Real Time Vision Based Fabric Defects Detection System for Textile Industries

Author

Umarah Qaseem

Regn Number

204829

Supervisor

Dr Hasan Sajid

DEPARTMENT RIME

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

January, 2020

Real Time Vision Based Fabric Defects Detection System for Textile Industries

Author

Umarah Qaseem

Regn Number

204829

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Robotics and Intelligent Machine Engineering

Thesis Supervisor:

Dr. Hasan Sajid

Thesis Supervisor's Signature: [illegible]

DEPARTMENT RIME

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,

ISLAMABAD

JULY, 2020

Declaration

I certify that this research work titled "*Real Time Vision Based Fabric Defects Detection System for Textile Industries"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

[illegible]

Signature of Student

Umarah Qaseem

2017-NUST-MS-RIME-204829

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

[illegible]

Signature of Student

Umarah Qaseem

Registration Number

204829

[illegible]

Signature of Supervisor

Copyright Statement

Acknowledgements

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every department of my life. I am particularly thankful to my mom for motivating me and dad for pushing me to complete my thesis.

I would also like to express special thanks to my supervisor Dr. Hasan Sajid for his uplifting support throughout my thesis technically as well as morally, and also for Deep Learning, Computer Vision, Machine Learning, Mechanics and Robotics-I courses which he has taught me. He is one of the best teachers I have ever had. I can safely say that I have not learned any other engineering subject in such depth than the ones which he has taught.

I would like to express my special thanks to Dr. Talha Naqash for his moral support and useful advices in managing office work along with my thesis and for my career.

I would also like to pay special thanks to my brother Ammar and my friend Saad for their tremendous support and cooperation. They kept sending me research papers patiently in Quarantine when I was not able to access them from my home internet. Each time I got stuck in something, they supported me emotionally and technically. They also let me use their high-end devices and systems online for testing purposes from Georgia, USA and Germany. Without their help and patience I would not have been able to complete my thesis.

I would also like to thank Dr. Yasar Ayaz and Dr Jawad Khan for being on my thesis guidance and evaluation committee and for teaching me AI, Robotics-II and Mobile Robotics courses. They are indeed the best and finest teachers of their field. I would further like to express my special thanks to Dr. Jawad Khan for his help and guidance while submitting thesis documents.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my exceptional parents and adored siblings whose tremendous support and cooperation led me to this wonderful accomplishment.

Abstract

Quality control of textile fabrics is a crucial problem for textile Industries. The inspection of fabric is done manually by experts which is a hectic and eye straining procedure. The conveyor belt of fabric machine is around 3m wide and moves with a speed of around 25 to 200 m/min. Humans cannot detect more than 60% defects and there is chance of error even at the speed of 25m/min. Therefore, an automated Intelligent system is required efficient both in terms of precision and speed, for Inspection of fabrics. The system must have both high accuracy and good frame rate. Current systems have either high Accuracy or high FPS (less processing time) since there is a trade-off between both. Machine Learning is computationally and economically very expensive to detect and mark faults on the images. Several existing vison-based systems are also either specific to certain type of defects or dependent on color of fabric, such systems have either low accuracy or low FPS, hence yielding poor results or very expensive for small companies.

We have designed a very simple yet very fast system with trivial pixel processing algorithm, which is optimized, is far more accurate than trivially running systems and inexpensive approach towards the resolution of such problems. Through our image acquisition system, we have acquired very High Definition Images and have processed them by our implemented system at very high frame rate on trivial personal computers. The approach works on all colors and for all types of defects with the same accuracy, gives coordinates of the fault to the robotic arm which marks the faults/cuts the defective area/raises alarm etc. A data set of 107000 images is used having dimensions of 1920x1080. Experiments yielded the results of over 93% on images and up to 100 % on video.

Key Words: Fabric Defect Detection, Pixel Processing, Textile Industry, Computer Vision, Textile Inspection System, Image Processing

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

With the up rise in Image processing and machine vision, many manual processes are automated. A large number of products are made from textile including clothes, wipes, bags, furniture and many other transportation and household items. So, the presence of defects in that fabric manufactured in textile industries causes the companies huge losses, if those defective pieces are not removed. There is a lot of competition and they must keep the quality under check at all times. According to A. Kumar the prices of fabrics are reduced up to 45% - 65% due to presence of these defects. [1] The quality assurance department of textile companies need to keep the quality under control and remove the defected areas in the bolts of fabrics. The detection and removal are usually done at the last stage of fabric production. For the inspection of fabric, manual procedure is followed, where human experts check for the defective area and mark it. This a hectic and an eye straining procedure which results in some defects being skipped and not marked. According to Che-Seung Cho, even in best cases the inspectors cannot detect more than 60% of the defects. Also, humans cannot deal with fabric moving faster than 30 meter/minute or wider than 2 meters. [2] Therefore, the implementation of an automated intelligent machine vison system for fabric defect detection is crucial for textile industries. Such a system will reduce the labor cost and increase the efficiency of the quality control procedure.

The fabric machine has a conveyor belt on which cloth is moving constantly. The cloth has to be inspected while it is moving. The width of the belt and hence the width of fabric, varies from 1 to 3 meters. The speed of the belt varies from 20 to 200 meters per minute. Therefore, the developed automated system should be efficient in terms of both accuracy and speed. By high accuracy, we mean that it should not mark wrong defects as clean fabric will be wasted in that case and it should not skip defects as that will result in bad fabric quality of the company. By speed we mean that the processing time for the defect detection should be minimum. If processing time is less, we will have a higher Frame rate or Frames per second. We will use the term Frames per second as FPS in our paper. Since the conveyor belt is moving with the above-mentioned speeds, it must be taken into account and a high FPS is needed. In addition, since humans can not deal with speed higher that 30 meter/min, if the automated inspection system has really high FPS, the speed of the conveyor belt can be increased beyond 30 meter/min and increase the quantity of work done in less time in the textile industry.

To implement a machine-vision based fabric detection system with a robotic manipulator, a camera is mounted on top of conveyor belt and video stream is captured. From that video stream, video frames are extracted which is basically an image. So, the input of algorithm is an image or matrix of pixels. That image is processed through the detection algorithm. The algorithm has two tasks, one to find if a defect exists and other to find the location if it exists. The detection algorithm gives the result that the defect is present or not and if present, then location is found. So, the output is coordinates in case of presence of defects and nothing in case of clean cloth. If defect is present coordinates can be passed to the robotic arm and the defected area can be cut/marked etc.

Some people have used machine learning algorithms for the solution of this approach. When using machine learning for defect detection object detection models/networks are used. Object detection networks comprise of classification + localization. (i) These models are computationally very expensive. Even when an image is passed through a simple classification Convolutional neural network it takes time, so when object detection is performed the processing time increases more, obviously. (ii) Also, we need heavy GPUs to train the system, that is cost expensive. (iii) Also, to implement the system or Robot we need a heavy card to run the system on. It cannot simply run on raspberry pi or any lighter board. (iv) In addition, another disadvantage is that the size of image has to be decreased. High definition images are not passed through CNNs as they are not feasible. So, decreasing the size results in information loss which can result in some defects being missed and hence less accuracy. (v) Another issue is that machine learning systems only learn the types of defects, and types of clothes on which it has been trained on. If a new type of defect come for which it was not trained it may not recognize it as a defect or if a new kind or texture of fabric comes in to the picture, once again it may not identify the defect and accuracy will decrease. So, to deploy this system in the industry it has to be trained on large amount of data from that industry for it to give best results. Therefore, a dataset from that factory or fabricator unit has to be collected first for some months at least. This is doable but it is still a hectic procedure compared to the other approach i.e. pixel processing approach which we have done. In that approach we also required and collected dataset once for the testing of our system, but that was one time. To deploy our system in any factory it does not require dataset again from that factory to work best. Our algorithm can work for any types of defect, on any type of fabric and any color of un-patterned fabrics.

Some people have used image or pixel processing approaches before. This approach usually is light and fast obviously as it does not require the heavy calculations of deep neural networks. While they are fast, they do not always have high accuracy. Also, these algorithms have to process all pixel in an image, so the existing approaches do not work on high definition images or videos as they again become computationally expensive. In addition, these algorithms do not always work for all types of defects, all colors or all types of fabrics. Although this approach is fast but existing algorithms are still not fast enough to have high FPS for a trivial system to be deployed in a textile factory.

There are variety of defects that occur due to dying, needle, thread, hole, grease marks etc. The most common of them are holes in cloth of varying sizes. The existing approaches are specific to the type of defects or they can cater only some types of defects. While they work good for some, they fail when the type of defect changes.

In conclusion, the efficiency of the automated defect detection system depends mainly on the algorithm used for detection. This means the algorithm should be efficient, accurate and computationally inexpensive. This task of defect detection can be further broken down into two sub tasks. The first one is finding out if there is a defect or not. The second one is, finding the location of defect i.e. the coordinates. Many attempts have been made for this before, but they have either good (high) accuracy or good (low) processing time but not the both. There is always a trade off in these two things, since complex algorithms will take more time, but they will be more accurate. Simpler algorithms will have less processing time, but they will not be as accurate. We have proposed a very simple approach which has both high FPS and high accuracy. This is proven through rigorous testing on a huge dataset as well as testing on a public dataset to evaluate our approach by comparing results with another approach. We have also evaluated our approach by comparing the results of another technique after applying it on our dataset. So, the comparison is done between algorithms by keeping a common entity between them i.e. the datasets.

NOTE Accuracy and FPS are inversely proportional and there is always some tradeoff between them.

Figure 1: Quality Inspection Fabric machine [3]

CHAPTER 2: LITERATURE REVIEW

In this chapter we will discuss previous work done in this area. This chapter is divided into two parts. First we will discuss the textile databases that have been used in the automation of inspection systems, then we will discuss different methodologies and approaches previously proposed in this area along with their gaps and shortcomings.

2.1 Textile Databases

Textile defect detection is an active field right now and many researchers have worked and are working in this area. As described in detail in the previous section that many attempts have been made in this area before, but they have either good (high) accuracy or good (low) processing time but not the both. Also, the resolution of images used for testing or for which the systems are designed are very low. Low resolution of reduction in image size always result in information losses, which could otherwise have been used for detection more accurately. Even if we argue that in some cases information loss is not a lot, but there is at least some information loss in every size reduction case, always and that is obviously not good. In many papers, the accuracy of proposed defect detection approaches is very good, but the results are provided by testing on very few images, in some cases even two images.

For performing proper and extensive research in the field of textile defect detection which can be used in developing a system in real time, first we need to have a proper set of samples with and without defects. This means to have a proper database of defects on which different proposed algorithms and methods can be tested and evaluated. Many works that have been done before are evaluated on few images. Since the set of images is too small it is a clear issue that those results cannot be considered acceptable to have a general application [6]. In case they are evaluated on sufficient number of images, even then they are tested on different sets of images so the results cannot be compared directly. Even the resolution of images is not fixed. Resolution of images directly affect the accuracy and speed of algorithm positively and negatively respectively, in general. Different algorithms in papers have shown results on different resolution of images. In many papers there is little or no information about resolution and other properties of images/camera. This makes it difficult to verify and compare results of new methods with the old ones, to show improved results as comparison of results cannot be made if they are tested in separate situations or datasets. [4,5,6]

There are many databases available which are not specific for textile but are used for analysis of more generic aspects. Many researchers have used them. For example, there is a database published on Berkeley Computer Vision Group website [7] and it is used by [8] for the contour detection problem. There are some well-known texture datasets too including Brodatz [9] and VisTex [10]. Many papers have used these in their research including [11,12,13]. CURet [14] and KTH-TIPS [15] are also present but they are not very well known. Another paper [16] has used dataset of patterned fabric which was provided by University of Hong Kong. Although that dataset is private. This dataset has 106 samples. 50 of them are defect-free, and 56 defected as used in Ref. [17]. Another paper has used this same dataset and it mentions 25 textile images in it [18], another using this same dataset mentions 30 defected samples while 30 defect-free samples [19]. Hence the number of images in this dataset are different in different papers as it must have grown over time.

No doubt texture analysis is relevant in textile research but we have not considered these works in this paper as they are not focused in the work of textiles and fabrics, some of them are quite old and also they do not have images of defected samples in many cases.

In this paragraph, we will discuss and summarize some textile datasets. They are used by many authors in different previous papers.

One of the most famous datasets used by many research publications in the fabric defect detection field is TILDA [20]. It is the acronym for Textile Texture-Database. It was developed in 1995 by the Technische Universität Hamburg. It was developed within the framework of the working group Texture Analysis of the DFG`s (Deutsche Forschungsgemeinschaft) major research programme "Automatic Visual Inspection of Technical Objects". There are eight representative textile types in this dataset. There are 8 sorts of classes for each textile kind and for each of the classes, 50 TIF pictures with resolution 768 x 512 pixels, and 8 bit gray-level image were acquired through relocation and rotation of the textile sample. The entire texture textile database consists of 3200 TIF pictures with a total size of 1.2 Giga byte. However, this database is not easy to access as researchers have to pay to use it [4].

Another dataset is known as PARVIS [21]. It does not have public access and is private. There are two kinds of textile types in it with 1117 elements.

2.2 Detection Methodologies

In this section, first we will briefly enlist the names of many techniques and methods which have been used or are popular in the field of textile defect detection in past papers, then afterwards we will discuss the latest and most common defect detection methodologies and their comparisons in terms of accuracy, datasets, number of images used for testing, resolution of images used, and speed of algorithm.

2.2.1 Brief Discussion of Techniques

There are many methods and algorithms present for the solution of textile defects detection problem. Some algorithms use edge detection for this purpose. Many papers have used different approaches for edge detection including the use first order derivative or gradient, second order derivative with autocorrelation function e.g. [22,23], colored edge detection, gaussian edge detection, zero crossing, and techniques based on mathematical morphology focused mainly on geometrical structure and topology of objects e.g. [24]

Some researchers have used fractal method for textile detection too [25]. Another technique works by assigning one pixel to a region based on local features of the image in that pixel and its immediate neighbors. This technique, also known as, feature based technique, includes grey level segmentation methods for detection of defects in textile e.g. [26]. Another feature-based technique is scale invariant feature transform (SIFT) and speeded-up robust features (SURF) which is based on SIFT [27]. Another defect detection technique which is based on segmentation of image in color is the use of principal component analysis (PCA) and Gabor filters. [28] Defect detection was also carried out using analysis of color channels by Xiaobo and Jinlian. For this first they segmented the jacquard image in colored channels then to characterize the defined defect they performed pattern comparison using Fourier transform and frequency spectrum analysis. [29]

Other techniques used which work in frequency domain for detection are Fourier [30,31], Gabor [28,32], Wavelet transform (WT) [33,34], wavelet packets [35], multiple adaptive wavelets [36], Gabor wavelet networks [37], Gabor WT [38], Mallat Wavelet Transform [39] etc. Also, some people have used methodologies which are based on optimized filters, e.g. finite impulse response (FIR) filters [40]. These methodologies are characterized as spectral approaches for defect detection by [1] and [3].

