
CHAPTER 02 .....	11
2.1. Methodology .....	11
2.1.1. Current Approach .....	11
2.1.2. Our Approach (Automated Process).....	12
2.1.3. Code Implementation .....	14
CHAPTER 03 .....	20
3.1. Results And Discussion.....	20
3.1.1. Output of the code .....	20
3.1.2. Analysis using R.....	21
3.1.3. Discussion.....	23
CHAPTER 04 .....	25
4.1. Conclusion.....	25
4.1.1. Achievements .....	25
4.1.2. Difficulties .....	25
4.2. Research Findings .....	26
4.3. References .....	27

# Figures

Figure 1 :Scale interpretation.....	8
Figure 2 : Four-point inspection system. ....	8
Figure 3: Fabric inspection Machine .....	8
Figure 4: Rejection process.....	9
Figure 5: Bluetooth Tape .....	10
Figure 6: Shrinkage board.....	10
Figure 7: Current Excel Sheet.....	10
Figure 8: Pillow Cases .....	11
Figure 9: Marking process .....	11
Figure 10: Setup.....	12
Figure 11: Web camera.....	13
Figure 12: Output plots .....	18
Figure 13: Output Result.....	20

## **ABSTRACT**

This report details the MT325 course project, which was customized for Brandix and involved developing and implementing an automated system for assessing fabric shrinkage. The project aims to decrease the duration and improve the efficacy of cloth roll shrinkage measurement in Brandix's textile manufacturing procedures. The system seeks to optimize the measuring workflow by integrating cutting-edge automation technologies, providing a dependable and efficient solution. The project's goals, methods, and the automated system's salient features are delineated in the study, which also highlights the system's ability to optimize operations and boost the precision of Brandix's fabric shrinkage measures.

# CHAPTER 01

## 1.1. COMPANY OVERVIEW

Brandix is Sri Lanka's largest garment exporter, with a yearly revenue of more than \$600 million. The company employs about 43,000 Associates and has 38 manufacturing plants in Sri Lanka, India, and Bangladesh. Through its backward-linked



businesses in textiles, thread, buttons, and hangers, the Group adds more than 50% of value locally. A look at an exclusive portfolio reveals Victoria's Secret, Gap, Next, and Marks and Spencer, among other great companies. The company specializes in casual bottoms, intimate and athletic clothing, woven and knitted textiles, and a wide range of garment industry accessories. Producing our own fabric, buttons, and hangers gives us a tactical advantage in textiles and reinforces our fundamental competencies of sophisticated research and development, great design, fabric printing, washing, dyeing, wet processing, and finishing.

### Company Information

Company Name - BRANDIX APPAREL SOLUTIONS LTD

Founded - 1978

Products / Services Range - Garment Items, fabric printing, washing, dyeing, wet processing, finishing

