Bilkent University

Department of Computer Engineering

Senior Design Project

Prelude

Final Report

Group Members: Samet Özcan, Osman Orhan Uysal, Osman Burak İntişah, Mehmet Alper Karadağ, Ziya Erkoç

Supervisor: Uğur Güdükbay

Innovation Expert: Veysi İşler

Jury Members: Hamdi Dibeklioğlu, Shervin Rahimzadeh Arashloo

Final Design Report

Dec 27, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS492/1.

Table of Contents

1. Introduction

In the textile industry, the clothes produced in some factories are manually examined in a cloth inspection machine to detect anomalies. When an anomaly occurs, the personnel operating that machine is responsible for either fixing the error right away or marking the improper section so that it can be replaced later. Yet, since people need to scan the cloth flowing continuously, they might miss some anomalies leading to faulty clothes. It can be caused by fatigue and the limitations of human nature, therefore, it causes a decrease in the companies' product prices and loss of their reputation [1, 2, 3]. Some automated cloth inspection machines are developed to speed up and enhance the quality control process. They require small human intervention since anomaly detection is done by the machine itself. These machines use methods such as Fuzzy Logic, Artificial Neural Network (Uster Fabricscan), and sometimes deterministic methods [4]. Yet, these machines cost too much money since the machines are sold as a whole instead of just the automation part as an add-on. Because some companies - including the Company A (anonymized) that we contacted - do not want to purchase new machines but improve theirs, they request an automated fabric anomaly detection product that is compatible with every cloth inspection machine [3].

Therefore, we are proposing software that can detect anomalies in the fabric using Deep Learning and create both visual and textual anomaly reports. In addition, any manager in the company can view the detailed reports and images of faulty fabrics so that they can take actions such as changing their thread vendor or renewing the weaving machines. We also ship a data labeling tool so that the company can collect their data and train them. We developed the software by constantly communicating with the textile company and went through several iteration stages to finalize it.

2. Requirements Details

We are going to talk about the Functional and Nonfunctional Requirements of our project within this section.

2.1 Functional Requirements

Data Labeling

Users can:

- draw a rectangle with his/her mouse to specify the bounding box of the anomaly.
- pan, zoom in, and zoom out the image to draw the rectangle with finer granularity.
- select the error type from the list with the mouse or type the number of the error type with the keyboard.
- capture a photo by clicking a button or pressing a key.
- browse the captured photos
- view and delete previously labeled anomalies
- enter production number

Anomaly Detection

Users can:

- start and stop detection process
- view type of lastly detected anomalies along with the bounding box and type of the anomaly
- mark the defects for undetected images

Report/Archive

Users can:

- view the detailed report of the anomalies by filtering the type, date, and production number.
- view the images of the fabric with anomaly

- view the visualization of where the anomalies occurred in a batch of fabric in the form of a 2D map

2.2 Non-functional Requirements

2.2.1 Performance

- The software must be able to detect anomalies within at most 1 second to match the speed of the fabric inspection machine [3].

2.2.2 Accuracy

- The software must be able to detect anomalies with at least 70% accuracy which is the rate of accuracy for a human [4].

2.2.3 Reliability

- The software should not let any fabric pass uninspected in case of an anomaly due to a software error or slow detection speed.
- The software should not mark the defective region as flawless. That is, it should not produce many false negatives.

2.2.4 Compatibility

- The software will run on the Windows operating system. Video cameras that can record in high definition will be used to capture the flow of the fabric.

2.2.5 Scalability

- The software should be able to handle the input from at least the number of cloth inspection machines.

2.2.6 Security

- The software should not leak any confidential data including cloth images and training data.

2.3 Pseudo Requirements

- The software will be developed using Python.
- YoloV4 [5] will be selected as the Deep Learning algorithm to detect anomalies and Darknet implementation will be used.
- Github will be our version control software.
- The software will be developed by following the Object Oriented Programming paradigm.
- High-frequency line scan cameras will be needed for both data collection and defect detection [6] .
- The data collected from the factory will not be shared with third-parties.
- We will use Turkish and English for our user interface since the factory that this application will be realized in is in Turkey and all of its employees know Turkish.

3. Final Architecture and Design Details

We have decided to use Python Kivy framework [7] for our Data Collection Tool because of its ease of use. In addition, since using Python is the most logical option for the Deep Learning part of the project, it was logical to combine both parts of the project in the same language and application to eliminate the communication between two parts. Moreover, we used PostgreSQL [8] as our database management system and Google Drive API [9] as our storage for fabric images. We used PostgreSQL python integration psycopg2 [10] to automate insertions and updates to the database. Finally, we have decided to use YOLO network [5] as our deep learning architecture. Its fast and accurate detection rates were the main reasons for our decision.

3.1 Overview

We will describe the architecture and design under three main sections: Data Collection Tool, Deep Learning and Database. Following image illustrates our system.