Many researchers have used machine learning for this problem. It is considered as a classification problem for one class and object detection for localization of defect. Several machine learning approaches have been proposed which use Artificial Neural networks (ANN) Ref [42 to 53], Support Vector Machines (SVM) [55] and Pulse coupled Neural networks (PCNN) [56, 57] etc. Although simple ANNs are expected to be most widely in use in this field according to [4]. Support vector data description (SVDD) which is a one class classifier, is also used as a detector in some researches e.g. [58, 59]. Convolution neural networks are gaining a lot of attention in this field too Ref [63 to 73]. Classification of dry-washed fabric as defected or defect-free using CNN and SURF was done in [60]. Deep Neural Networks are also very famous for textile defect detection and used in many new researches e.g. in [74,75]. The use of autoencoders for fabric defect detection can also be seen in the work of some researchers with a Multi-Scale Convolutional Denoising Autoencoder Network Model [41]. Some researchers have also used genetic algorithms along with neural networks for detecting defects in fabrics [54]. Faster-RCNN is also used for textile defect detection [62] by first generating proposal regions through Region proposal Network (RPN) before applying faster-RCNN by [61].

2.2.2 Comprehensive Comparisons of Techniques.

Below is the comprehensive discussion on the defect detection methodologies. The key aspects of comparison are accuracy, datasets, testing methodologies, number of images used for testing, resolution of images used, and speed of algorithm. None of the publications discussed in this section are older than 5 years i.e. all the work is recent and is of the year 2015 or onwards.

In 2019, Sun, G., Zhou, Z., and his team has worked on defect detection in fabrics.They have used classification and histogram back projection for it in their paper. They have published a good accuracy of 96.12 % but the dataset they have used consists only of 1000 images, among which 500 are from TILDA dataset while remaining 500 were from their own dataset. Out of 1000 images, they used 800 for training their network and only 200 were used for testing. Also, the resolution they have used is very low which is 200 by 200 pixels of one image. Their results are not good in terms of processing time on a frame to find a defect. Some frames take up to 7 seconds. Although their average processing time of a single frame is 0.72 seconds, but it is to be noticed that this speed is on a low 200x200 resolution. Also 0.72s on one frame means (1/0.72) 1.38 FPS which is not very good either for real time. Since this is a machine learning approach, this processing time is impressive too and they have done some good work in this field. [76]

A method was proposed for fabric defect detection using local homogeneity analysis and neural network by Rebhi and his team in 2015. Their accuracy was 96.25 % but the dataset they used only consisted of 89 images out of which 76 had defects and remaining 13 did not have defects. Their dataset is provided by a partner textile industry in Tunisia, they have given this link for it. [77] They used a very low resolution of 256 by 256 pixels. They used only 54 (60 percent) of those 89 images for training while 35 (40 percent) were used for testing which is obviously not a good number to train a neural network. In their paper, they have mentioned the processing speed as 0.8 seconds for defect detection by scanning one image of 256 by 256 pixels, which is too slow with such low resolution as it corresponds to the FPS of 1.25 (1/0.8 please refer to formula (9) in section VII). [42]

The same group published another paper in the following year (i.e. 2016) with a slightly different approach for the same problem. Their paper was titled fabric defect detection using local homogeneity and morphological processing. They used image processing approach instead of machine learning approach which they used previously. This time their accuracy was 95.6 %. Image resolution is same as their previous paper i.e. 256 x 256 pixels. All their results were calculated using MATLAB simulation. Total 90 images were used for testing. They have not mentioned the processing speed for this approach. [78]

Yundong Li and Cheng Zhang worked on an automated vision system for fabric defect inspection using Gabor filters and PCNN in 2016. They have only used two images and their accuracy is 98.6%. They have not mentioned any other dataset in their paper. All the comparisons and tests are performed only on those two images, although they have performed 20 iterations of their model on each of those two images and compared results of detection on those iterations. A processing speed of 5 FPS (frames per second) is mentioned. [79]

In 2018, Hong-wei Zhang, Ling-jie Zhang and their team published their work of Yarn-dyed Fabric Defect Detection with YOLOV2 Based on Deep Convolution Neural Networks. They acquired their dataset from an enterprise. The total images in their database (training + validation + testing) are only 276. All of them are defect samples. There are no defect free samples. All these images are of yarn dyed fabric. The resolution or size of images is not mentioned anywhere. They have used YOLO9000, Tiny-YOLO and YOLO-VOC network model structures. The first two among these mode structures showed very poor results (0% average recall and precision by YOLO9000 and 6%, 36% average recall and precision respectively by Tiny-YOLO), so they selected YOLO-VOC whose results were better. They further changed the learning rate, iterations, and structure of this model two times and published results for that. They have not mentioned accuracy anywhere in their paper, but they have published average recall as 88.24% and average precision as 86.83% for YOLO-VOC which are not very good. Their processing time is 0.023 seconds which is quite good as it gives 42 FPS but still less than us, but it should be kept in mind that this time is on GPU not CPU. (They have used Intel i7-5930 with four NVIDIA GeForce TitanX 12G and eight 8G memory). Furthermore, although this processing time is good, but we do not know what size of image this processing time is for, because size of image drastically effects the processing time (directly proportional). Also, they have not used any negative samples in their testing which make results biased towards positive samples only. Lastly, this method will only work for yarn dyed cloth. [80]

In 2017, Yan, H., Paynabar, K., & Shi, J. worked on anomaly detection in images with smooth background. They used smooth-sparse decomposition method for this purpose which they called as SSD. They have used only two samples. Resolution of one sample image is 200 x 2910 and the other's is 90 x 550. Their approach is not specific to fabrics and these two images are of silicon surface. The processing time for those two samples are 0.350s and 0.034s seconds which means FPS of 2.85 and 29.4, respectively. Accuracy is not mentioned but they have mentioned a false-positive rate of 0.018,

While a false-negative rate of 0.0118. [81]

M. S. Sayed published his work on Robust fabric defect detection algorithm in 2016. He has used entropy filtering, minimum error thresholding and morphological operations for the detection. He has used only 60 images from TILDA database which are too less for real time testing. The number of defected and defect free images are not mentioned in these 60 images. Resolution of TILDA database is 512 x 768 pixels. Matlab R2012 was used for implementation and evaluation of the algorithm. The mentioned accuracy is: 96.66%. The processing time for the said method is not mentioned. [82]

Last year i.e. in 2019 Peng, Junli and their team worked on textile fabric defect detection based on low-rank representation. They have used low-rank representation based on eigen value decomposition and blocked matrix. Their dataset has 500 images from TILDA dataset. The resolution they have used is 768 x 512. Their accuracy is 89.2 %. Although the accuracy is not too high compared to other papers, but they have done some good work as their resolution is higher than many other related works and their dataset has considerable number of images for proper testing. It should also be noticed that they have divided their experimentation into different parts. Each part has different accuracy from 50% to 100%. The accuracy mentioned above (89.2%) is the one they have written as an overall combined accuracy of all experiments. The average time taken for one sample (image/frame) is 25.38104 seconds which makes (1/25) i.e. 0.04 FPS. [83]

Jielin Jiang and his team has recently worked in this area and published their work in February 2020. They have used a Sobel operator combined with patch statistics algorithm for detecting defects in fabrics. They have used a patch of size 7x7. They have not published any quantitative result in their paper, therefore no accuracy, speed, dataset information or image size is mentioned. They have only published qualitative results (result in form of image output) on only 12 images. Moreover, even in those output images, neither location coordinates of defect are given, nor a box is drawn on the detected defect. The only reason this research work is considered in this paper because it is one of the latest papers as it has been published earlier this year. [84]

Zhoufeng Liu and his team designed a lightweight CNN model for the detection purpose in 2019 and called it DefectNet. They have created their own private dataset which is mentioned in of

3000 greyscale images having a resolution of 512 x 512 pixels. Their dataset is of considerate size for testing. Although they have neither mentioned the number of defect and defect-free images, nor the number of training, validation, and test images. The accuracy and processing speed of algorithm is good compared to other papers, but they have not mentioned any other quantitative results or the method through which they calculated their accuracy. Their accuracy is 97.7 and speed is 0.0327 s. We can calculate FPS from this speed as 30.5 which is good for real time working of algorithm in a factory. It should be kept in mind that this processing speed is acquired by running their algorithm on GPU (NVIDIA Quadro M5000). We have not used GPU at all for our algorithm testing, rather a simple personal laptop is used. Even after using GPU, their algorithm takes more processing time to detect than us i.e. their FPS is lower than us. [85]

Yudi Zhao and her team worked on visual long-short-term memory based integrated CNN model for fabric defect classification, earlier this year i.e. 2020. They have used DHU dataset collected from Donghua University, China. This dataset is further subdivided into two datasets, one having 500 images and the other having 1000 images, although they have neither mentioned the division between training and test data in this dataset, nor the number of defect and defect-free samples. They have also used another dataset in their paper named "Aliyun-FD-10500". This dataset is collected from TianChi competition. [86] The number of images present in this dataset are not mentioned in their paper, nor on the website they have provided i.e. [86]. The resolution of images in every dataset they have used is 224x224 which is quite less. They have divided the Aliyun dataset into training ¾ parts and test set ¼ part. Since they have used Aliyun dataset to choose best hyperparameters, that is why they used the test set of this dataset to evaluate their proposed model. Their accuracy on this dataset is 95.73%. They have provided results of other performance metrics too as their qualitative result, which is good. Their average processing test time on all datasets is 0.026s, that gives a frame rate of 38.5 which is less, even when their image resolution is quite low. Their minimum test time is on Aliyun dataset which is 0.018s and 55.5 frame rate which is still less than our frame rate on i7. Also, it must be noted that they have used GPU, NVIDIA Titan XP, on i7 6800k with 64 GB RAM, while the specifications of our testing systems are very simple (8 GB RAM and without GPU) hence cost effective, which we have mentioned properly in Testing section of this paper. [87]

CHAPTER 3: SYSTEM ARCHITECTURE

The system contained several modules, beginning from video cameras and ending at robotic manipulators via different boards and setups. This section describes the working of the system.

3.1 Hardware Setup of Factory

Bolt of fabric is moving on the textile machine. A system is made which has two iron rods on the base. Their height is according to the height of the textile machine. It has a box to keep the board or processor on which the system will run. To acquire camera stream for collecting data we used the board Jetson tx2. Although the execution of our defect detection algorithm did not require tx2 as the algorithm is very simple. We just used Jetson tx2 to get video stream for dataset collection. From the base of our hardware system, there is another rod which is placed on an angle and it goes straight up where the cloth is moving and a wide strip of the bolt of fabric is visible. We have mounted our camera and light there.

3.2 Fabric Machine Specifications

The specifications of the textile machine are present below in the table 1.

TABLE I
FABRIC MACHINE SPECIFICATIONS

Sr.	Quantity	Measurements
1	Machine Base	12'7 inches
2	Upper back	9 feet
3	Lower back	5'8 inches
4	High back	7'7 inches
5	Slope length	191cm
6	Fabric speed	20-25 meters/min (it varies according to cloth type)
7	Width of Fabric Bolt	92 inches max
8	Camera FOV	69 degrees

Specifications of the fabric Machine on which our system was mounted.
FOV is Frame of Vision.

3.3 System Design

The cameras are installed above the fabricator unit over a height of 16 feet. A special 2000 lumen LED cool day light was fitted along the cameras to provide sufficient light for video capturing. The cameras are then connected to the Texas Instrument board, TX2 which does all the processing on the frame and returns the coordinates of the fault and raises alarm in the form of red LED installed along the board. The manipulator will then be controlled through these coordinates.

3.4 Video Stream Acquisition

Two cameras from Logitech, model name brio 4K and c920 HD pro are installed above the conveyor for video capturing purpose. A pipeline was developed to capture a high definition dataset at frame rates ranging from 30 to 300 fps but typically 60 fps is used for testing and algorithm designing purposes. Videos were made of 10 seconds each, which were used during system design and the same system is now used for fault detection.

The acquired stream was first stored on SSDs attached with the TX2 board.

3.5 Figures

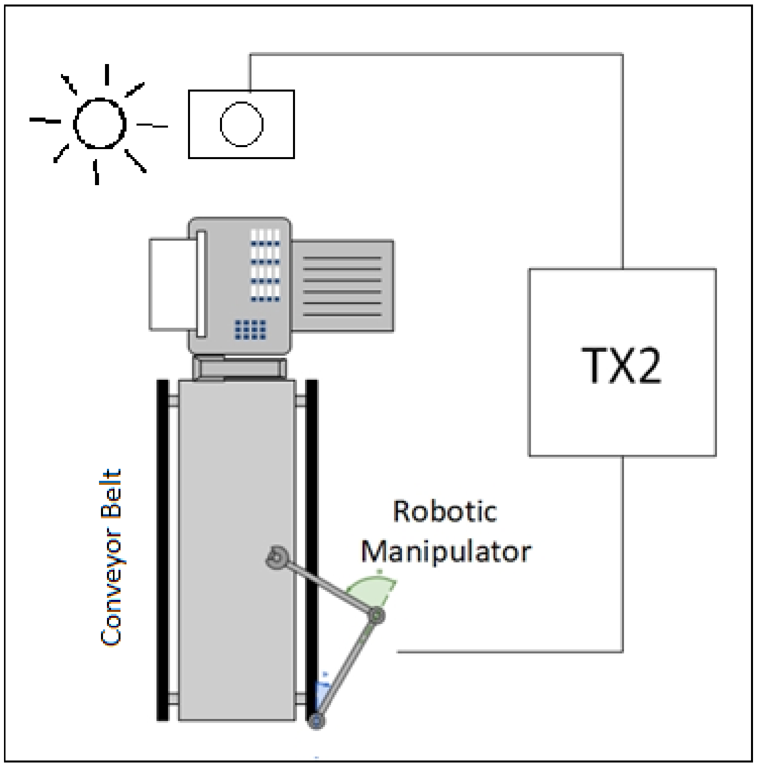


Figure 2: Hardware Setup of factory

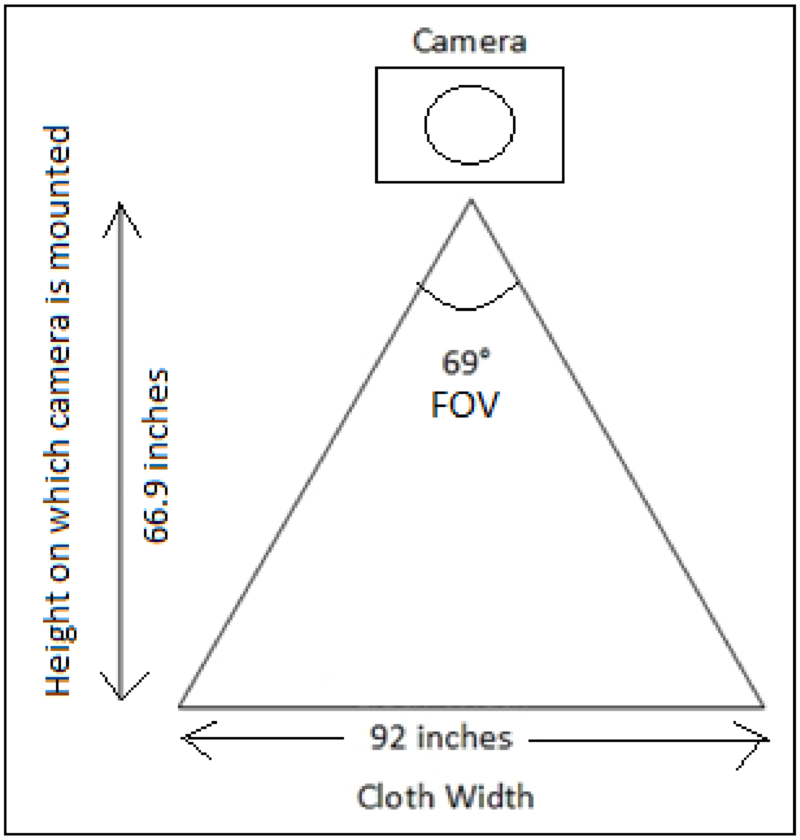


Figure 3: Camera FOV Diagram

CHAPTER 4: IMPLEMENTATION

During implementation, different techniques and methodologies were used and developed on different platforms and languages. Few of the final approaches used and tested are explained in this document.