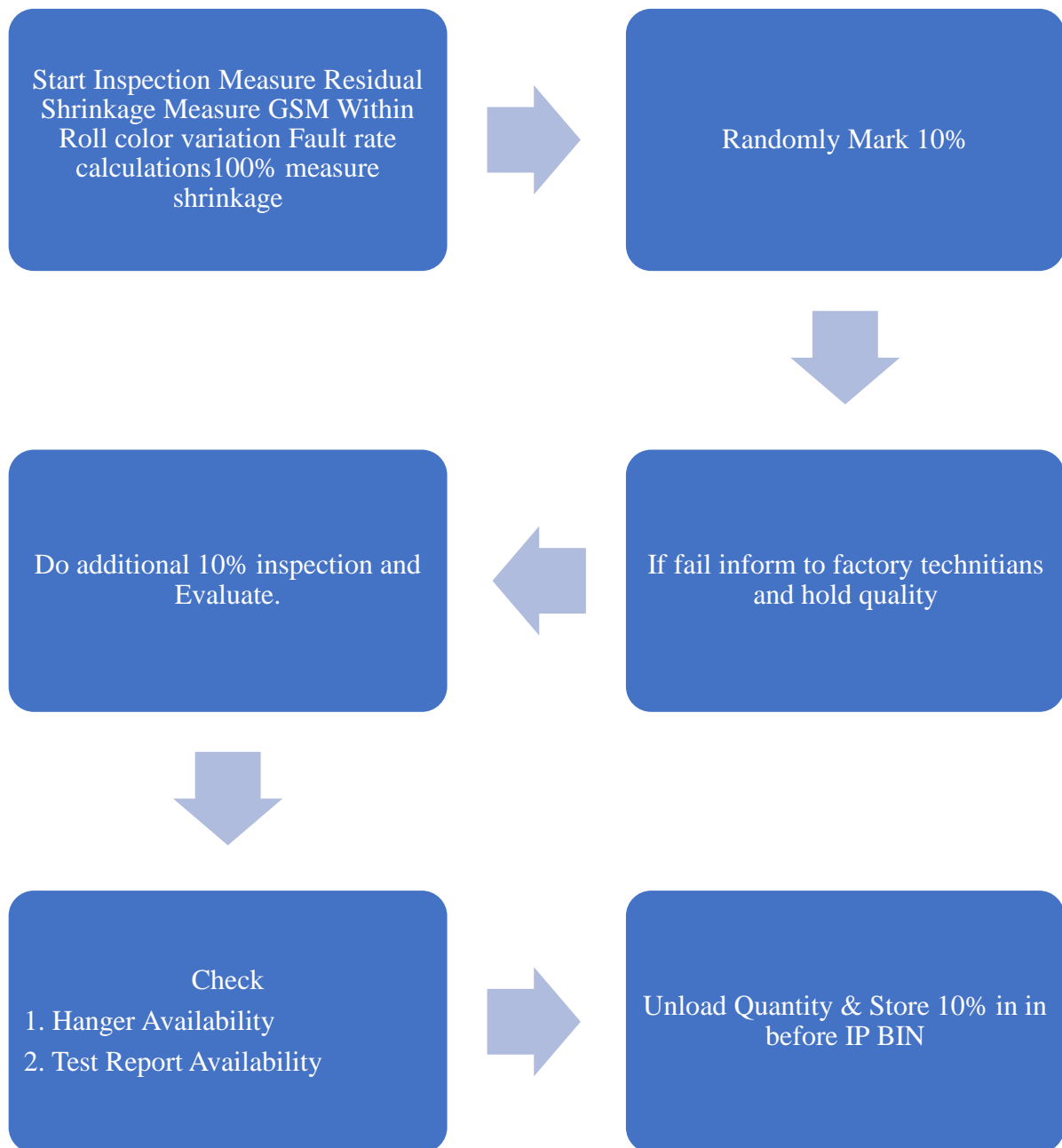
#### General Contact Info

Address - No.21, Temple Road Ekala Ja-Ela

Telephone - (94) 11-4735100 / (94) 11-4735111

Fax - (94) 11-4735412

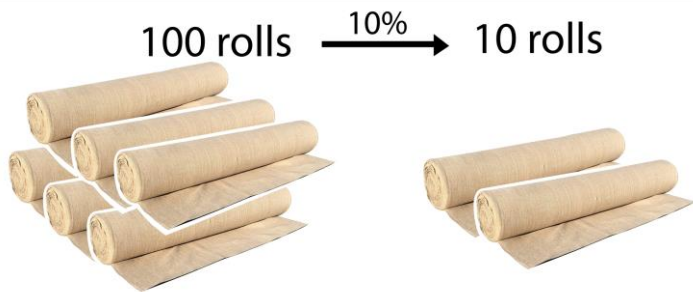
## 1.2. PRODUCTION PROCESS (Fabric Inspection Process)





## HOW IS IT DONE

01. When fabric is unload select 10% of fabric of that and after do visual inspection process.



*Figure 1 :Scale interpretation.*

02. In the visual inspection process. They use **four-point inspection** system to identify the defects on fabric.

Length of Defect	Defect Points
3 Inches or less	1
3 to 6 Inches	2
6 to 9 Inches	3
Over 9 Inches	4

*Figure 2 : Four-point inspection system.*

If the total points are greater than a certain threshold. They reject the sample.



*Figure 3: Fabric inspection Machine*

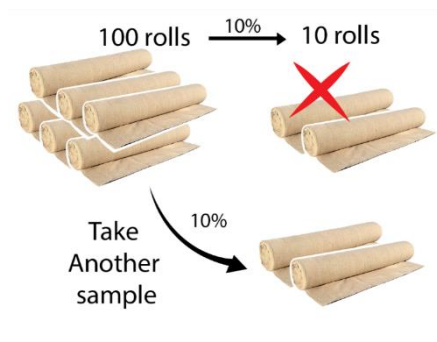
03. After defects identifying process if it fails do another 10% inspection for selected fabric from the whole fabric. Because in garment field it is very difficult reject and return fabric. So always when

they proceed the fabric inspection process they keep the option for the proceed fabric. Under visual inspection process they check whether shadings exist within the roll color.



04. Color Shading Under visual inspection process they check within the roll whether color shadings exist.

- But since they now have a machine which scans the color shadings, they do not require our support for the color Shading part.



*Figure 4: Rejection process*

### 1.3. PROBLEM STATEMENT

#### 1.3.1. MEASURING THE SHRINKAGE

Brandix is currently measuring shrinkage through a manual procedure, a process that consumes significant human resources and results in the unnecessary use of paper and time. To enhance efficiency and reduce environmental impact, there is a desire to automate this procedure.

#### 1.3.2. WHAT IS MEANT BY SHRINKAGE?

Shrinkage of cloth refers to the reduction in size or dimensions of a piece of fabric after Washing, dying of fabric procedure, and when exposed to the relaxing of fabric.

### 1.3.3. MAJOR ISSUES IN BRANDIX'S CURRENT PROCEDURE

2. The accuracy of measurements with the Bluetooth tape relies on manual procedures performed by individuals.
3. There is a possibility of obtaining inaccurate results if the tape is not consistently positioned in the same location each time. This is a hands-on process, and if the tape is not consistently placed, the measurements may not be accurate.



Figure 5: Bluetooth Tape

Process	Average Time
Manual Process	60 seconds
Automated process	20 seconds

As one can see, time is saved by a considerable amount when each roll is measured, thus increasing the efficiency of the process when a whole batch is taken into consideration.



Figure 6: Shrinkage board

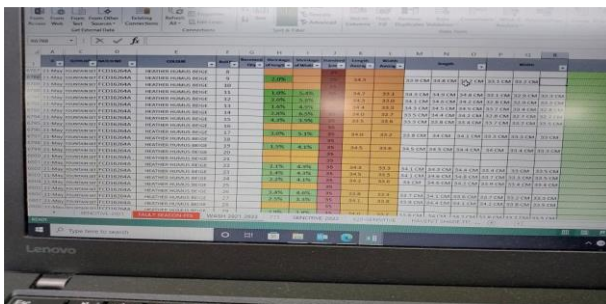


Figure 7: Current Excel Sheet

## CHAPTER 02

### 2.1. Methodology

#### 2.1.1. Current Approach

##### 2.1.1.1. Pillow Case process

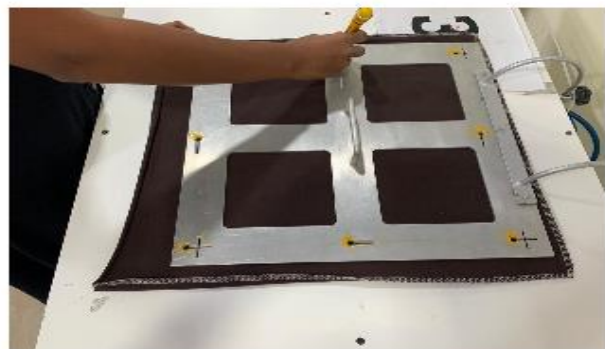
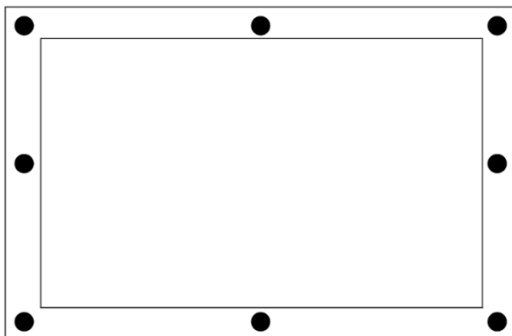
1. Take 1 yard of cloth of piece from each roll and make it a pillowcase. (This piece of cloth includes batch number as well as roll number).
2. Take that piece of fabric and mark 8 points using a textile pen using a standard shrinkage board. Those measurements are taken from Bluetooth tape which is a machine that will update the measurements in Excel automatically.