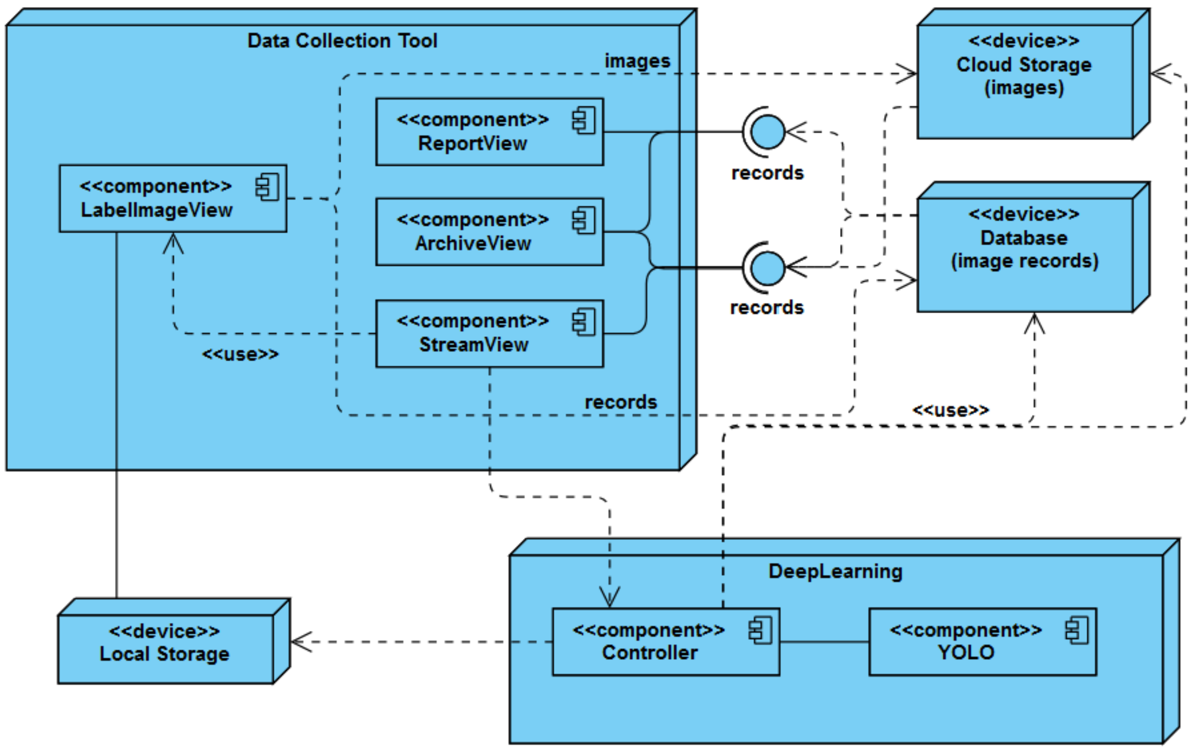


Figure 1: Component diagram

3.2 Data Collection Tool

The main purpose of the data collection tool is to provide an image labeling interface for our needs. Despite the name suggests, collecting and labeling images is not the only functionality of Data Collection Tool. It also has the functionalities such as presenting statistical information about fabrics via ReportView, displaying saved images according to production numbers via ArchiveView and monitoring current fabric batch as a stream via StreamView . We decided to include these statistical and monitoring functionalities in the same tool in order to keep the application compact so that the end user wouldn't have to switch between applications.

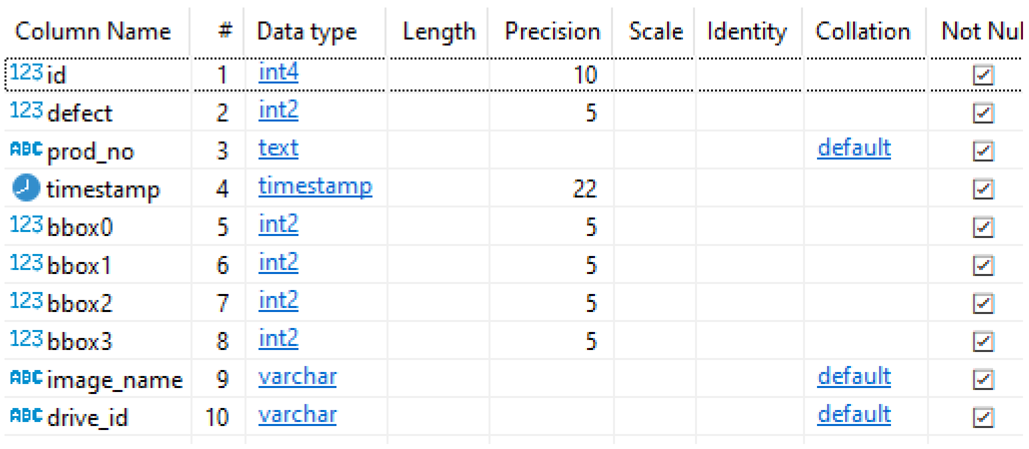
Moreover, we use a database to keep the records of defected images, defect entries and various statistics which are used to generate reports.