4.1 Image operations used in our approaches

Few most basic and commonly used pre-image processing operations are applied in our system with little different parameters explained below.

Whenever we use the below described terms in this document, we refer to these specific values of the operations for processing for all approaches.

4.1.1 Grayscale

The hardware setup especially the camera installation over a fabricating unit in the production factory has different lightening conditions than usual white light. This configuration forced us to set different RGB values accordingly than the ones we use for normal image processing operations. These values are (0.24,.69,0.07)

4.1.2 Binarization and Thresholding

Normally, fifty to fifty percent ratio is used in normal image processing operations but our setup used sixty-one to thirty-nine percent due to camera and pipeline configurations.

4.1.3 Erosion

Using the three by three to eleven by eleven structuring element, one approach used in this morphological operation to erode the fault. Whenever this operation is applied, its counter operation is always applied to keep the shape of fault close to the original as maximum as possible.

4.1.4 Dilation

Ranging from three by three to eleven by eleven, but mostly higher sized structuring elements are used in this operation. Less often, only dilation is applied on the frame to maximize the fault area to detect the problem in the fabric.

4.1.5 Opening

This is a morphological operation applied using a structuring element of some square sized window like a 3x3 matrix. The window moves as a slide over an image and according to the structuring element, it alters the values of the resulting pixels. in our case, we used a matrix of 3x3 to 11x11 matrix and applied the opening operation up to 7 times after pre-image processing operations to combine the binarized preprocessed resulting pixels. the problem is that in the high definition images, the texture is so neat that it creates a salt and pepper noise effect after applying gray scaling and black and white operations. this technique is useful in that case to overcome such scenarios. further image processing operations are applied afterwards.

4.1.6 Closing

This is an inverse of the opening operation. Using the same structuring element, it is also applied after applying opening operation because opening operation increases the size of the subjected area that must be decreased before further processing the image, so this operation is applied.

4.1.7 Gabor filters

Gabor filter in digital image processing is a linear filter which analyze the texture of image, or presence of fault in our case. In simple words, it analyzes if there is a particular frequency (defect /anomaly) present in the image in specific directions in a localized region where we are analyzing around a point or region. This is used in the approach with which we have compared our method. Refer to section VIII of this paper.

The generic expression of 1D Gabor filter is as follows as written in [88]

$$\mathrm{G(t)} = \ \mathrm{k}e^{j\theta}\,\mathrm{w(at)s(t)}$$

where,

$$\mathrm{w(t)} = e^{-\pi \mathrm{t2}}$$

$$s(t) = e^{j(2\pi f_0 t)}$$

The generic form of 2D Gabor filter family as written in [89] is

$$G\left(x,y\right) = \exp\left(-\frac{x'^2}{2\sigma_x^2}\right)\exp\left(-\frac{y'^2}{2\sigma_y^2}\right)\cos\left(\frac{2\pi x'^2}{\lambda}+\varphi\right)$$

where,

$$\begin{aligned} x' &= (x-m_x)\cos\gamma-(y-m_y)\sin\gamma \\ y' &= (x-m_x)\sin\gamma-(y-m_y)\cos\gamma, \end{aligned}$$

4.2 Designing of Algorithm – Analysis via high end Techniques

Several techniques were used to design and optimize the algorithm. To perform a thorough analysis of dataset images for the designing of our proposed algorithm, we developed a software application in C#. This app gave the flexibility to show a variety of graphs, variable and multiple thresholding, gray scaling, and Image operation values for state-of-the-art analysis.

4.2.1 Luminosity Analysis

As the fault occurs, the uniformity of the picture gets disturbed and so does the lightness of the picture. The luminosity curve can be used to determine the fault along with other techniques.

Let us take the example of the image in figure no. 4 below.

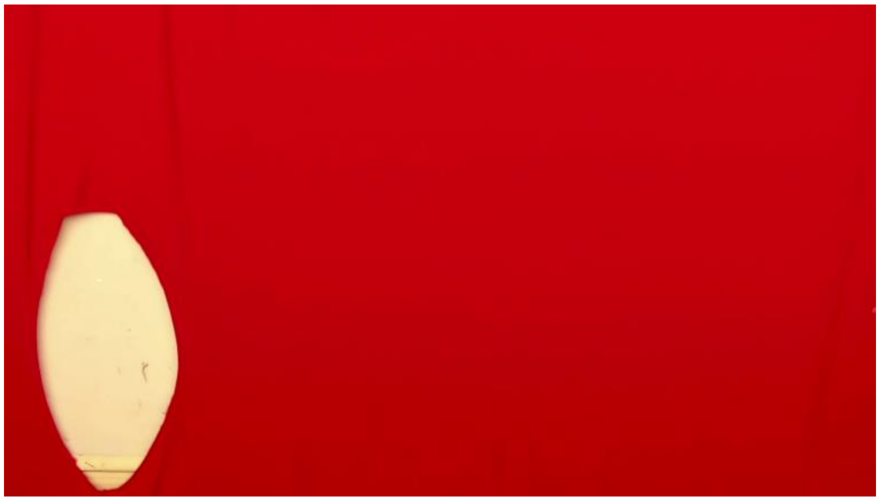


Fig.4. Example of defected Image for analysis

The light is not same as in the defected area. The luminosity curve of the image in figure no. 4 after equalization is shown in the plot of luminosity, between pixels and lightness in figure no. 5.

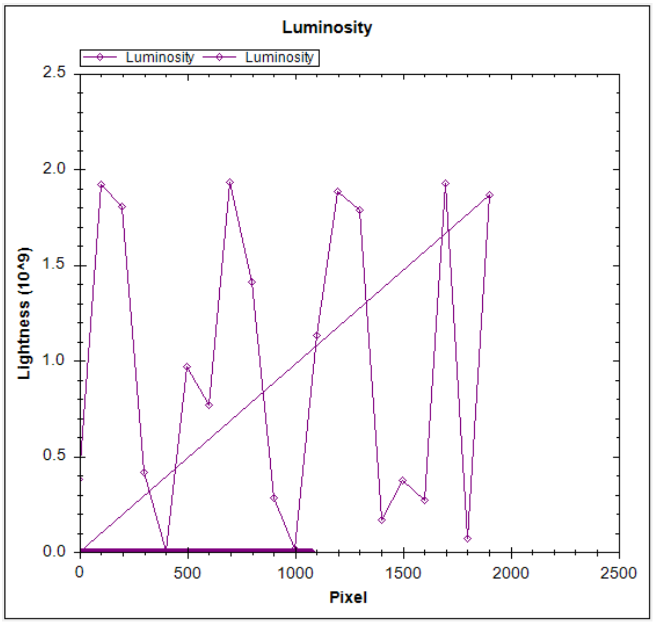


Fig.5. Plot of Luminosity between Pixels and Lightness

Firstly, it was all along zero, the cluster can be seen in the bottom left side of the graph, then it raised to some error values which can be seen as higher marks in the plot.

4.2.2 Histogram Analysis

Taking the example of the same image in figure no. 4. Let us plot the histogram of this image.

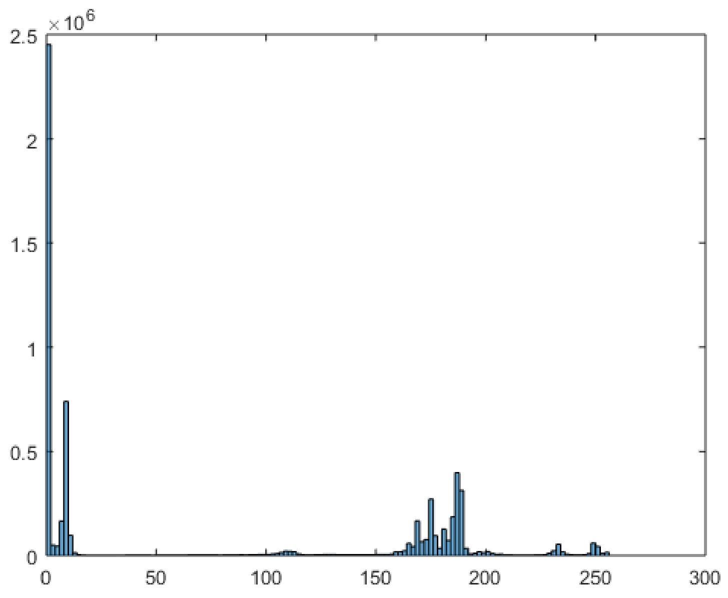


Fig.6. Histogram of image in figure no. 4

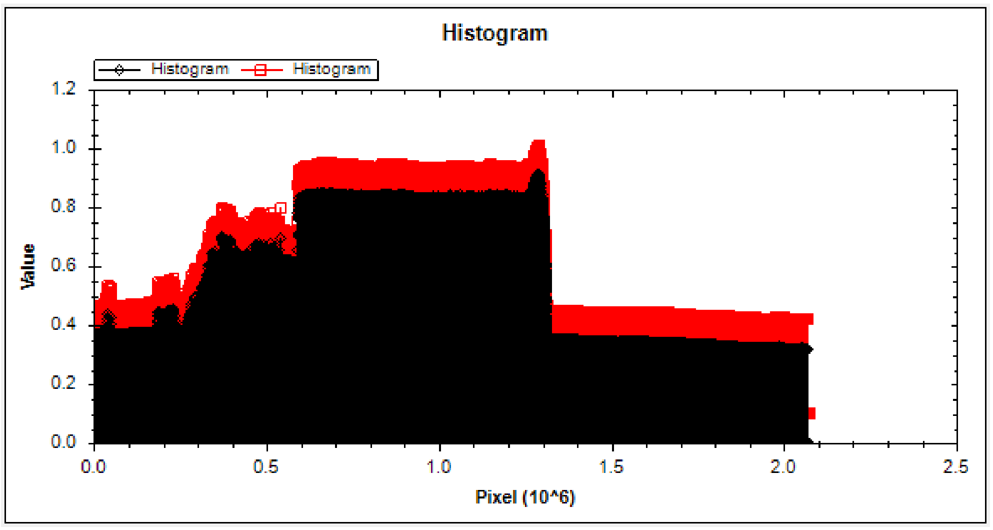


Fig.7. Histogram of image in figure no. 4

4.2.3 Noise Analysis

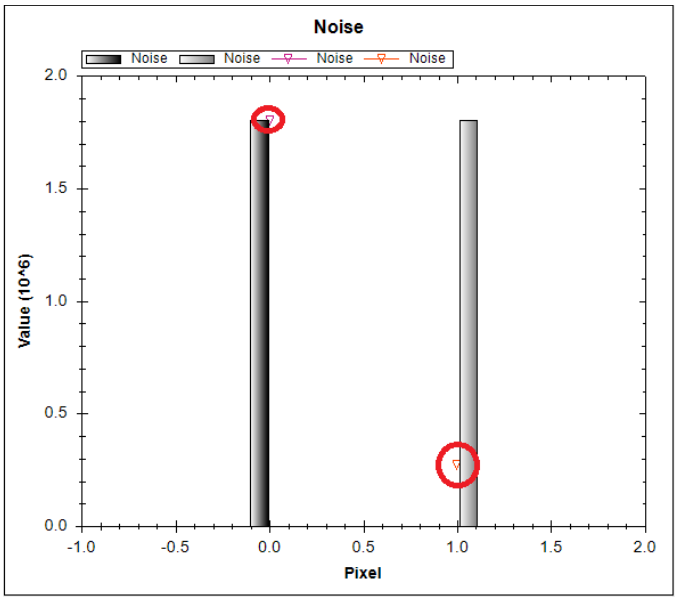


Fig.8. Noise of image in figure no. 5

4.3 Development of App

As explained in previous section, we developed a windows software application for analysis to design our algorithm. This said application was made in C#.NET using EmguCV.NET. Interface of that said application is shown in figure no. 25.

4.4 Fault Detection Solution 1

This approach was developed in C# using Aforge.NET and EmguCV.NET. Due to the limitation of the libraries and processors, the frame handling and processing time was about 10 to 12 seconds which was far more than the required.

In this approach, only the image dataset was used. Image was taken from the dataset stored on the local Solid-State Drive, Gray scaled, using above defined parameters, binarized, eroded by 3x3 structuring element and then evaluated. The iterator ran and processed each pixel which consumed too much time and yielded poor performance but good accuracy. The accuracy of this system was tested only on 296 images which was over 80 percent but stopped due to very slow processing time.

4.5 Fault Detection Solution 2

This idea was implemented in python using complex mathematical formulae and two libraries namely OpenCV and NumPy. OpenCV was used for basic image processing operations. All other system was designed from scratch.

It loaded the video from local drive and then started frame by frame processing. The output of the implementation are the coordinates of the faults which are to be fed to the robotic manipulator to mark the defected area on the fabric.

The system took the video from the drive and applied operations on each frame in series. If the fault prevailed for 5 consecutive frames, the coordinates are given to the next system. The pattern of fabricator is that the cloth travels from top to the bottom and so does the fault. This technique also makes sure that if the problem is too small or too large, it does not affect the detection mechanism as it travels from top to bottom, when one third of the frame is marked faulty, the computational algorithm only then raises the alarm and marks it faulty.

4.5.1 Pixel Operations

This methodology takes the video of dimensions 1920x1080 (FHD), extracts frames, applies few basic image processing operations for detection, and computes each pixel individually.

One of the conventions used in this technique is that the fault always remains less than fifty percent of the frame size. Even if it is more, the remaining part of the cloth is still not usable so there is absolutely no problem with this convention.

After applying the pre-processing operations, the image inverted, if required, to get the coordinates of the fault. The number of pixels were counted and if that is above the calculated threshold of 131 of the fault, the frame is marked faulty.

4.5.2 Accuracy

This technique yielded the accuracy of above 94% on image and up to 100% on video. Detailed results are discussed in the results section of this paper. Because, few images contain very small faults, which are not detected by the system but in video, in the consecutive frame, the fault is either rectified or amplified and hence the accuracy on video is 100% except few scenarios where the frames are blurry or the video feed gets stuck or corrupted, only then it yields inaccuracies.

The processing time of each frame is 0.022 seconds i.e. 45 frames per second on Intel Core i7 6500 CPU without dedicated graphics processor running windows operating system. This FPS is 114 corresponding to processing time of 0.0087s on i7 8700 CPU without GPU. The detailed result of FPS with graphs can be seen in Testing section.

The rigorous testing is done on complete dataset on all kinds of images and all the videos of every nature, and color.

4.6 Fault Detection Solution 3

This approach is very similar to second approach. The only addition is the addition of the vectorization. Each row of the image is taken as vector and using multiple processes at once, feed them dynamically to each thread for counting and processing. This yields double fps on the same CPU and almost 100 fps on Intel Core i9 9900 CPU with the same accuracy.

4.7 Final word about solutions

We prepared three techniques in total but only the last two are commendable due to their efficiency in terms of error rate and processing time. An optimized version where all constants are placed in place of variables like the input feed resolution and similar places, the speed of the system was further increased but it can only be done when there is absolutely no change in these variables. In short, this can be done when system is installed as a product in the cloth fabricating industries.