*Figure 8: Pillow Cases*

##### 2.1.1.2. Marking

The points are marked on each fabric pillow case manually by hand using a yellow marker.



*Figure 9: Marking process*

3. Then send it to the wash and dye.

##### 2.1.1.3. Measuring

4. Keep that cloth to dry and once received fabric check whether distance between points that measured before using Bluetooth tape by man.

5. Then they measure the variation. (0.5 cm differs it 1.5% effect, 1 cm differs it 3% effect likewise)
6. That variation is sent to the technical team and according to that variation they design how the rolls are used.

## 2.1.2. Our Approach (Automated Process)

### 2.1.2.1. Camera Initialization and Image Capture:

You can see the below Setup we have used for this process.

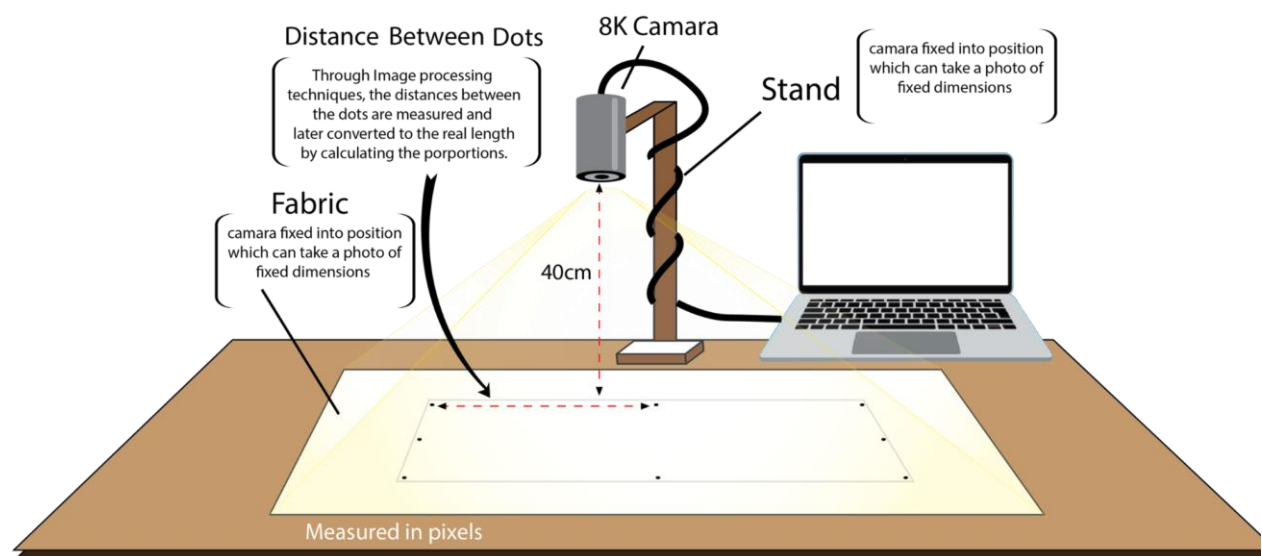
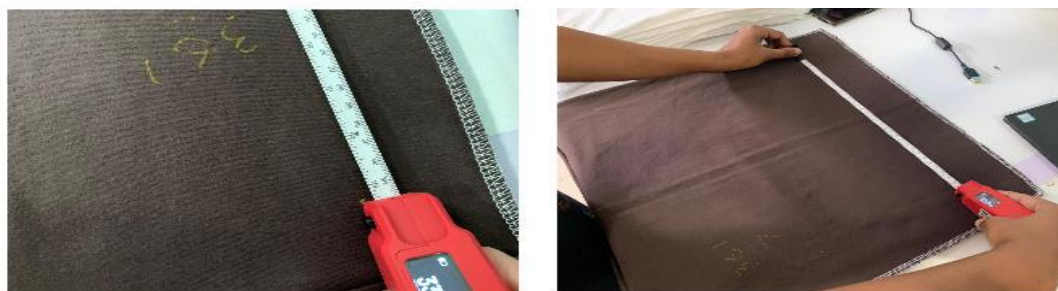


Figure 10: Setup

After having discussions with the relevant members of the group, 8k webcam is suggested for this process. Because using an 8k webcam we can get a quality image. You can refer the camera using below link.

<https://www.walmart.com/ip/DESTYER-8K-Full-High-Definition-Webcam-Autofocus-Fill-Light-Camera-with-Mic-Computer-Camera/1677769306>



#### 2.1.2.2. Camera Model selection:



33MP high-quality image can be extracted using the above webcam model. The calculated resolution is shown below.

Figure 11: Web camera

## Megapixel calculator

A "megapixel" is one million (1,000,000) pixels. Enter the pixel dimensions of your photo, camera, screen or tv, and calculate how many megapixels that is.

Width x Height

7680

4320

Calculate!

### Calculate megapixels

Pixel resolution

- Size = 7680 x 4320 = 33,177,600 pixels
- Pixel count = 33.18MP pixels
- Resolution = 33.18MP pixels

Aspect Ratio

- Width / Height: 1.78
- "landscape" (horizontal) orientation
- Type of aspect ratio = "HD 16:9"

The advantages of using an 8K Full HD High Definition Webcam can be stated as follows.

- **Autofocus Technology**  
Autofocus in maintaining sharp and clear images during video communication.
- **Ease of Use**  
Webcams are user-friendly and require minimal technical expertise. They are suitable for individuals who may not be familiar with more advanced camera settings.
- **Real-Time Preview**  
Webcams often provide a real-time preview on the computer screen, allowing users to compose the shot and adjust framing before taking the photo.

Weaknesses when using a web camera are as follows,

- **Limited Focus Range**  
Some webcams have a fixed focus range optimized for video calls. As a result, they may



struggle to focus on objects at varying distances when used for photography.

- **Low Image Quality**

Webcams are primarily designed for video conferencing and may not deliver the same image quality as dedicated digital cameras. The resolution and color reproduction may be limited.

- **Limited Manual Controls**

Webcams typically lack advanced manual controls that photographers might need for adjusting settings such as aperture, shutter speed, or ISO. This limits creative control over the captured images.

- **Shutter Lag**

Webcams may have a noticeable delay between the moment you click the capture button and when the photo is taken. This can result in missed moments, especially for fast-paced subjects.

- **Low Light Performance**

Many webcams struggle in low-light conditions. They might produce grainy or noisy images when the lighting is not optimal.

### 2.1.3. Code Implementation

#### Camera Initialization and Image Capture

Image capturing is done using the cv2 package and the camera port is assigned after checking with the available devices for the laptop.

This section of code initializes the camera using OpenCV and captures a frame from the image.

```
7   # Load the image
8   #image = cv2.imread('Fabric/dots7.jpg', cv2.IMREAD_GRAYSCALE)
9
10  cam = cv2.VideoCapture(0)
11
12  # reading the input using the camera
13  result, image = cam.read()
```


## Convert To Grayscale.

Then the image is converted to a greyscale image to reduce the color variation and to reduce the unwanted patterns and colors of the image.

```
16 # Convert the image to grayscale
17 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

The "cv2.cvtColor" function of the OpenCV library is used and the BGR colored pixels in the image are converted to GRAY.

Now the image only contains the colors from 0 to 255 of a single color.

0  255

Now in order to remove the unnecessary details from the image we can blur the image.

## Blur the image

```
19 # Apply Gaussian blur for noise reduction
20 image_blurred = cv2.GaussianBlur(gray_image, (5,5),0)
```

5 square pixels are used to blur the image and remove the unnecessary noise in the image. Here the resulting grayscale image is blurred using the GaussianBlur function in the OpenCV package.

## Creating a Binary Image

Then the blurred image is converted to binary by giving two threshold values as lower bound and upper bound. These variables can be adjusted according to the intensity of the environment and the quality of the image captured.

```
22 # Define the lower and upper bounds for the intensity range
23 lower_bound = 0 # Adjust this value for your specific intensity range
24 upper_bound = 100 # Adjust this value for your specific intensity range
25
26 # Create a binary mask based on the intensity range
27 binary_mask = np.logical_and(image >= lower_bound, image <= upper_bound).astype(np.uint8) * 255
```

Then the binary mask is converted into binary image.

```
31 binary_mask_gray = cv2.cvtColor(binary_mask, cv2.COLOR_BGR2GRAY)
32 _, binary_image = cv2.threshold(binary_mask_gray, 127, 255, cv2.THRESH_BINARY)
```

Then the binary image is created with 127 placed as the threshold value.

Now the image only contains two sets of colors, 0 and 255

## Drawing contours

Here when the binary image is created the dots are filtered. And the contours are drawn in the image.



```

34 # Find contours in the binary image
35 contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

```

And then the contours are filtered according to their sizes by giving two threshold values as minimum area and maximum area.

```

37 # Filter contours based on size or other criteria
38 min_area = 0
39 max_area = 300
40 # Adjust this value based on your requirements
41 filtered_contours = [contour for contour in contours if max_area > cv2.contourArea(contour) > min_area]

```

Then the filtered contours are considered, and centroids are marked on each dot and centroids are converted to coordinates and the contours are drawn in a copy of the original image using the drawContours function in OpenCV.

```

43 # Calculate centroids of filtered contours
44 centroids = [np.mean(contour, axis=0)[0] for contour in filtered_contours]
45
46 # Convert centroids to float32
47 centroids = np.float32(centroids)
48
49 # Draw contours on the original image
50 image_with_contours = image.copy()
51 cv2.drawContours(image_with_contours, contours, -1, (0, 255, 0), 2)

```

And then the centroids are connected together to create a polygon in order to calculate the area in pixels. and then the image with the polygon is saved and displayed later.

### Drawing the Polygon

The convexHull function is used to draw the contours and polylines function is used to draw the polygon.

```

53 # Calculate convex hull to order centroids correctly
54 hull = cv2.convexHull(np.array(centroids), clockwise=True, returnPoints=True)
55 hull = np.int32(hull)
56
57 # Create an empty black image
58 polygon_image = np.zeros_like(image)
59
60 # Draw the convex hull polygon
61 cv2.polylines(polygon_image, [hull], isClosed=True, color=255, thickness=2)
62
63 # Calculate the area of the polygon
64 polygon_area = cv2.contourArea(hull)

```

### Calculate Distances

Then a function is created to calculate the distances between each dot and store the data on a DataFrame.

```

66     distances_list = []
67     # Calculate distances between centroids and print them
68     for i in range(len(centroids)):
69         for j in range(i + 1, len(centroids)):
70             distance = np.linalg.norm(centroids[i] - centroids[j])
71             distances_list.append([f'Dot {i+1}', f'Dot {j+1}', distance])
72             print(f'Distance between Dot {i+1} and Dot {j+1}: {distance:.2f} pixels')

```

Inputting Roll Number

```

95     # Print the area of the polygon
96     print(f"Area of the polygon: {polygon_area} square pixels")
97
98     # Include the area in the distances list
99     distances_list.append(['Polygon Area', '', polygon_area])
100
101     # Convert the distances list to a DataFrame
102     distances_df = pd.DataFrame(distances_list, columns=['Dot 1', 'Dot 2', 'Distance'])

```

Exporting to the Excel file

After the roll number is asked from the user the polygon area and the distances are exported to an excel file. Then ExcelWriter function in the pandas library is used to create a new sheet to input the data in the saved data frame and using the if statement the function chooses to create a new excel file in the directory or append the data to the existing file by creating a new sheet in the name of the roll number if needed.