CHAPTER 5: ALGORTITHM

5.1 Pseudocode

CustomGrayscaleConversion: Conversion to grayscale using weighted formula and our custom weights

Algorithm I:

```
Input: Image Frame
Output: Converted Image
Method:
1: r, g, b ← initialization with our custom/analyzed values
2: Convert to gray scale using the luminosity/weighted average formula.
3: return converted image
4: End
```

Binarization: Apply binarization on image frame

Algorithm II:

```
Input: Image Frame
Output: Binarized Image
Method:
1: thresh ←set threshold
2: Convert to a binary image using the set thresh
3: return binarized image
4: End
```

FaultInversion: Force the fault to be of one color

Algorithm III:

```
Input: Binarized Image
Output: Inverted Image
Method:
1: I ← Apply bitwise operations on frame to force the fault to be inverted.
2: return I
3: End
```

FaultLocalization: Find coordinates of fault present

Algorithm IV:

```
Input: Image with defect b, pixelCount
Output: Defect Coordinates
Method:
1: if (ratioOfDefectedCloth(b,pixelCount))
2:          coord←0,0,maxlen(image),maxwid(image)
3: else
4:          faultAreas = get indices of zero values in b
5:          sx=min(faultAreas[1])
6:          ex=max(faultAreas[1])
7:          sy=min(faultAreas[0])
8:          ey=max(faultAreas[0])
9:          coord←sx,ex,sy,ey
10: return coord
3: End
```

RatioDefectedCloth: Find how much cloth is defected

Algorithm V:

```
Input: image frame m, pixelCount
Output: boolean value
Method:
1: f ← get indices of zero values in m
2: c ← get count of non-zero values in f
3: if ( c > (pixelCount/2) )
4:          return TRUE
5: return FALSE
3: End
```

Algorithm VI: Proposed Defect Detection Approach

```
(1)   Input:  Image Path
(2)   Output: Fault Indicator, Fault Coordinates
(3)   original ← LoadImage(Image Path)
(4)   img ←CustomGrayscaleConversion(original)
(5)   conv ←Binarization(img)
(6)   len,wid←extract length and width of conv
(7)   pixelCount←getImagePixelCount(len,wid)
(8)   if (nonZeroPart(conv) < (pixelCount/2))
(9)       conv←FaultInversionProcedure(conv)
(10)  if (pixelCount – nonZeroPart(conv) > 131)
(11)      fault_indicator = TRUE
(12)      coord←ExtractCoordinatesOfFault(conv,
      pixelCount)
(13)      Drawbox(coord,original)
(14)  else
(15)      fault_indicator = FALSE

(16)  return fault_indicator, coord
```

5.2 Flow Chart

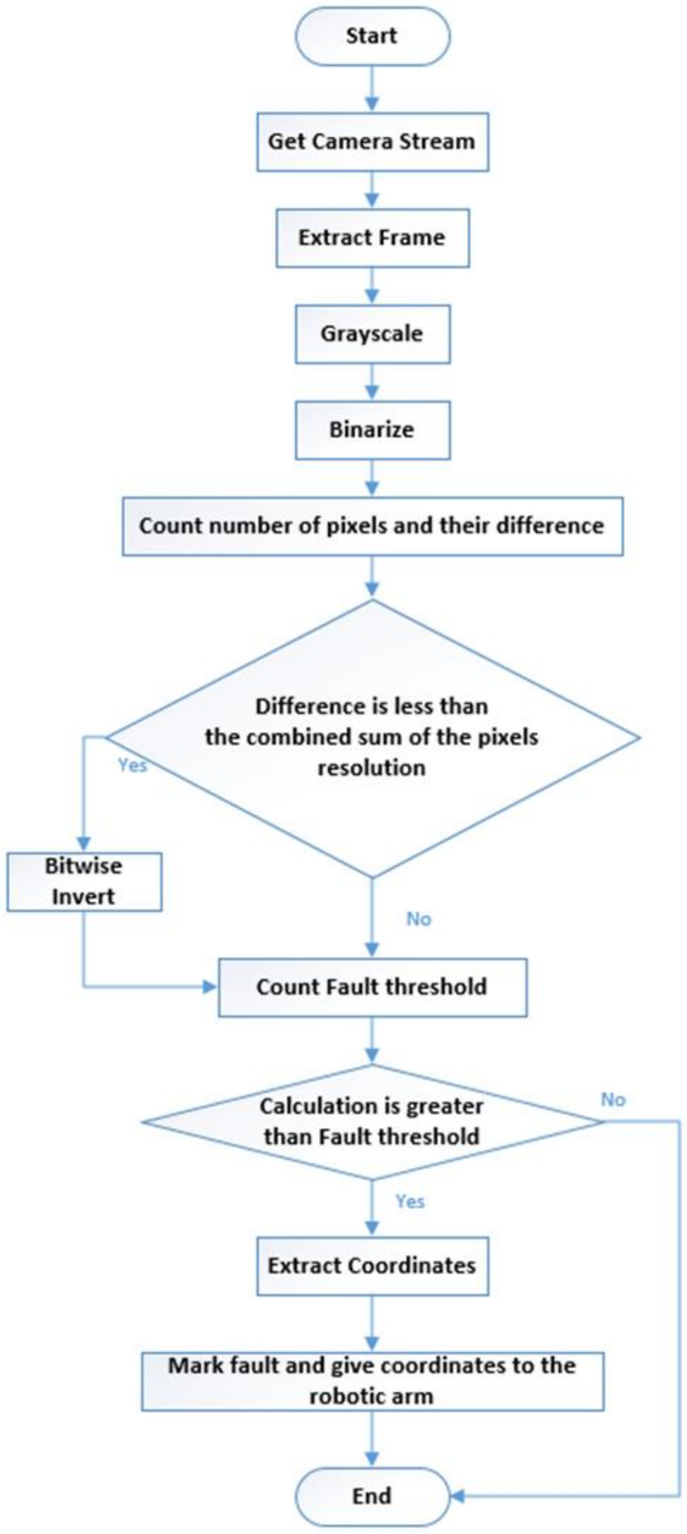

Start
Get Camera Stream
Extract Frame
Grayscale
Binarize
Count number of pixels and their difference
Difference is less than the combined sum of the pixels resolution
Yes
Bitwise Invert
No
Count Fault threshold
Calculation is greater than Fault threshold
No
Yes
Extract Coordinates
Mark fault and give coordinates to the robotic arm
End

Fig.9. Flow Chart of proposed Solution

CHAPTER 6: TESTING

The system is designed on the windows platform first, but later it is also optimized for Linux based system. Due to its lightness, it is also implemented in Raspberry Pi Model 2B. No cameras were attached to feed the live data of the fabric, but the only testing was done from local drive which proved to be very good. The system was choked later due to the power supply issues as it was an old board with an old Raspbian operating system, but we believe that on newer Pi models, it can work flawlessly. The FPS count never dropped below 30 on older board with under rated power supply.

The initial idea was implemented on EmguCV.NET which was later converted to OpenCV to implement it on the Raspberry Pi and Linux based operating systems.

A software application was developed in C# to analyze different lightening conditions, plot different types of graphs and compute different formulae for the final algorithm. The screenshot of the said application can be seen in figure 3.

6.1 Testing in different environments

The system was tested in different environments which include three different operating systems (windows, Linux and Raspbian) and three different devices (intel Core i5 2nd generation, intel Core i7 6th generation, intel Core i9 9th generation).

The testing details including processing speeds, Frames processed per second, on each environment is given below in the table II and III. Frames per second (FPS) is calculated through the equation stated in next section of this paper refer to. (9)

6.1.1 Different Operating Systems

TABLE II

TESTING ON DIFFERENT OPERATING SYSTEMS

Sr.	PROCESSOR	GPU	FPS	Algorithm Time per frame (sec)
1	*Windows (i7-6500)*	No	45	0.022
2	*Linux (i7-8700)*	No	114.9	0.0087
3	*Raspbian*	No	34	0.028

Comparison of Speed of Algorithm on different operating systems.

6.1.2 Different Devices

TABLE III

TESTING ON DIFFERENT DEVICES

Sr.	PROCESSOR	GPU	FPS	Algorithm Time per frame (s)
1	intel Core i5 2nd Gen	No	40	0.025
2	intel Core i7 6th Gen	No	45	0.022
3	intel Core i7 8th Gen	Yes, but not used	114.9	0.0087
4	Intel Core i9 9th Gen	Yes, but not used	180	0.0054
5	Raspberry Pi 2 Model B+	No	34	0.028

Comparison of Speed of Algorithm on different devices.

6.2 Frame rate accuracy in video

The times taken to process each frame and frames per second as accuracy are given for each environment, in table II and III above. Comparison of FPS on those 4 devices is illustrated through graph in the figure no. 10

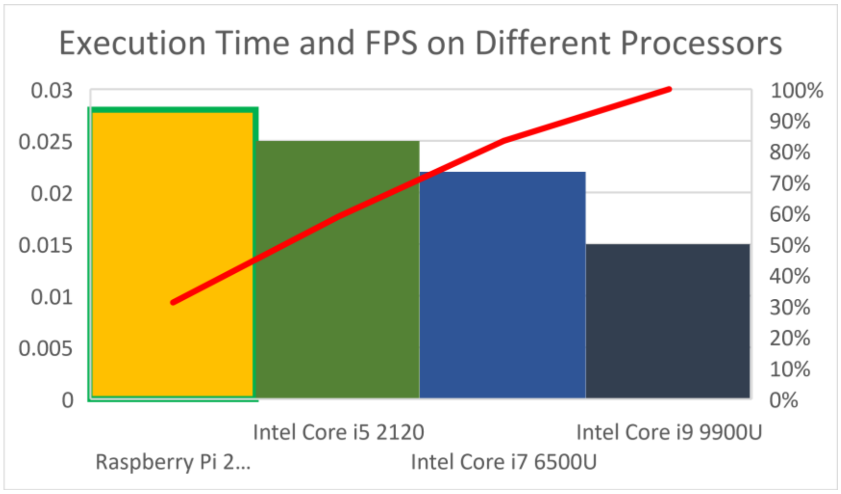


Fig.10. Graph of Testing Results of Execution Time and FPS on different Processors

The time shown in graph is in milliseconds on Y-Axis and the processors used for testing is on X-Axis. The bars display the number of milliseconds taken by each frame and the red line displays the number of frames processed per second for each processor. For raspberry Pi, the FPS was lowest, and time taken by each frame is the highest. While on Core i9 9900 CPU, it was inverse of the said thing and can be seen on the graph.

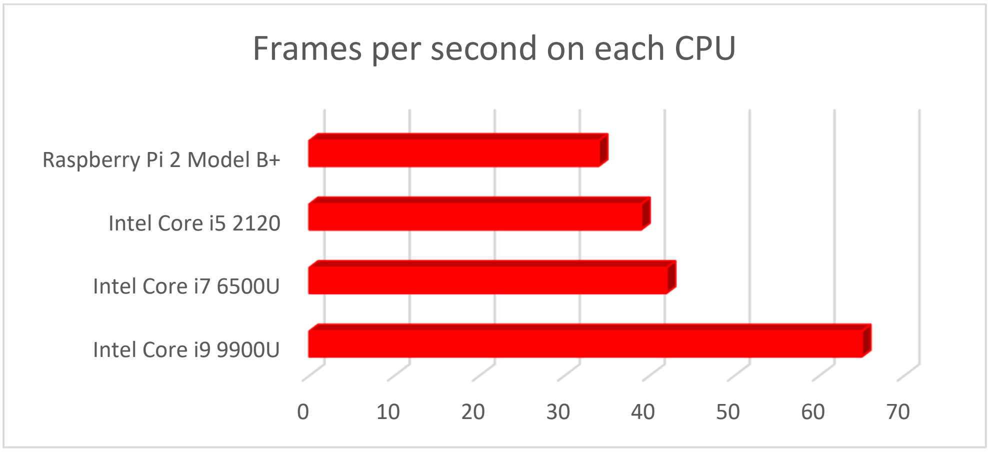


Fig.11. Graph of FPS on each Processor

CHAPTER 7: EXPERIMENTAL RESULTS (QUANTITATIVE)

We will discuss only the results of our final approach in this section. The description of dataset is completely explained in the respective section. The same dataset was used for testing and evaluating purposes. Evaluation was done on another published dataset too but that is explained in next section. For both images and videos, the algorithm was evaluated. A separate data file for the results is also generated which contains the result on each image and if the fault exists, it has the coordinates of the fault. A glimpse is given below in figure no. 24.

The algorithm is fed with a large dataset to perform rigorous testing on 33430 images and results are generated which are as follows.

7.1 Performance Measures

We have measured the efficiency of the overall algorithm through a tool known as confusion matrix. It measures the performance of our algorithm on our manually annotated dataset. We have also measured the performance on a public published labeled dataset in the next section. The confusion matrix has rows and columns where each column of the matrix represents the values in actual/annotated class (defected or defect-free) while each row represents the values of predicted/detected class. The performance metrics that we have used in this paper are based on the count of true positives (TPs), true negative (TNs), false positive (FP). We have categorized them as illustrated below in table no. IV.

TABLE IV
OUTCOMES OF STATISTICAL CLASSIFICATION

	Defected Image	Defect Free Image
Detected as defected	TP	FP
Detected as defect free	FN	TN

Table of outcomes of statistical classification for fabric defect detection.

The values of TP, TN, FP, FN can be used to calculate difference performance metrics to evaluate the overall performance algorithm including specificity, sensitivity, precision, and F-measure.

The performance metrics that we have used for evaluation purposes are following.

7.1.1 Sensitivity or Recall

Sensitivity, which is also known as recall, describes the probability of our algorithm to predict the result as positive (defected) when defects are present in fabric. It is determined by dividing the number of defected images detected correctly with the total number of defected images of fabrics in the dataset. (1) It is also known as true positive rate or TPR.

7.1.2 Specificity

Specificity is the probability of the algorithm to detect the result as negative (defect free) when the cloth is clean. It is determined by dividing the number of defect free images detected correctly with the total number of defect-free images of fabrics in the dataset. (2) Specificity is also known as true negative rate or TNR

7.1.3 Accuracy

Accuracy which is the percentage of the image samples detected correctly is determined by dividing total number of correctly detected, defected and defect free samples with the total samples in dataset. (3) Accuracy is also known as Detection Success Rate or DSR. (6)

7.1.4 Precision

Precision is the positive value of detection. It is the proportion of actual defected image samples of fabric among the samples detected as defected by our algorithm. It tells how much data was actually relevant from the one detected as relevant by the algorithm. (4)

7.1.5 F measure / F Score

F measure or F score is the harmonic mean of the precision and recall. Its best value is 1 (when precision and recall are perfect) and worst value is 0. (5)

7.1.6 DSR

Detection Success rate is the rate of correctly detected image samples from total images in dataset. (6) It is the same as Accuracy. (3)

7.1.7 DR

Detection Rate is the rate of algorithm to detect positive values i.e. detection of defected images from total defected images. (7) It is the same as sensitivity. (1)

7.1.8 FAR

False Alarm rate is rate of algorithm to detect positive results i.e. defected fabric even when defect is not present in the sample. It is determined by dividing number of defect free samples which were detected as defected with total number of defect free samples. (8)

7.1.9 FPS

It is the Frame rate per second also known as Frames per second. We can calculate it by dividing the processing time of our algorithm on each frame or image by 1 to get how many frames are processed in one second.