```

107     # Check if the Excel file exists
108     excel_file_path = 'distances.xlsx'
109     if not os.path.exists(excel_file_path):
110         # If the file doesn't exist, create a new Excel file
111         with pd.ExcelWriter(excel_file_path, engine='openpyxl', mode='w') as writer:
112             distances_df = pd.DataFrame(distances_list, columns=['Dot 1', 'Dot 2', 'Distance'])
113             distances_df.to_excel(writer, index=False, sheet_name=sheet_name)
114     else:
115         # If the file exists, append to the existing file
116         with pd.ExcelWriter(excel_file_path, engine='openpyxl', mode='a') as writer:
117             distances_df = pd.DataFrame(distances_list, columns=['Dot 1', 'Dot 2', 'Distance'])
118             distances_df.to_excel(writer, index=False, sheet_name=sheet_name)

```

Then several plots are created to oversee whether each image on the different stages is done, and the dots are identified correctly on the image.

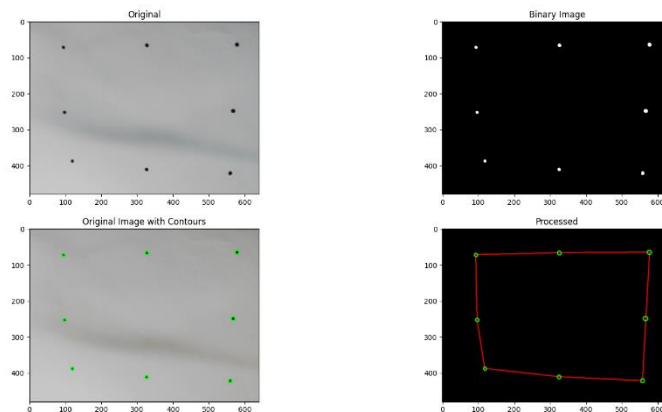
Plots

4 plots are drawn using the Matplotlib library. First the original captured image is displayed and then the binary image where the dots are identified, then the image with contours is displayed next and finally the polygon drawn from the centroids marked on the image.

```

120 #Showing the original image and processed image...
121 plt.subplot(221)
122 plt.imshow(image,'gray')
123 plt.title("Original")
124
125 # Display the binary image with contours
126 plt.subplot(222)
127 plt.imshow(binary_image, 'gray')
128 plt.title("Binary Image")
129
130 # Display the original image with contours
131 plt.subplot(223)
132 plt.imshow(cv2.cvtColor(image_with_contours, cv2.COLOR_BGR2RGB))
133 plt.title("Original Image with Contours")
134
135 plt.subplot(224)
136 plt.title("Processed")
137 plt.contour(binary_image,levels=[2],colors='lime')
138 plt.imshow(polygon_image)
139
140 plt.show()

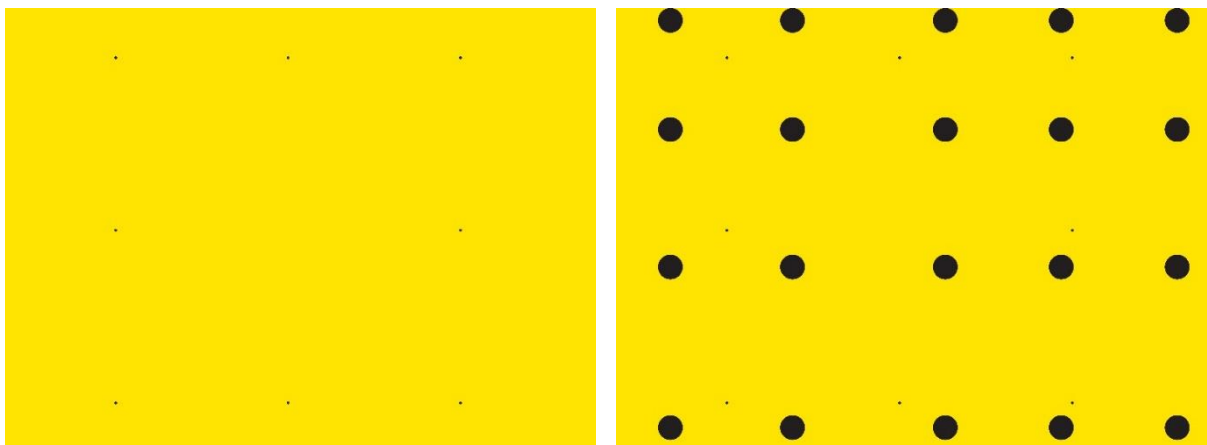
```

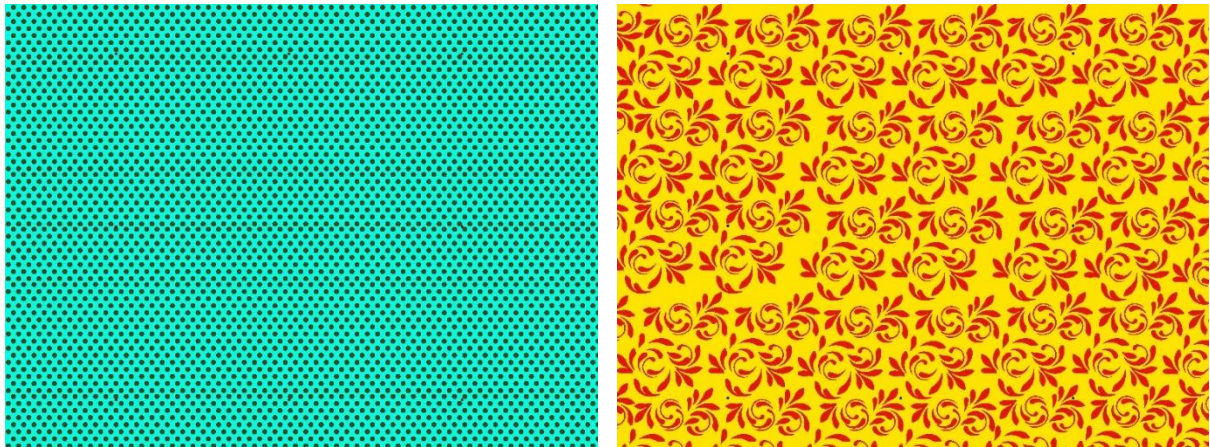


*Figure 12: Output plots*

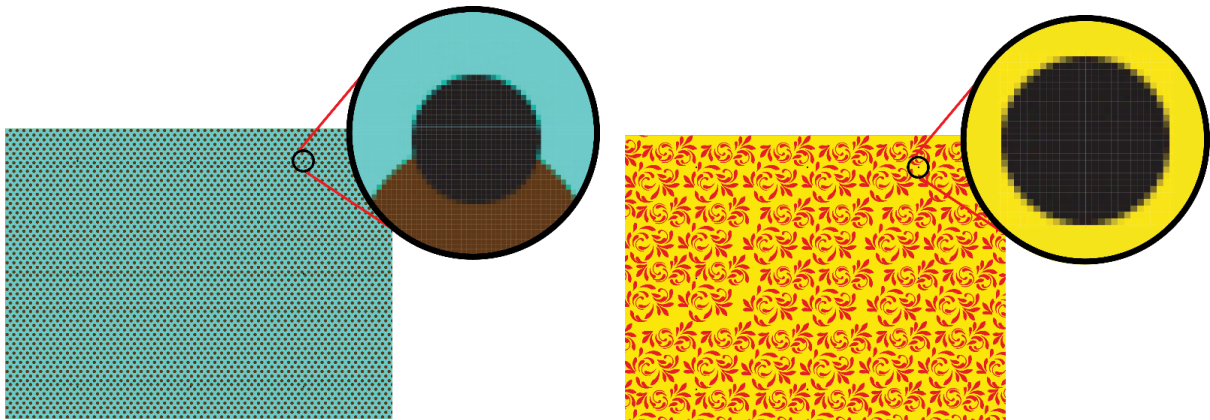
The following images were tested and proved to be successful, and the accuracy is well achieved after testing and code multiple times.

Here are some of the image's input for the code to measure the shrinkage.





Once the threshold values are adjusted, mainly the color of the marker and the size of an average marker, the code filters the pattern from the dots successfully and the measurements are given without any issue.



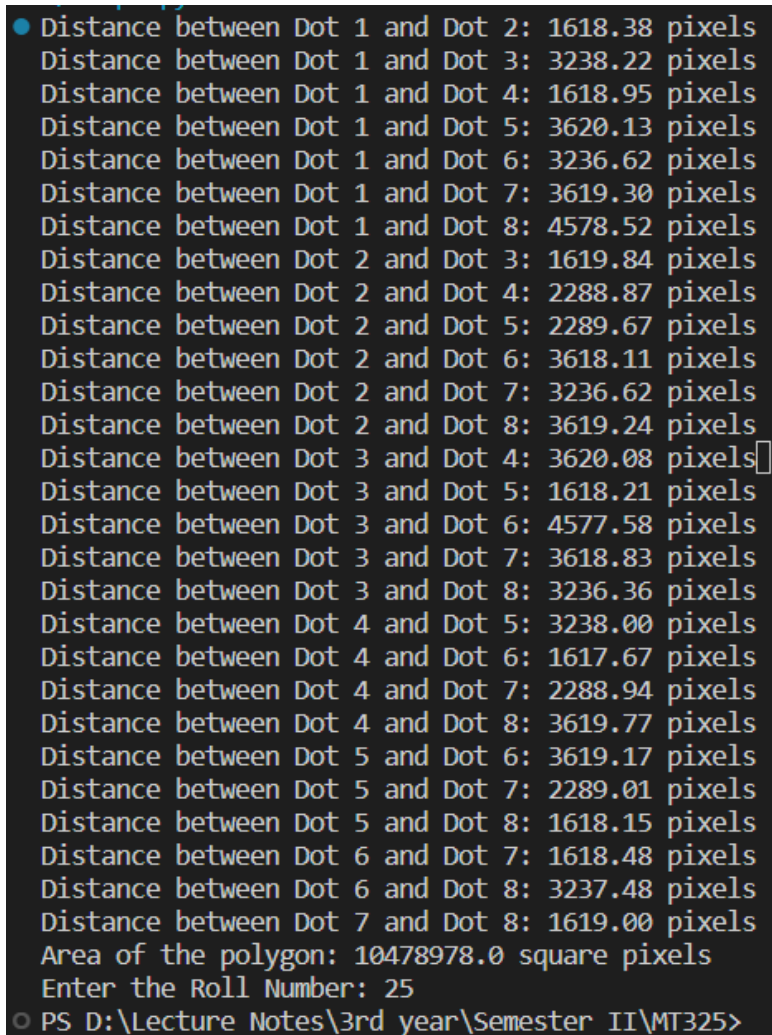
When dealing with fabric including various patterns, the chances of the markers overlapping with the pattern are high. As above, the code must carefully filter the dots among the pattern without the noise and other patterns structures.

## CHAPTER 03

### 3.1. Results And Discussion

#### 3.1.1. Output of the code

The exact output that is fed to the excel file is also printed on the console and can be viewed each time the code is run.



```
● Distance between Dot 1 and Dot 2: 1618.38 pixels
Distance between Dot 1 and Dot 3: 3238.22 pixels
Distance between Dot 1 and Dot 4: 1618.95 pixels
Distance between Dot 1 and Dot 5: 3620.13 pixels
Distance between Dot 1 and Dot 6: 3236.62 pixels
Distance between Dot 1 and Dot 7: 3619.30 pixels
Distance between Dot 1 and Dot 8: 4578.52 pixels
Distance between Dot 2 and Dot 3: 1619.84 pixels
Distance between Dot 2 and Dot 4: 2288.87 pixels
Distance between Dot 2 and Dot 5: 2289.67 pixels
Distance between Dot 2 and Dot 6: 3618.11 pixels
Distance between Dot 2 and Dot 7: 3236.62 pixels
Distance between Dot 2 and Dot 8: 3619.24 pixels
Distance between Dot 3 and Dot 4: 3620.08 pixels
Distance between Dot 3 and Dot 5: 1618.21 pixels
Distance between Dot 3 and Dot 6: 4577.58 pixels
Distance between Dot 3 and Dot 7: 3618.83 pixels
Distance between Dot 3 and Dot 8: 3236.36 pixels
Distance between Dot 4 and Dot 5: 3238.00 pixels
Distance between Dot 4 and Dot 6: 1617.67 pixels
Distance between Dot 4 and Dot 7: 2288.94 pixels
Distance between Dot 4 and Dot 8: 3619.77 pixels
Distance between Dot 5 and Dot 6: 3619.17 pixels
Distance between Dot 5 and Dot 7: 2289.01 pixels
Distance between Dot 5 and Dot 8: 1618.15 pixels
Distance between Dot 6 and Dot 7: 1618.48 pixels
Distance between Dot 6 and Dot 8: 3237.48 pixels
Distance between Dot 7 and Dot 8: 1619.00 pixels
Area of the polygon: 10478978.0 square pixels
Enter the Roll Number: 25
○ PS D:\Lecture Notes\3rd year\Semester II\MT325>
```

Figure 13: Output Result



### 3.1.2. Analysis using R