7.2 *Formulae:*

$$Sensitivity/Recall = \frac{TP}{(TP + FN)}\ x\ 100 \qquad (1)$$

$$Specificity = \frac{TN}{(TN + FP)} x100 \qquad (2)$$

$$Accuracy = \frac{TN + TP}{(TN + FP + FN + TP)} x100 \qquad (3)$$

$$Precision = \frac{TP}{(TP + FP)} x100 \qquad (4)$$

$$F\ measure = 2 * \frac{Precision * Recall}{(Precision + Recall)} \qquad (5)$$

$$DSR = \frac{Correctly\ Detected\ Image\ Samples}{Total\ Samples\ in\ Dataset} \qquad (6)$$

$$DR = \frac{Correctly\ Detected\ Defected\ Samples}{Total\ Defected\ Samples\ in\ Dataset} \qquad (7)$$

$$FAR = \frac{Defect\ Free\ Samples\ detected\ as\ Defected}{Total\ Defect\ free\ Samples\ in\ Dataset} \qquad (8)$$

$$FPS = \frac{1}{Processing\ Speed\ of\ one\ frame} \qquad (9)$$

7.3 Our Results

There were 23568 images which were of defected fabric i.e. they had faults in them. There were 9862 images of clean fabric. The algorithm detected faults on 21658 images out of 23568 images. The only missed images are those which have very small fault which is below the set threshold. Such faults are visible to the human eye but not detected by the system in that frame.

Although this situation does not create a problem as the system is getting continuous video stream. If the fault is very small at the initial stage and it persists continuously in video for few frames, it is either amplified or rectified. In any case respective action is taken. Due to this property, the accuracy on video is up to hundred percent.

TABLE V
ALGORITHM RESULT ON DEFECTED IMAGES

Sr.	Image Category	Count
1	Total Images	33430
2	Total Defected Images	23568
3	Detected as defected	21658
4	Detected as defect free	1910

Result on images of defected fabric in our dataset

TABLE VI
ALGORITHM RESULT ON DEFECT FREE IMAGES

Sr.	Image Category	Count
1	Total Images	33430
2	Total Defect Free Images	9862
3	Detected as defected	0
4	Detected as defect free	9862

Result on images of defect free fabric in our dataset

TABLE VII
CONFUSION MATRIX

	Defected Image	Defect Free Image
Detected as defected	TP 21658	FP 0
Detected as defect free	FN 1910	TN 9862

Values of Confusion matrix of our algorithm for fabric defect detection on our dataset.

7.4 Calculations

TABLE VIII

PERFORMANCE MEASURES

Sr.	Measure	Value
1	Sensitivity/Recall	0.918957909
2	Precision	1
3	F measure	0.957767656
4	Specificity	1
5	Accuracy	**94.2866%**
6	DR	0.918957909
7	FAR	0
8	DSR	0.94286569
9	Average FPS (i7 8700)	114

Values of Performance metrics of our algorithm on our dataset.
All these values have been calculated using the formulae stated in part A of section VII of this paper.

7.5 Types of defects caught

The system catches all the faults which are visible to the human eye. Most importantly cuts, missing cloth, mis fabrication, grease marks, threads, misprint, and all other similar faults. It was in the scope of this research to classify which kind of defect is present, the goal was just to find if the defect is occurring or not regardless of type and get it's location, which is done successfully with good performance metrics for real time detection.

7.6 Frame Rate

Results of processing speed and number of frames processed per second are explained in detail with the help of tables and graphs in previous section (VI) of this paper.

7.7 Image and video error rate difference

The accuracy on image dataset is 94.2866 as of now. The algorithm, when applied on video, yields the accuracy of 100 percent because of the fault persistence and capturing mechanism explained above in part B of this section.

CHAPTER 8: EVALUATION OF ALGORITHM

Evaluation has been done through two different methods. First method has a common public dataset while second has our dataset common. They both are explained below in detail in their sections along with the comparison of their results.

8.1 First Method of Evaluation: Public Dataset

A public published dataset is downloaded and used in this method of evaluation. That dataset is kept common while two different algorithms/techniques are applied on it. One is our technique and the other is the technique which was already applied and published along with results with that public dataset. Results of both are compared and stated below

8.1.1 Public Dataset Description

Javier and his team have compiled a public annotated dataset of images with and without defects of fabric to provide a benchmark so that direct comparison of fabric defect detection methods can be made. They have published it on 4, December 2019 in AUTEX Research Journal which is focused on Textile research. [6] Their dataset is one of the latest datasets available. [90] They have reviewed different existing techniques and applied one using Gabor filters on their dataset in their paper. We have compared the results of their technique with ours on their dataset keeping their dataset as single entity to compare results properly, below in table no. IX and table no X.

8.1.2 Results of Gabor filter technique on public dataset

They have published two results using Gabor filter technique using two different values of sigma. Both their results are given below in table no. IX.

TABLE IX
PERFORMANCE MEASURES OF GABOR FILTER ON PUBLIC DATASET
TEST 1

Sr.	Measure	Value in %
1	Sensitivity/DR	78.10
2	Specificity	97.14
3	FAR	2.90
4	DSR/Accuracy	88.98

TEST 2

Sr.	Measure	Value in %
1	Sensitivity/DR	86.67
2	Specificity	88.57
3	FAR	11.4
4	DSR/Accuracy	87.76

Values of Performance metrics of Gabor filter technique on public dataset. All these values have been calculated using the formulae stated in part A of section VII of this chapter.

8.1.3 Results of our algorithm on public dataset

TABLE XIV
CONFUSION MATRIX OF PUBLIC DATASET

	Defected Image	Defect Free Image
Detected as defected	TP 105	FP 16
Detected as defect free	FN 1	TN 125

Values of Confusion matrix of our algorithm for fabric defect detection on our dataset.

TABLE X
PERFORMANCE MEASURES OF OUR ALGORITHM ON PUBLIC DATASET

Sr.	Measure	Value in %
1	Sensitivity/DR	99.05
2	Specificity	88.652
3	FAR	11.3475
4	DSR/Accuracy	93.11741

Values of Performance metrics of our algorithm on public dataset. All these values have been calculated using the formulae stated in part A of section VII of this paper.

8.2 Second Method of Evaluation: Our Dataset

In the second method we have implemented the technique of the Gabor filters used in the paper published by Javier and his team [6]. We have applied that technique and our algorithm on our dataset. So, in this method of evaluation we have used the dataset collected by us as a common entity and compared the results of both techniques on it.

8.2.1 Our Dataset Description

We have described our dataset in detail section X.

8.2.2 Results of our algorithm on our dataset

We have described our results in detail in part C of section VII.

8.2.3 Results of Gabor filter technique on our dataset

Refer to next page please.

TABLE XI
PERFORMANCE MEASURES OF GABOR FILTER ON OUR DATASET

Sr.	Measure	Value
1	Sensitivity/Recall	0.887559403
2	Precision	0.949911448
3	F measure	0.91767751
4	Specificity	0.888156561
5	Accuracy	88.7736%
6	DR	0.887559403
7	FAR	0.111843439
8	DSR	0.887735567
9	FPS	3.9

Values of Performance metrics of Gabor filter technique on our dataset.
All these values have been calculated using the formulae stated in part A of Chapter VII of this thesis.

CHAPTER 9: DATASET

Video and Image acquisition systems have been discussed in detail in above respective section.

NOTE This section cover details for the dataset we have collected ourselves, not the public dataset which has been discussed in above sections.

9.1 Data Collection

For the collection of datasets, the same system was used. We deployed that system on a fabric machine in a textile factory. Name of the factory is confidential that is why it is not mentioned.

On the fabric machine bolt of fabric moves with a speed of 20 to 25 meters per minute. Traditionally the marking of defected fabric is done by human workers. Their format of marking and removing the defected cloth was observed. Whenever a defected appeared they marked the cloth while it was moving on the conveyer belt. By doing so, their hand would appear in the frame of vision (FOV) of our camera. We wrote a python script that would recognize human hand using different skin detection algorithms. To collect the defected data, we wrote a multi-threaded python script that was recording video continuously on one thread, recording the time stamp on which a hand would appear in the FOV of camera on the other thread in parallel and saving in excel file, while the third thread would process videos simultaneously by cropping a +-10 second portion of videos captured by first thread around the timestamp of defected cloth saved by second thread. The duration of each thread was 1 hour i.e. they would save videos and timestamps hour by hour. They script was run in loop for 24 hours and 7 days a week for around three months on the textile machine. The third thread would always process the videos of previous hour after they were saved.

Later, frames were extracted from those cropped videos that were saved by third thread, which were 107000 in number. Since the defected frames were gathered through code of hand recognition and they were from a video portion of 10 seconds before and after appearing of hand, they are not reliable and have both defected, defect free and use less images in them. Therefore, the data was manually analyzed, and each image was manually annotated.

9.2 Annotation of Defected and Non defected Dataset

The dataset gathered through our system was of 103520 images/frames, which were extracted from videos. Not all of it was useful. We manually analyzed and marked/annotated it. Since each and every image in dataset is manually marked as defected or defect-free, the dataset we have compiled is real and trustworthy.

We have done annotation through two methods. For the first 5000 images we have used a tool named “labelImg” to draw boxes on the defect manually and the coordinates of boxes were saved in xml files, for the rest of the images we have manually analyzed each image generated excel files to categorize image in both categories.

So, at the end final dataset which is manually marked and used to test our algorithm is of 33430 images. The details are described below.

9.3 Description of the collected and annotated data

There are three types of datasets used in the implementation, two of which we have gathered. One is image and second is the video dataset. The third one is the public published dataset [6] which is discussed in previous sections of this paper. Both the datasets are captured using some parameters and set of rules described below.

9.3.1 Video Dataset

Video dataset is only used for test FPS and to ensure the working of this algorithm in real time as a product.

The fundamental parameters of videos captured are.

TABLE XII

VIDEO DATASET SPECIFICATIONS

Sr.	Measure	Value
1	Resolution:	1920x1080
2	Frame Rate:	30 to 300 fps
3	Format:	mp4
4	Color Format:	RGB
5	Typical Length:	10 seconds
6	Average Bit rate:	3800 bps
7	Total Videos	468

9.3.2 Image Dataset

The fundamentals of image dataset are,

TABLE XIII

IMAGE DATASET SPECIFICATIONS

Sr.	Measure	Value
1	Resolution:	1920x1080
2	Color	RGB
3	Bit Depth	24
4	Total Images	33430
5	Defected Images	23568
6	Defect Free Images	9862

Specifications of Image dataset

Results of this dataset are discussed in detail in the results section, including accuracies and all other essential factors for both video and images.

9.4 Types of defects in dataset

There are all types of common defects present in dataset which are visible to the human eye, as the system was incorporated in a textile factory and recording was performed 24 hours every day for some months. Therefore, all the possible and real time defects which occur in the industry are present in the dataset most importantly cuts, missing cloth, mis fabrication, grease marks, threads, lines, and all other similar faults.

CHAPTER 10: FUTURE WORKS

For the future work, we intend to work on several aspects, which includes working on very minor faults which are difficult to detect by bringing changes to our current algorithm or proposing a new one. We also intend to increase the dataset further and work on more types of faults. Furthermore, we intend to work on using different but complex techniques for more precise, flawless, and accurate detection and markings of the fault, two of which are explained briefly below.

10.1 Infrared LED Matrix Transceiver system

This technique will give almost hundred percent results. The idea is to install an array of Infrared throwing LEDs over a fabricator, somewhere along the camera and the receiver of the IR LED to be installed below the cloth. Whenever there is a cut or missing cloth, it will be immediately marked with this technique.

In current system, the background is very light in color. So, if the defected fabric is in the similar shade, the probability of missing the fault increases. With this technique, it is almost impossible to miss such faults.

10.2 Ultraviolet guns transceiver system

In the current system, when there are grease marks over a dark fabric, like blackish mark on a black or grey cloth, the probability of missing a fault increase. If we install an Ultraviolet gun which throws a beam of such light, which is deflected and then detected by the sensor, it will almost never miss such faults. If we set the angle of the gun to be 45 degrees as of the camera and the cloth. The sensor should be installed somewhere opposite of beneath the cloth which is still yet to be decided by the experiments, but this will increase the accuracy insanely.

CHAPTER 11: QUALITATIVE RESULTS AND VISUALIZATION OF WORKING OF ALGORITHM

Qualitative results will be presented in this chapter. The output images of experimental tests can be seen below. Images from different stages of our approach are also included.

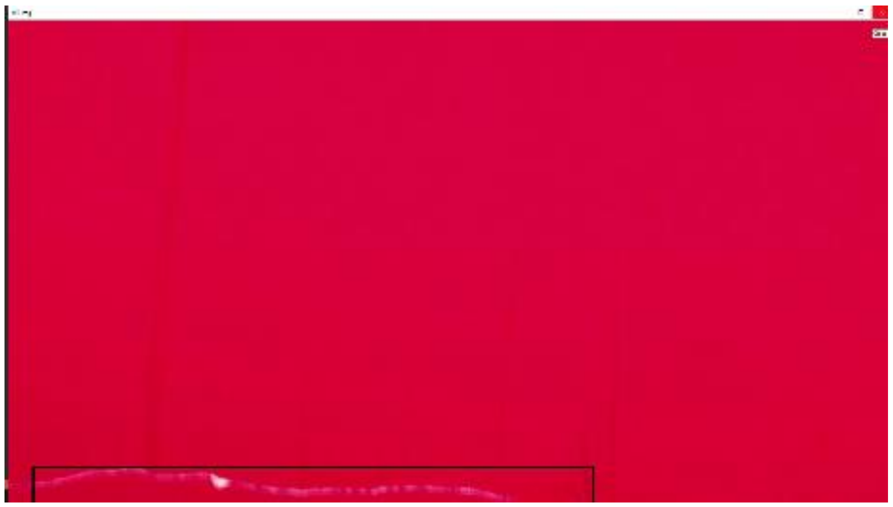


Fig.12. Detected Defect

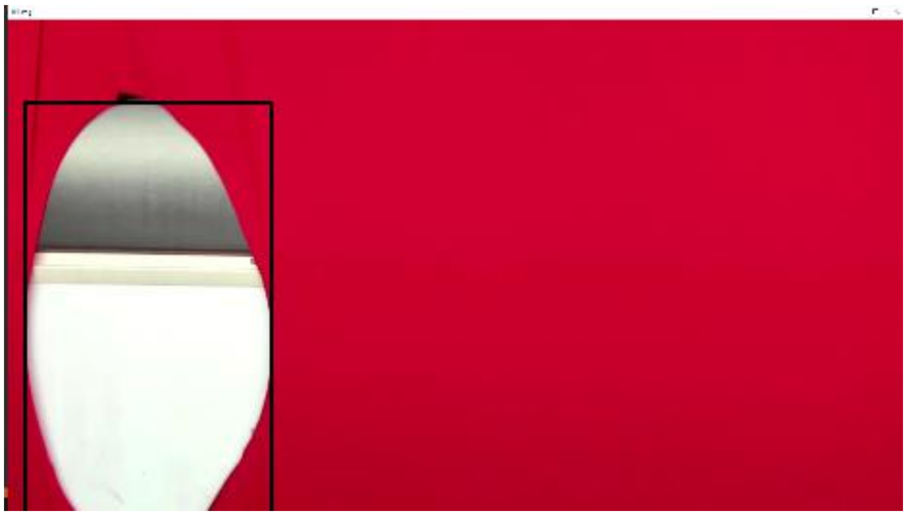


Fig.13. Detected Defect

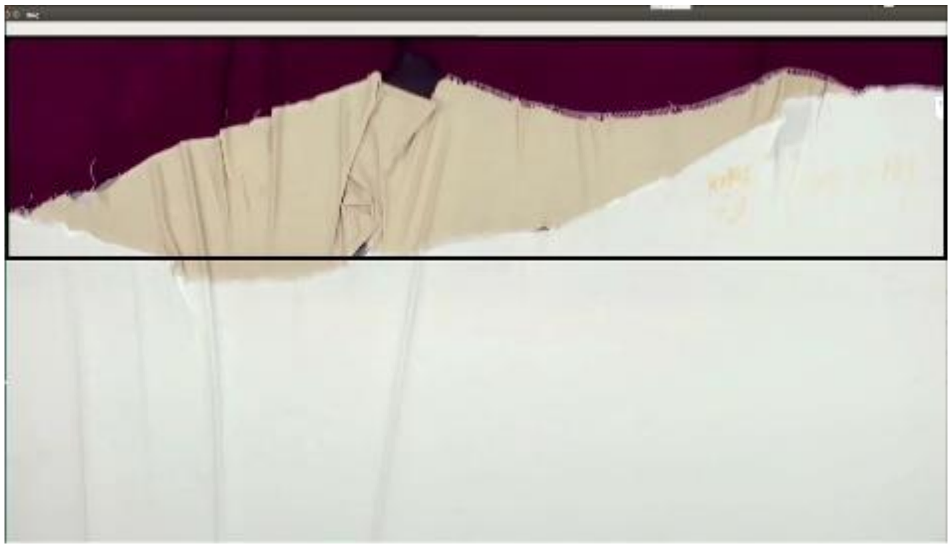


Fig.14. Detected Defect

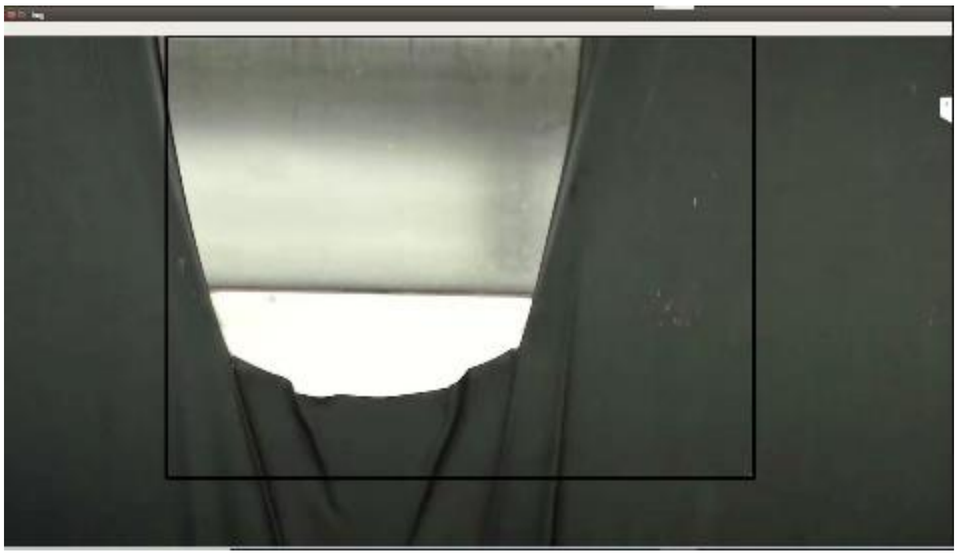


Fig.15. Detected Defect

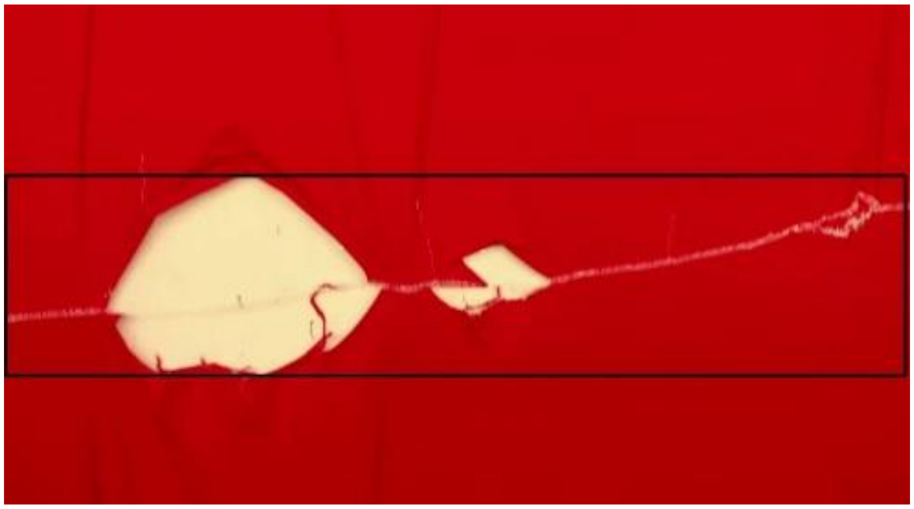
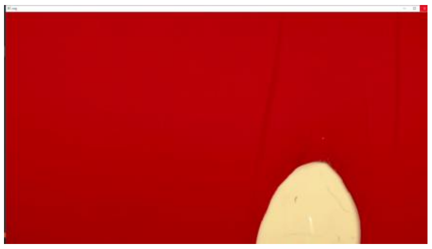
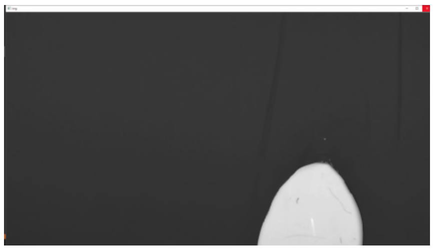
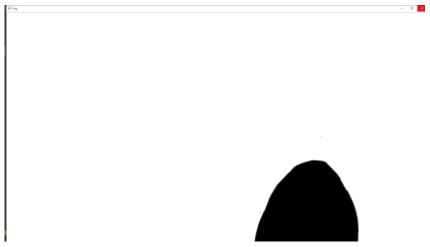
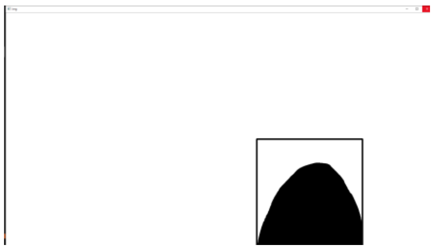


Fig.16. Detected Defect


17(a)


17(b)


17(c)


17(d)

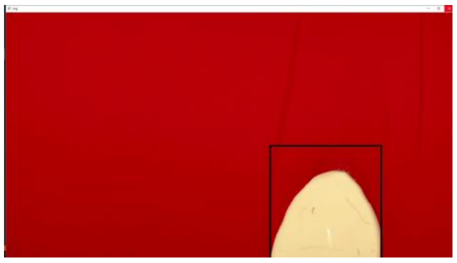
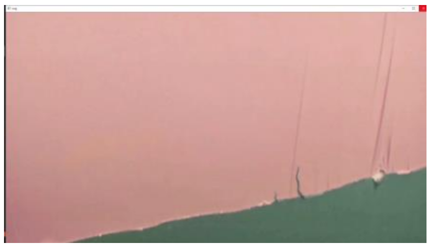
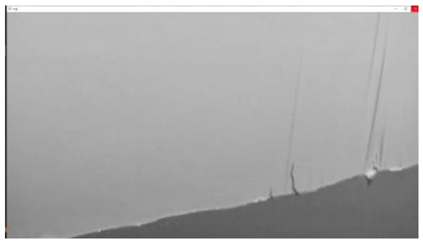
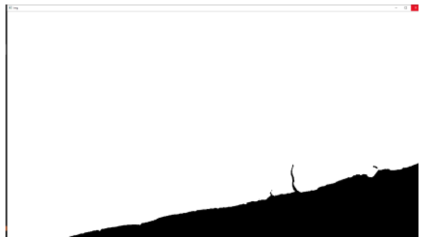
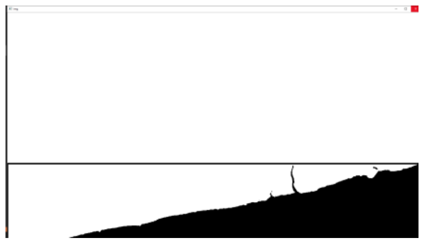

17(e)

Fig.17. Test Results stepwise. 17(a) Original Image of defected Cloth sample. 17(b) Customized grayscale converted image. 17(c) Binarized image on a custom threshold. 17(d) Fault localization on binarized image. 17(e) Fault localization and drawn boxes on the original image.


18(a)


18(b)


18(c)


18(d)

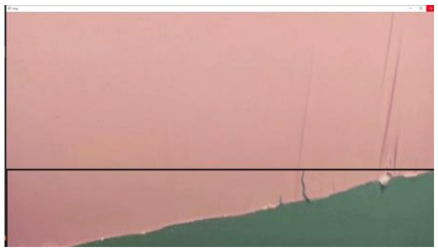
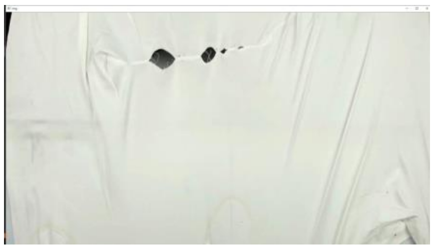
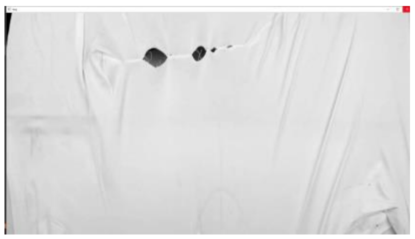
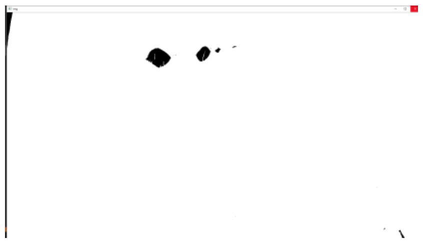
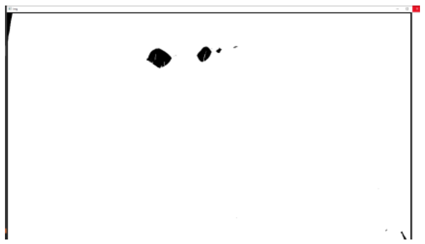

18(e)

Fig.18. Test Results stepwise. 18(a) Original Image of defected Cloth sample. 18(b) Customized grayscale converted image. 18(c) Binarized image on a custom threshold. 18(d) Fault localization on binarized image. 18(e) Fault localization and drawn boxes on the original image.


19(a)


19(b)


19(c)


19(d)

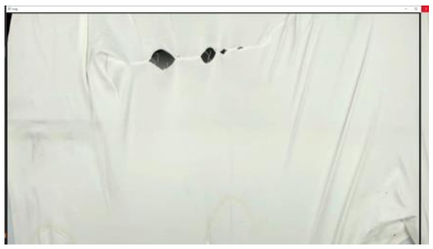
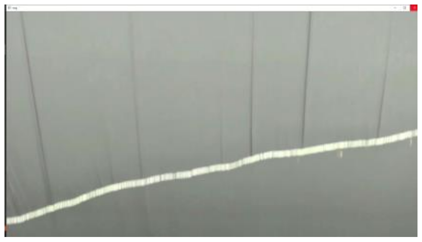
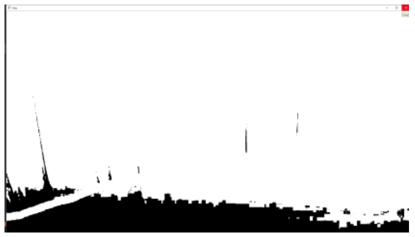
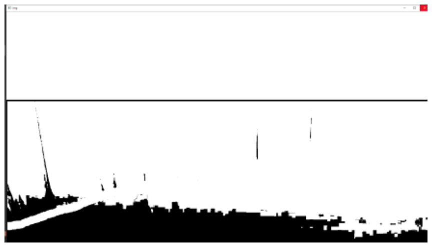

19(e)

Fig.19. Test Results stepwise. 19(a) Original Image of defected Cloth sample. 19(b) Customized grayscale converted image. 19(c) Binarized image on a custom threshold. 19(d) Fault localization on binarized image. 19(e) Fault localization and drawn boxes on the original image.


20(a)

20(b)


20(c)


20(d)

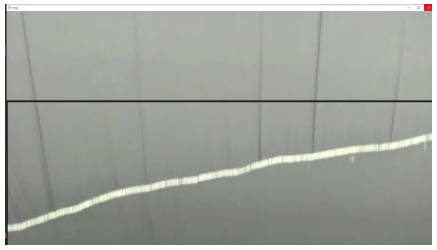
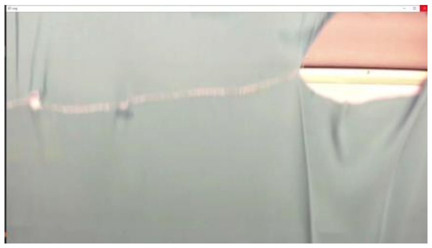
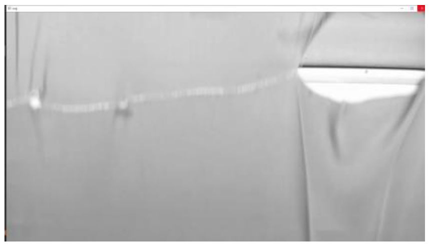

20(e)

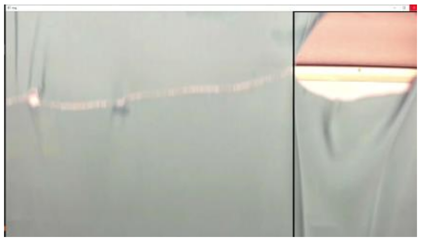
Fig.20. Test Results stepwise. 20(a) Original Image of defected Cloth sample. 20(b) Customized grayscale converted image. 20(c) Binarized image on a custom threshold. 20(d) Fault localization on binarized image. 20(e) Fault localization and drawn boxes on the original image.



21(a)

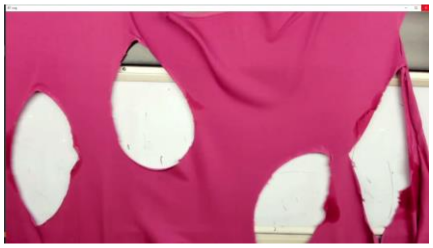


21(b)

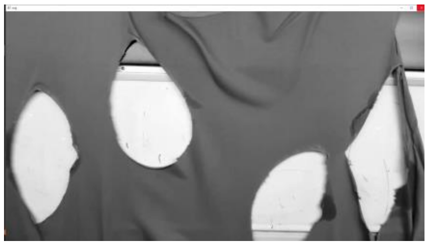


21(c)

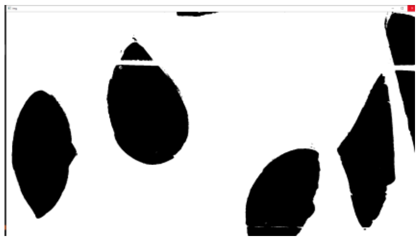
Fig.21. Test Results stepwise. 21(a) Original Image of defected Cloth sample. 21(b) Customized grayscale converted image. 21(c) Fault localization and drawn boxes on the original image.