```
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.2
## Warning: package 'ggplot2' was built under R version 4.3.2

## — Attaching core tidyverse packages ————— tidyverse
2.0.0 —
## ✓ dplyr      1.1.3      ✓ readr      2.1.4
## ✓ forcats   1.0.0      ✓ stringr    1.5.0
## ✓ ggplot2    3.4.4      ✓ tibble     3.2.1
## ✓ lubridate  1.9.2      ✓ tidyr      1.3.0
## ✓ purrr     1.0.2
## — Conflicts —————
tidyverse_conflicts() —
## ✗ dplyr::filter() masks stats::filter()
## ✗ dplyr::lag()     masks stats::lag()
## ⓘ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors

library(tinytex)

## Warning: package 'tinytex' was built under R version 4.3.2

library(ggplot2)

data1 <- read_csv("../Data/before_washing.csv")

## Rows: 29 Columns: 3
## — Column specification
##
## Delimiter: ","
## chr (2): Dot 1, Dot 2
## dbl (1): Before Distance
##
## ⓘ Use `spec()` to retrieve the full column specification for this data.
## ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.

data2 <- read_csv("../Data/after_washing.csv")

## Delimiter: ","
## chr (2): Dot 1, Dot 2
## dbl (1): After Distance
##
## ⓘ Use `spec()` to retrieve the full column specification for this data.
## ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.

data <- merge(data1, data2, all = TRUE)

width <- data[c(5,12,18),]

d1 <- mean(width$`Before Distance`)
d2 <- mean(width$`After Distance`)

width_percentage <- abs(d1-d2)*100/d1
sprintf("width_percentage: %s", width_percentage)
## [1] "width_percentage: 6.12707239213264"

length <- data[c(2,19,27),]

d3 <- mean(length$`Before Distance`)
d4 <- mean(length$`After Distance`)

length_percentage <- abs(d3-d4)*100/d3
sprintf("length_percentage: %s", length_percentage)
## [1] "length_percentage: 0.0205888408482556"