22(a)

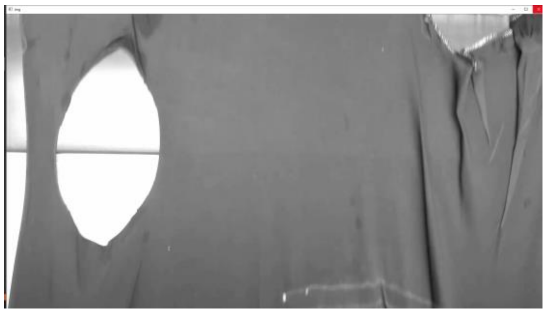


22(b)

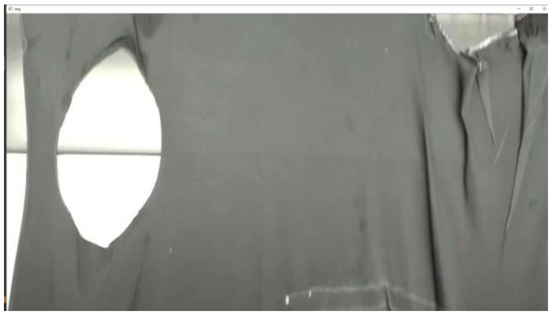


22(c)

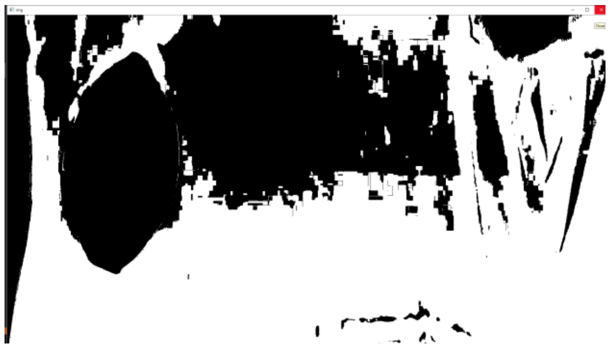
Fig.22. Test Results stepwise. 22(a) Original Image of defected Cloth sample. 22(b) Customized grayscale converted image. 22(c) Binarized image on custom threshold.



23(a)

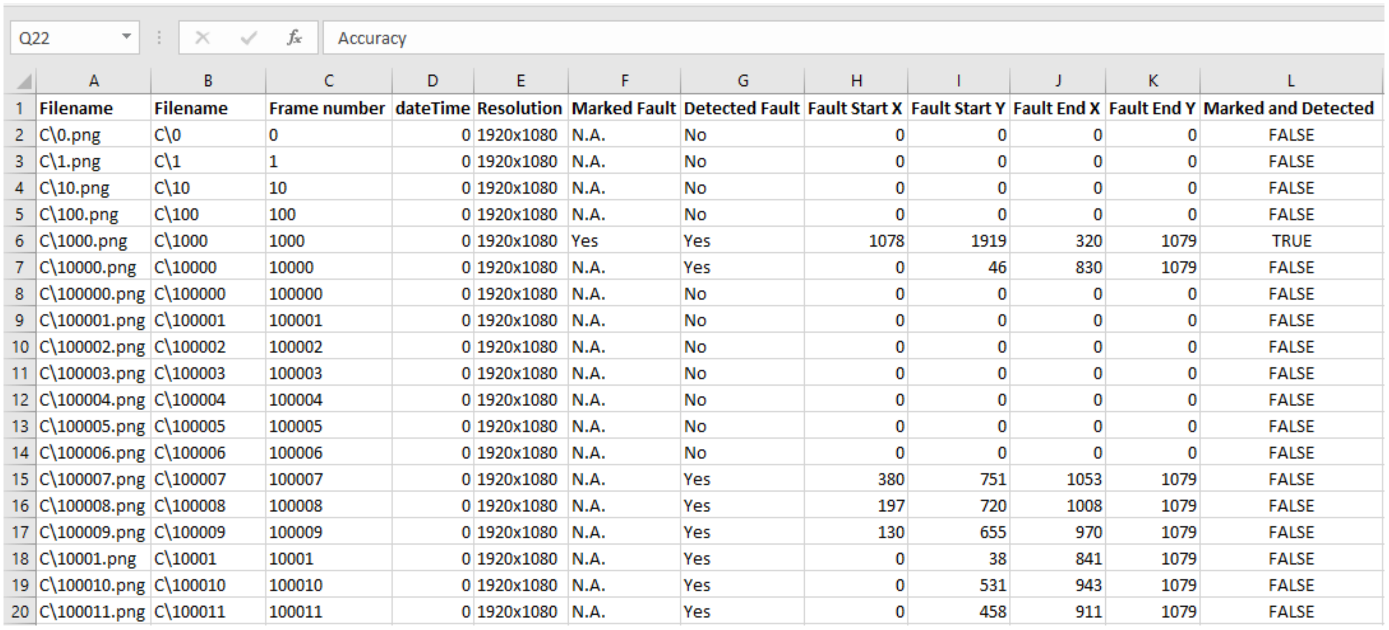


23(b)



23(c)

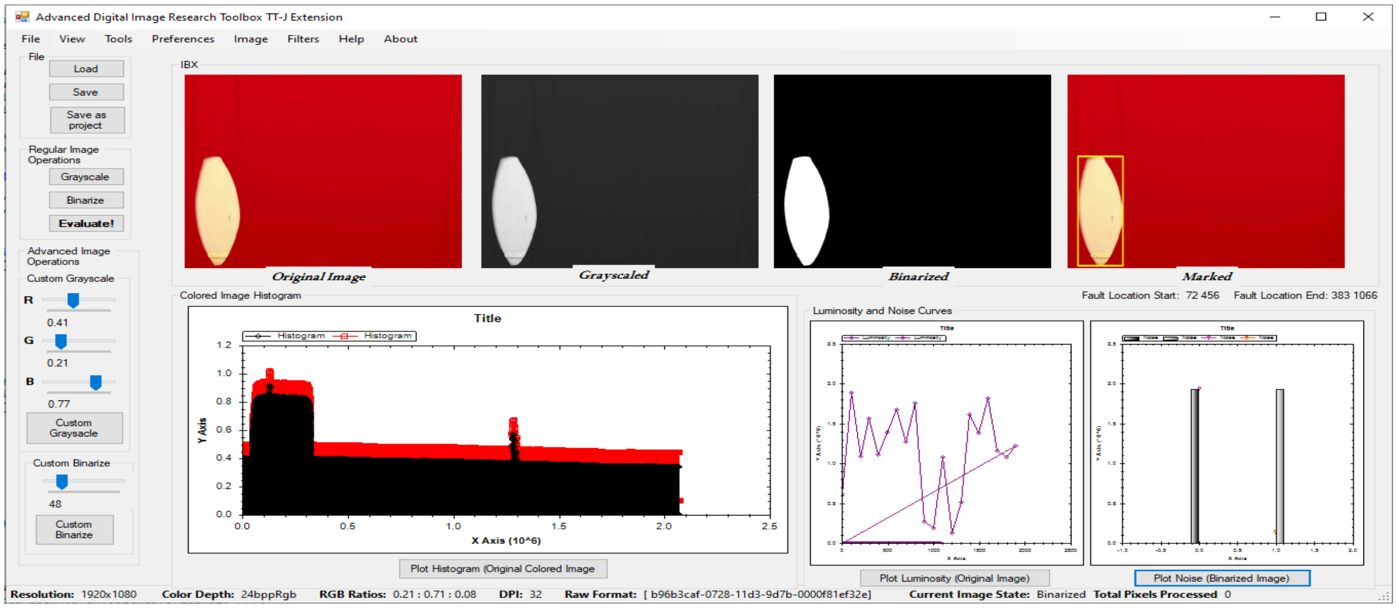
Fig.23. Test Results stepwise. 23(a) Original Image of defected Cloth sample. 23(b) Customized grayscale converted image. 23(c) Binarized image on custom threshold.



Q22 | Accuracy

	A	B	C	D	E	F	G	H	I	J	K	L
1	Filename	Filename	Frame number	dateTime	Resolution	Marked Fault	Detected Fault	Fault Start X	Fault Start Y	Fault End X	Fault End Y	Marked and Detected
2	C\0.png	C\0	0	0	1920x1080	N.A.	No	0	0	0	0	FALSE
3	C\1.png	C\1	1	0	1920x1080	N.A.	No	0	0	0	0	FALSE
4	C\10.png	C\10	10	0	1920x1080	N.A.	No	0	0	0	0	FALSE
5	C\100.png	C\100	100	0	1920x1080	N.A.	No	0	0	0	0	FALSE
6	C\1000.png	C\1000	1000	0	1920x1080	Yes	Yes	1078	1919	320	1079	TRUE
7	C\10000.png	C\10000	10000	0	1920x1080	N.A.	Yes	0	46	830	1079	FALSE
8	C\100000.png	C\100000	100000	0	1920x1080	N.A.	No	0	0	0	0	FALSE
9	C\100001.png	C\100001	100001	0	1920x1080	N.A.	No	0	0	0	0	FALSE
10	C\100002.png	C\100002	100002	0	1920x1080	N.A.	No	0	0	0	0	FALSE
11	C\100003.png	C\100003	100003	0	1920x1080	N.A.	No	0	0	0	0	FALSE
12	C\100004.png	C\100004	100004	0	1920x1080	N.A.	No	0	0	0	0	FALSE
13	C\100005.png	C\100005	100005	0	1920x1080	N.A.	No	0	0	0	0	FALSE
14	C\100006.png	C\100006	100006	0	1920x1080	N.A.	No	0	0	0	0	FALSE
15	C\100007.png	C\100007	100007	0	1920x1080	N.A.	Yes	380	751	1053	1079	FALSE
16	C\100008.png	C\100008	100008	0	1920x1080	N.A.	Yes	197	720	1008	1079	FALSE
17	C\100009.png	C\100009	100009	0	1920x1080	N.A.	Yes	130	655	970	1079	FALSE
18	C\10001.png	C\10001	10001	0	1920x1080	N.A.	Yes	0	38	841	1079	FALSE
19	C\100010.png	C\100010	100010	0	1920x1080	N.A.	Yes	0	531	943	1079	FALSE
20	C\100011.png	C\100011	100011	0	1920x1080	N.A.	Yes	0	458	911	1079	FALSE

Fig.24. Sample of Excel Files generated by testing system



R 0.41
G 0.21
B 0.77
Custom Binarize 48

Fault Location Start: 72 456 Fault Location End: 383 1066

Resolution: 1920x1080 Color Depth: 24bppRgb RGB Ratios: 0.21 : 0.71 : 0.08 DPI: 32 Raw Format: [b96b3caf-0728-11d3-9d7b-0000f81ef32e] Current Image State: Binarized Total Pixels Processed 0

Fig.25. C# Software App which we developed to analyze dataset and design our algorithm.

CHAPTER 12: CONCLUSION

In this research project, we have proposed a new algorithm for the automation of textile defect inspection system. Our algorithm will detect all kinds of visible defects and can work in real time in fabric industry because of its fast processing speed. We have performed rigorous testing of our algorithm using a huge amount of test data which includes ~34,000 image samples of both defected and defect-free images and ~500 videos (exact details of dataset are mentioned in section IX). Our accuracy on limited number of images (~350 images) was almost 100% but it dropped somewhat when we increased the dataset for thorough testing. Many previous works have used dataset ranging from 2 images to 300 images and showing 100% accuracies, which does not contribute to a considerable result for real time working of their approaches.

We have evaluated all the results of our method properly through two ways. Accuracy and other quantitative results cannot just be compared with other research papers as both their dataset and approach are different than yours. For proper comparison and evaluation one entity must be same. For this purpose, we used a public dataset and the results of its approach. We ran our algorithm on that public dataset to get our results, then compared those results with their approach's results. This way we kept the public dataset common between both the approaches and our accuracy and FPS were higher. The second method of our evaluation was to keep our dataset a common entity between our approach and another approach with which we have our results and our results were again higher. We have discussed the results and evaluation thoroughly in detail in section VII and VIII. So, we have actually used two datasets for testing purposes.

We have tested the FPS and processing speed of our algorithm on different devices and different operating system for proper testing. The result was of course different on each. The devices included Intel core i5 2nd Gen, i7 6th Gen, i7 8th Gen, i9 9th Gen and Raspberry Pi 2 Model B+. The operating systems included Windows 10, Linux Ubuntu 16.04, and Raspbian. GPU is not used in any case of testing, only CPU is used.

Our method is superior to others because of our huge dataset corresponding to rigorous testing, our high processing speed on a personal laptop having simple specifications without the use of GPU hence cost effective and our good accuracy. No previous work has a processing speed or Frame rate equal to us even after using GPUs because of the simplicity of our algorithm.

We have achieved an accuracy of 94.3% and an FPS of 114. Our FP (False positive) is zero giving precision and specificity or TNR as 1 or 100%. This means our algorithm will never detect a clean cloth as defected and good fabric will never be wasted due to wrong marking. In other words, false alarm rate (FAR) is zero. Although our algorithm is sensitive towards FN (False Negative) as FN is not zero which is the only reason our accuracy was less than 100% on images. Due to this our Recall/Sensitivity or TPR is 0.92%. This means there is a slight possibility that some of the defected fabric will be not detected. We analyzed our FN samples and came to conclusion that these were those samples which were either faded/bright due to sudden changes in light intensity or had a very small defect size which was not properly visible in the camera FOV. Although, this is not a problem in real time working of inspection system because in real time continuous video stream is being processed with 30 or 60 frames (images) per second so, if the defect is not caught in one frame (image) it will be caught in the next 2 to 5 frames. This is because the light intensity is normalized again in next frames and if the fault is very small in one frame at initial stage, it persists continuously in video for few frames, it is either amplified or rectified and hence caught. This is the reason our accuracy on videos in up to 100%. Both precision and recall contributed to our F-measure score which is 0.96%.

We have deployed our system in a textile factory for collection of data and for testing purposes as well.

REFERENCES

[1] Kumar, A. (2008). Computer-vision-based fabric defect detection: A survey. IEEE transactions on industrial electronics, 55(1), 348-363.

[2] Cho, C. S., Chung, B. M., & Park, M. J. (2005). Development of real-time vision-based fabric inspection system. IEEE Transactions on Industrial Electronics, 52(4), 1073-1079.

[3] Ngan, H. Y., Pang, G. K., & Yung, N. H. (2011). Automated fabric defect detection—a review. Image and vision computing, 29(7), 442-458.

[4] Hanbay, K., Talu, M. F., & Özgüven, Ö. F. (2016). Fabric defect detection systems and methods—A systematic literature review. Optik, 127(24), 11960-11973.

[5] Goyal, A. (2018). Automation in fabric inspection. In Automation in Garment Manufacturing (pp. 75-107). Woodhead Publishing.

[6] Silvestre-Blanes, J., Albero-Albero, T., Miralles, I., Pérez-Llorens, R., & Moreno, J. (2019). A Public Fabric Database for Defect Detection Methods and Results. Autex Research Journal, 19(4), 363-374.

[7] The Berkeley Segmentation Dataset and Benchmark. (2013). [Online]. Website: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html.

[8] Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2010). Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence, 33(5), 898-916.

[9] Brodatz, P. (1996). Textures: a photographic album for artists and designers, Dover, New York. Web site: http:// www.ux.uis.no/~tranden/brodatz.html.

[10] MIT MediaLab. (1995). VisTex texture database. [Online]. Website: http://vismod.media.mit.edu/vismod/imagery/ VisionTexture/vistex.html.

[11] Dong, Y., & Ma, J. (2011). Bayesian texture classification based on contourlet transform and BYY harmony learning of Poisson mixtures. IEEE Transactions on Image Processing, 21(3), 909-918.

[12] Selvan, S., & Ramakrishnan, S. (2007). SVD-based modeling for image texture classification using wavelet transformation. IEEE transactions on image processing, 16(11), 2688-2696.

[13] Li, L., Tong, C. S., & Choy, S. K. (2010). Texture classification using refined histogram. IEEE Transactions on Image Processing, 19(5), 1371-1378.

[14] Dana, K. J., Van Ginneken, B., Nayar, S. K., & Koenderink, J. J. (1999). Reflectance and texture of real-world surfaces. ACM Transactions On Graphics (TOG), 18(1), 1-34.

[15] Fritz, M., Hayman, E., Caputo, B., Eklundh, J.-O. (2006). Kth-tips database. [Online]. Website: http://www.nada.kth. se/cvap/databases/kth-tips/.

[16] Tajeripour, F., Kabier, E., Sheikhi, A. (2008). Fabric defect detection using modified local binary patterns. EURASIP Journal on Advances in Signal Processing, 2008, 60.

[17] Ngan, H. Y., & Pang, G. K. (2008). Regularity analysis for patterned texture inspection. IEEE Transactions on automation science and engineering, 6(1), 131-144.

[18] Yang, X. Z., Pang, G. K. H., & Yung, N. H. C. (2002). Discriminative fabric defect detection using adaptive. Optical Engineering, 41(12), 3116-3125.

[19] Ngan, H. Y., Pang, G. K., Yung, S., Ng, M. K. (2005). Wavelet based methods on patterned fabric defect detection. Pattern Recognition, 38(4), 559-576.

[20] Technische Universitt Hamburg-Harburg. (1995). TILDA Textile Texture-Database. [Online].
Website: http://lmb.informatik.uni-freiburg.de/resources/datasets/tilda. en.html.

[21] PARVIS, “PARVIS,” provided by PARVIS srl, [Online]. Web site: http://www.parvis.it/.

[22] Zhu, S. W., Hao, H. Y., Li, P. Y., Shi, M. H., & Qi, H. (2007, August). Fabric defects segmentation approach based on texture primitive. In 2007 International Conference on Machine Learning and Cybernetics (Vol. 3, pp. 1596-1600). IEEE.

[23] Sari-Sarraf, H., & Goddard, J. S. (1998, May). Vision system for on-loom fabric inspection. In 1998 IEEE Annual Textile, Fiber and Film Industry Technical Conference (Cat. No. 98CH36246) (pp. 8-1). IEEE.

[24] Tiwari, V., & Sharma, G. (2015). Automatic fabric fault detection using morphological operations on bit plane. International Journal of Computer Science and Network Security (IJCSNS), 15(10), 30.

[25] Conci, A., & Proença, C. B. (1998). A fractal image analysis system for fabric inspection based on a box-counting method. Computer Networks and ISDN Systems, 30(20-21), 1887-1895.

[26] Liu, Z. (2009, December). Computer testing method of defect feature of fabric. In 2009 International Conference on Test and Measurement (Vol. 1, pp. 169-172). IEEE.

[27] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). Computer vision and image understanding, 110(3), 346-359.

[28] Bissi, L., Baruffa, G., Placidi, P., Ricci, E., Scorzoni, A., & Valigi, P. (2013). Automated defect detection in uniform and structured fabrics using Gabor filters and PCA. Journal of Visual Communication and Image Representation, 24(7), 838-845.

[29] Yu, X., Hu, J., & Baciu, G. (2005). Defect detection of jacquard fabrics using multiple color-channel analysis. Research Journal of Textile and Apparel, 9(1), 21.

[30] Chan, C. H., & Pang, G. (1999, October). Fabric defect detection by Fourier analysis. In Conference Record of the 1999 IEEE Industry Applications Conference. Thirty-Forth IAS Annual Meeting (Cat. No. 99CH36370) (Vol. 3, pp. 1743-1750). IEEE.

[31] Malek, A. S., Drean, J. Y., Bigue, L., & Osselin, J. F. (2013). Optimization of automated online fabric inspection by fast Fourier transform (FFT) and cross-correlation. Textile Research Journal, 83(3), 256-268.

[32] Hu, G. H. (2014, April). Optimal ring Gabor filter design for texture defect detection using a simulated annealing algorithm. In 2014 International Conference on Information Science, Electronics and Electrical Engineering (Vol. 2, pp. 860-864). IEEE.

[33] Han, Y., & Shi, P. (2007). An adaptive level-selecting wavelet transform for texture defect detection. Image and Vision Computing, 25(8), 1239-1248.

[34] Guan, S., & Gao, Z. (2014). Fabric defect image segmentation based on the visual attention mechanism of the wavelet domain. Textile Research Journal, 84(10), 1018-1033.

[35] Kumar, A., & Pang, G. (2001, September). Identification of surface defects in textured materials using wavelet packets. In Conference Record of the 2001 IEEE Industry Applications Conference. 36th IAS Annual Meeting (Cat. No. 01CH37248) (Vol. 1, pp. 247-251). IEEE.

[36] Yang, X., Pang, G., & Yung, N. (2005). Robust fabric defect detection and classification using multiple adaptive wavelets. IEE Proceedings-Vision, Image and Signal Processing, 152(6), 715-723.

[37] Mak, K. L., Peng, P., Lau, H. Y. (2005). Optimal morphological filter design for fabric defect detection, in Industrial Technology, 2005. ICIT 2005. IEEE International Conference on, Hong Kong.

[38] Mak, K. L., Peng, P., & Yiu, K. F. C. (2009). Fabric defect detection using morphological filters. Image and Vision Computing, 27(10), 1585-1592.

[39] Mallat, S. (1999). A wavelet tour of signal processing. Elsevier.

[40] Kumar, A., & Pang, G. K. (2002). Defect detection in textured materials using optimized filters. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 32(5), 553-570.

[41] Mei, S., Wang, Y., & Wen, G. (2018). Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model. Sensors, 18(4), 1064.

[42] Rebhi, A., Benmhammed, I., Abid, S., & Fnaiech, F. (2015). Fabric defect detection using local homogeneity analysis and neural network. Journal of photonics, 2015.

[43] Jadhav, S. P., & Biradar, M. S. Fabric Defect Detection by using Neural Network technique.

[44] Kumar, A. (2003). Neural network based detection of local textile defects. Pattern Recognition, 36(7), 1645-1659.

[45] Banumathi, P., & Nasira, G. M. (2012). Fabric inspection system using artificial neural networks. Int. J. Computer Engin. Sci, 2(5), 20-27.

[46] Kang, Z., Yuan, C., & Yang, Q. (2013, August). The fabric defect detection technology based on wavelet transform and neural network convergence. In 2013 IEEE International Conference on Information and Automation (ICIA) (pp. 597-601). IEEE.

[47] Mursalin, T. E., Eishita, F. Z., & Islam, A. R. (2008). FABRIC DEFECT INSPECTION SYSTEM USING NEURAL NETWORK AND MICROCONTROLLER. Journal of Theoretical & Applied Information Technology, 4(7).

[48] Tsai, I. S., & Hu, M. C. (1996). Automatic inspection of fabric defects using an artificial neural network technique. Textile Research Journal, 66(7), 474-482.

[49] Tsai, I. S., Lin, C. H., & Lin, J. J. (1995). Applying an artificial neural network to pattern recognition in fabric defects. Textile Research Journal, 65(3), 123-130.

[50] Karayiannis, Y. A., Stojanovic, R., Mitropoulos, P., Koulamas, C., Stouraitis, T., Koubias, S., & Papadopoulos, G. (1999, September). Defect detection and classification on web textile fabric using multiresolution decomposition and neural networks. In ICECS'99. Proceedings of ICECS'99. 6th IEEE International Conference on Electronics, Circuits and Systems (Cat. No. 99EX357) (Vol. 2, pp. 765-768). IEEE.

[51] Habib, M., & Rokonuzzaman, M. (2011). Distinguishing Feature Selection for Fabric Defect Classification Using Neural Network. Journal of Multimedia, 6(5).

[52] Chen, J. J., & Xie, C. P. (2006). Fabric defect detection technique based on neural network. Journal of Textile Research, 27(4), 36.

[53] Chandra, J. K., Banerjee, P. K., & Datta, A. K. (2010). Neural network trained morphological processing for the detection of defects in woven fabric. The Journal of The Textile Institute, 101(8), 699-706.

[54] Zhang, Y. H., Yuen, C. W. M., Wong, W. K., & Kan, C. W. (2011). An intelligent model for detecting and classifying color-textured fabric defects using genetic algorithms and the Elman neural network. Textile Research Journal, 81(17), 1772-1787.

[55] Kumbhar, P., Mathpati, T., Kamaraddi, R., & Kshirsagar, N. (2016). Textile fabric defects detection and sorting using image processing. International Journal for Research Emerging Science and Technology, 3(3), 19-24.

[56] Si, X., Zheng, H., & Hu, X. (2012). Fabric Defect Detection Based on Regional Growing PCNN. Journal of Multimedia, 7(5).

[57] Li, Y., & Zhang, C. (2016). Automated vision system for fabric defect inspection using Gabor filters and PCNN. SpringerPlus, 5(1), 1-12.

[58] Bu, H. G., Huang, X. B., Wang, J., & Chen, X. (2010). Detection of fabric defects by auto-regressive spectral analysis and support vector data description. Textile Research Journal, 80(7), 579-589.

[59] Mingde, B., Zhigang, S., & Yesong, L. (2012). Textural fabric defect detection using adaptive quantized gray-level co-occurrence matrix and support vector description data. Information Technology Journal, 11(6), 673.

[60] Siegmund, D., Samartzidis, T., Fu, B., Braun, A., & Kuijper, A. (2017, June). Fiber defect detection of inhomogeneous voluminous textiles. In Mexican Conference on Pattern Recognition (pp. 278-287). Springer, Cham.

[61] Liu, Z., Liu, X., Li, C., Li, B., & Wang, B. (2018, April). Fabric defect detection based on faster R-CNN. In Ninth International Conference on Graphic and Image Processing (ICGIP 2017) (Vol. 10615, p. 106150A). International Society for Optics and Photonics.

[62] Wei, B., Hao, K., Tang, X. S., & Ren, L. (2018, June). Fabric defect detection based on faster RCNN. In international conference on artificial intelligence on textile and apparel (pp. 45-51). Springer, Cham.

[63] Zhang, M., Wu, J., Lin, H., Yuan, P., & Song, Y. (2017). The application of one-class classifier based on CNN in image defect detection. Procedia computer science, 114, 341-348.

[64] Jing, J., Dong, A., Li, P., & Zhang, K. (2017). Yarn-dyed fabric defect classification based on convolutional neural network. Optical Engineering, 56(9), 093104.

[65] Ouyang, W., Xu, B., Hou, J., & Yuan, X. (2019). Fabric defect detection using activation layer embedded convolutional neural network. IEEE Access, 7, 70130-70140.

[66] Sun, Y., & Long, H. R. (2009, March). Detection of weft knitting fabric defects based on windowed texture information and threshold segmentation by cnn. In 2009 International Conference on Digital Image Processing (pp. 292-296). IEEE.

[67] Zhao, Y., Hao, K., He, H., Tang, X., & Wei, B. (2020). A visual long-short-term memory based integrated CNN model for fabric defect image classification. Neurocomputing, 380, 259-270.

[68] Wen, Z., Zhao, Q., & Tong, L. (2020). CNN-based minor fabric defects detection. International Journal of Clothing Science and Technology.

[69] Liu, Z., Cui, J., Li, C., Ding, S., & Xu, Q. (2019, October). Real-time Fabric Defect Detection based on Lightweight Convolutional Neural Network. In Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition (pp. 122-127).

[70] Jing, J. F., & Ma, H. (2019, May). Yarn-dyed fabric defect detection based on convolutional neural network. In Tenth International Conference on Graphics and Image Processing (ICGIP 2018) (Vol. 11069, p. 110693W). International Society for Optics and Photonics.

[71] Liu, Z., Wang, B., Li, C., Li, B., & Liu, X. (2017, November). Fabric Defect Detection Algorithm Based on Convolution Neural Network and Low-Rank Representation. In 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR) (pp. 465-470). IEEE.

[72] Wei, B., Hao, K., Tang, X. S., & Ding, Y. (2019). A new method using the convolutional neural network with compressive sensing for fabric defect classification based on small sample sizes. Textile Research Journal, 89(17), 3539-3555.

[73] Racki, D., Tomazevic, D., & Skocaj, D. (2018, March). A compact convolutional neural network for textured surface anomaly detection. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 1331-1339). IEEE.

[74] Jing, J. F., Ma, H., & Zhang, H. H. (2019). Automatic fabric defect detection using a deep convolutional neural network. Coloration Technology, 135(3), 213-223.

[75] Seker, A., Peker, K., Yüksek, A. D. E. (2016). Fabric defect detection using deep learning, in 24th Signal Processing and Communication Application Conference (SIU), Zonguldak.

[76] Sun, G., Zhou, Z., Gao, Y., Xu, Y., Xu, L., & Lin, S. (2019). A Fast Fabric Defect Detection Framework for Multi-Layer Convolutional Neural Network Based on Histogram Back-Projection. IEICE TRANSACTIONS on Information and Systems, 102(12), 2504-2514.

[77] Partner textile industry Tunisia (2013). [Online].
Website: http://www.partnertextile.com/.

[78] Rebhi, A., Abid, S., & Fnaiech, F. (2016, November). Fabric defect detection using local homogeneity and morphological image processing. In 2016 International Image Processing, Applications and Systems (IPAS) (pp. 1-5). IEEE.

[79] Li, Y., & Zhang, C. (2016). Automated vision system for fabric defect inspection using Gabor filters and PCNN. SpringerPlus, 5(1), 1-12.

[80] Zhang, H. W., Zhang, L. J., Li, P. F., & Gu, D. (2018, May). Yarn-dyed fabric defect detection with YOLOV2 based on deep convolution neural networks. In 2018 IEEE 7th data driven control and learning systems conference (DDCLS) (pp. 170-174). IEEE.

[81] Yan, H., Paynabar, K., & Shi, J. (2017). Anomaly detection in images with smooth background via smooth-sparse decomposition. Technometrics, 59(1), 102-114.

[82] Sayed, M. S. (2016, October). Robust fabric defect detection algorithm using entropy filtering and minimum error thresholding. In 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS) (pp. 1-4). IEEE.

[83] Li, P., Liang, J., Shen, X., Zhao, M., & Sui, L. (2019). Textile fabric defect detection based on low-rank representation. Multimedia Tools and Applications, 78(1), 99-124.

[84] Jiang, J., Jin, Z., Wang, B., Ma, L., & Cui, Y. (2020). A Sobel Operator Combined with Patch Statistics Algorithm for Fabric Defect Detection. KSII Transactions on Internet & Information Systems, 14(2).

[85] Liu, Z., Cui, J., Li, C., Ding, S., & Xu, Q. (2019, October). Real-time Fabric Defect Detection based on Lightweight Convolutional Neural Network. In Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition (pp. 122-127).

[86] Xuelang Manufacturing AI Challenge-Visual Computing Assists Good Product Inspection, 2018. [Online]
Website: https://tianchi.aliyun.com/competition/entrance/231666/information

[87] Zhao, Y., Hao, K., He, H., Tang, X., & Wei, B. (2020). A visual long-short-term memory based integrated CNN model for fabric defect image classification. Neurocomputing, 380, 259-270.

[88] Movellan, J. R. (2002). Tutorial on Gabor filters. Open Source Document.

[89] Bankman, I. (Ed.). (2008). Handbook of medical image processing and analysis. Chapter 15: Two-Dimensional Shape and Texture Quantification. Elsevier.

[90] Aitex Fabric Image Database, (2019). [Online].
Website: https://www.aitex.es/afid/