rol_no <- 1

rol_1 <- data.frame(rol_no,width_percentage,length_percentage)
view(rol_1)
```

The above R codes performed the following task

1. Loading the required packages
2. Reading data

There are two CSV files read into data frames called data1 and data2, "before washing.csv" and "after washing.csv," respectively

Before washing data set

	Dot 1	Dot 2	Distance
1	Dot 1	Dot 2	1619
2	Dot 1	Dot 3	3238
3	Dot 1	Dot 4	1619
4	Dot 1	Dot 5	3619
5	Dot 1	Dot 6	3237
6	Dot 1	Dot 7	3619.2
7	Dot 1	Dot 8	4575
8	Dot 2	Dot 3	1619
9	Dot 2	Dot 4	2282
10	Dot 2	Dot 5	2289.6
11	Dot 2	Dot 6	3615
12	Dot 2	Dot 7	3237
13	Dot 2	Dot 8	3617
14	Dot 3	Dot 4	3620.1
15	Dot 3	Dot 5	1619
16	Dot 3	Dot 6	4575
17	Dot 3	Dot 7	3619
18	Dot 3	Dot 8	3237
19	Dot 4	Dot 5	3238
20	Dot 4	Dot 6	1618
21	Dot 4	Dot 7	2284
22	Dot 4	Dot 8	3619.74
23	Dot 5	Dot 6	3616
24	Dot 5	Dot 7	2288
25	Dot 5	Dot 8	1618
26	Dot 6	Dot 7	1619
27	Dot 6	Dot 8	3238
28	Dot 7	Dot 8	1619
29	Dot 8	Dot 8	1619

After washing data set

	Dot 1	Dot 2	Distance
1	Dot 1	Dot 2	1619
2	Dot 1	Dot 3	3238
3	Dot 1	Dot 4	1619
4	Dot 1	Dot 5	3619
5	Dot 1	Dot 6	3237
6	Dot 1	Dot 7	3619.2
7	Dot 1	Dot 8	4575
8	Dot 2	Dot 3	1619
9	Dot 2	Dot 4	2282
10	Dot 2	Dot 5	2289.6
11	Dot 2	Dot 6	3615
12	Dot 2	Dot 7	3237
13	Dot 2	Dot 8	3617
14	Dot 3	Dot 4	3620.1
15	Dot 3	Dot 5	1619
16	Dot 3	Dot 6	4575
17	Dot 3	Dot 7	3619
18	Dot 3	Dot 8	3237
19	Dot 4	Dot 5	3238
20	Dot 4	Dot 6	1618
21	Dot 4	Dot 7	2284
22	Dot 4	Dot 8	3619.74
23	Dot 5	Dot 6	3616
24	Dot 5	Dot 7	2288
25	Dot 5	Dot 8	1618
26	Dot 6	Dot 7	1619
27	Dot 6	Dot 8	3238
28	Dot 7	Dot 8	1619
29	Dot 8	Dot 8	1619

The three columns in each dataset are "Dot 1," "Dot 2" (both character types), and either "Before Distance" or "After Distance" (numeric type).

### 3. Merging the data

Based on shared columns, the merge function is used to combine data1 and data2, and the outcome is saved in a new data frame named data.

### 4. Calculating width percentage

A different subset of the data is taken out and put into a data frame named length that contains the particular rows (2, 19, 27).

The length columns labeled "Before Distance" and "After Distance" are averaged (d3 and d4).

The variable is computed and contains the percentage change in length.

### 5. Creating a summary data frame

The outcomes are kept in a data frame called rol\_1.

It contains the length percentage (length\_percentage), width percentage (width\_percentage), and roll number (rol\_no).

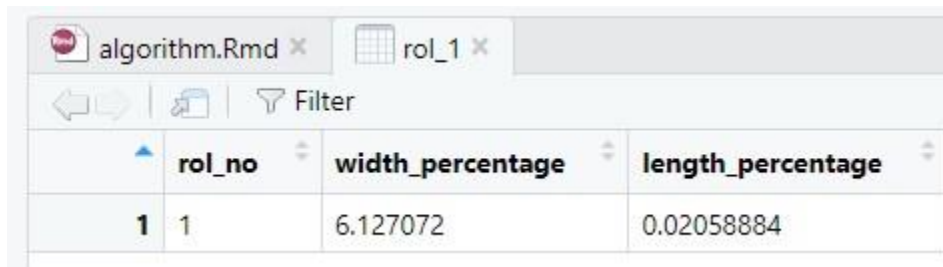
### 6. Viewing the summary data frame

To see what's in the rol\_1 data frame, the view() function is invoked.

### 7. Outputting result

“Sprintf” is used to print the final width and length percentage changes as a string.

We can get output like the one below.



	rol_no	width_percentage	length_percentage
1	1	6.127072	0.02058884

Now we have shrinkage percentages. If the percentages are less than 5% for the roll we accept and if the percentages are greater than 5% we reject the roll.

### 3.1.3. Discussion

In this study, we employed Python programming to preprocess and analyze data related to roll shrinkage in a manufacturing process. Initially, we implemented a Python code to calculate and extract distance measurements between each pair of points, representing specific attributes of the manufacturing rolls. The data were then organized into two distinct Excel files, one capturing the state before washing and the other after washing. These files served as crucial datasets for our subsequent



analysis. Leveraging the power of the R programming language, we conducted a comprehensive examination of the Excel file data. Our analysis focused on deriving valuable insights into roll shrinkage percentages, a key parameter in the manufacturing process. By comparing the data before and after washing, we were able to identify patterns and trends that contributed to the overall understanding of roll behavior. The combination of Python for data preprocessing and R for in-depth analysis proved to be a robust approach, enabling us to draw meaningful conclusions regarding roll shrinkage and contribute to the optimization of the manufacturing process.

# CHAPTER 04

## 4.1. Conclusion

### 4.1.1. Achievements

- **Successful Image Processing Implementation:**

Our project has demonstrated the successful implementation of image processing techniques using Python to analyze fabric images before and after washing. The identification of marked dots was achieved with good accuracy, showcasing the usefulness of our algorithms in handling real-world variations in fabric patterns and textures.

- **Shrinkage Percentage Calculation:**

Through careful analysis of the marked dots' displacement, our Python code accurately calculates fabric shrinkage percentages. This precision is crucial for industries where even slight deviations can impact the quality and dimensions of the final product. The ability to provide reliable shrinkage data adds significant value to manufacturing processes.

- **Excel Integration for Data Management:**

The integration of Excel for output data management streamlines the accessibility and usability of our results. The final shrinkage percentages are neatly organized in an Excel sheet, facilitating easy sharing, interpretation, and integration into existing production workflows. This user-friendly output format enhances the practicality of our solution.

- **Industry Adoption:**

The successful execution of our project highlights its potential for practical application within the textile industry. By identifying the issues of fabric shrinkage, our Python-based image processing solution has the potential to enhance product quality, reduce production costs, and contribute to overall process optimization.

### 4.1.2. Difficulties

- **Image Variability and Noise:**

One of the primary challenges encountered was the variability in fabric images and the presence of noise. Different fabric textures, colors, and patterns in the fabric made harder of dot identification difficult. However, we have identified how to solve the difficulty but ongoing improvements may be needed to further enhance adaptability.

- **Dot Identification Accuracy:**

In the situations of overlapping dots or faint markings, iterative refinement of our image processing algorithms. Also, the marker color that if a fabric is the same color as the marker came, So after facing the challenges we have improved the code.

- **Data Annotation Challenges:**

The accuracy of our shrinkage calculations heavily relied on precise data annotation, specifically marking and tracking dots in the fabric images. Manual annotation processes introduced the potential for human error, and implementing automated or semi-automated annotation methods could further improve the reliability of our results.

- **Equipment:**

Finding equipment for the apparatus was a bit challenging because the image needed to be of good quality for the identification and analysis of the dots.

Despite these challenges, our project has successfully overcome numerous obstacles to deliver a better solution for fabric shrinkage optimization, and ongoing efforts in research and development will further refine and advance the applicability of our image-processing approach in real-world scenarios.

## 4.2. Research Findings

### SmartShrink Shrinkage Rate Tester



SmartShrink Shrinkage Rate Testing device determines shrinkage rate test results in an average of 5 seconds for fabrics after steaming, washing, and dry-cleaning according to the standards of ISO 3759, ISO 5077, and other testing methods.

SmartShrink Shrinkage Rate Tester takes a picture of a fabric sample by uses a camera equipped on the top and it automatically measures the distances between the marked dots and calculates the shrinkage percentage results by the patented vision inspection algorithm. The shrinkage test result of fabric will

be real-time shared with the IoT-connected SmarTexLab app installed on the computer or smartphone.

SmartShrink Shrinkage Rate Testing Machine automatically measures fabric shrinkage and calculates the test results and percentage of the shrinkage and it is designed to avoid the manual errors thus making the test more accurate and reliable for the company; It automatically saves the analyzed data and sample photos, and shares the test results in real-time, making the test more efficient, trustworthy and user friendly; The whole process for the test can be reduced from 6 minutes to only 5 seconds, making the test more rapid and reducing the cost by more than 90% thus increasing the efficiency and reducing the human error (ordnur, n.d.).

### 4.3. References

ordnur, n.d. *The Future of Checking Fabric Shrinkage: A New Way to Test*. [Online]  
Available at: <https://ordnur.com/textile/checking-fabric-shrinkage/#>

Sari-Saraf, H. & Hequet, Eric & Abidi, Nouredine & Dai, Y. & Chan, H.Y. (2002). Automatic Measurement of Fabric Shrinkage. American Association of Textile Chemists and Colorists Review. 2. 20-23.

GeeksforGeeks. (2021). *How to capture a image from webcam in Python?* [online] Available at: <https://www.geeksforgeeks.org/how-to-capture-a-image-from-webcam-in-python/>.