Furthermore, since the company requested to keep the collected data inside, we store the labeled data in a local storage. However, we upload the defects that are found by our model to a cloud storage for easy access by company authorities.

3.3 Deep Learning

After investigating several methods for object detection and anomaly detection, we decided on using the YOLOv4 algorithm [5]. We used the YOLO architecture as is with our data. In order to format the data, regulate the training of the model and prediction of incoming images we implemented a controller class through which we can access the newly collected data on the local storage, update the database and cloud storage.

3.4 Database



Column Name	#	Data type	Length	Precision	Scale	Identity	Collation	Not Nul
id	1	int4		10				✓
defect	2	int2		5				✓
prod_no	3	text					default	✓
timestamp	4	timestamp		22				✓
bbox0	5	int2		5				✓
bbox1	6	int2		5				✓
bbox2	7	int2		5				✓
bbox3	8	int2		5				✓
image_name	9	varchar					default	✓
drive_id	10	varchar					default	✓

Figure 2: Database scheme

We have used a database to store information about the defects detected by the Deep Learning algorithm. We fetch these data from Report and Archive pages which are explained in the manual in detail. We will talk about each of the column:

- **id**: It is a unique ID assigned to each row by the database engine and incremented by one with each row.
- **defect:** type of the defect
- **prod_no**: production number of the defect
- **timestamp:** time when the defect has occurred
- **bbox0**: x-coordinate of the top-left corner of defect’s bounding box

- **bbox1**: y-coordinate of the top-left corner of defect's bounding box
- **bbox2**: width of defect's bounding box
- **bbox3**: height of defect's bounding box
- **image_name**: name of the defect's image
- **drive_id**: We upload the image of this defected fabric to the Google Drive and receive a unique id for that file on the Drive. Then, we save this ID to this table, so that we can download this image from the Drive using that ID.

4. Development / Implementation Details

4.1. Data Collection Tool

We have developed our Data Collection Tool as a desktop application, which provides a user-interface for workers to label the images, in Python using the Kivy framework . The framework provides convenient ways to layout Buttons, Labels, and Input Fields to the window. In addition, it provides a 2D canvas so that we can draw arbitrary shapes such as a rectangle [7]. During the development of this tool, we were in contact with people from a textile company and we have gone through several iteration cycles to make sure that we created what they needed.

After finalizing the tool, we had to ship it to the company in the form of a .exe file although our code is in Python. To this end, we have used pyinstaller command-line-interface to pack the Python project into an executable file along with all of the DLL files of both Python and the libraries we have used.

4.2. Training and Detection

To train our Deep Learning model, we have used the Command-Line-Interface of Darknet YoloV4 library. We first needed to compile the library using the CMake tool. As we wanted to train on GPU, we had to install and enable the CUDA framework of Nvidia [11].

We have built several Python scripts to upsample the data, convert bounding box formats between each other, and convert the mask data to bounding box data. Since our data size was limited, we have upsampled the data to see how it affects performance. The bounding box format generated by our Data Collection Tool is different from what YoloV4 accepts; therefore, we had a small subroutine to handle that. Finally, since we could not collect data from the textile company, we used a dataset we found on the internet [12]. However, the data was prepared for image segmentation; so; it included masks instead of bounding boxes. As our algorithm accepts bounding boxes, we had to run a small script to fix that problem.

Our detection tool is also developed as a desktop application using Python's Kivy library. For detection, we again used the Darknet YoloV4 library. Yet, for this time, we have generated a DLL file for the library and referred to it from Python code to run detection.

We are uploading the images to Drive storage and defect information to a relational database as the defects are detected. Since image uploading and database insertion are time consuming processes we do not want to overwhelm the main-thread with these operations. Therefore, we are creating a seperate thread for these operations so that as the detection continues the image uploading and database manipulation can take place concurrently. We are using the same approach when we show the users 2D map and defected images that are stored in the database. Since downloading images might take some time, in a separate thread we are appending images one by one as they are downloaded while the user can interact with already downloaded images.

5. Testing Details

5.1 Software Testing

We have tested our application manually on various Windows computers. We have applied both end-to-end tests and isolated tests for each component (i.e. data labelling, report generator, prediction and streaming). We also had a chance to get our software tested by people from the textile company. We even had our data labelling part of our software run on the computer next to the defect inspection machine. We could not attend these test sessions in-person but could monitor the status through a remote desktop connection application called AnyDesk [13].

5.2 Deep Learning Algorithm Experimental Results

We have conducted our experiments on a computer with AMD Ryzen 5 3600 processor, 16 GB RAM and NVIDIA RTX 2060 GPU (w/ 6 GB VRAM). First of all, the image below shows the validation-mAP (mean average precision) and training-loss curve during the training. Training is conducted with around ~500 data using a dataset found online [13]. Originally, the dataset contained masks of the defects but we converted them to bounding boxes. Half of the images were defected and the other half were defectless. We also enabled the upsampling option of the YoloV4 algorithm. We inferred that after around 2500 iterations, the network starts overfitting because the mAP value, which is calculated on the validation set, starts decreasing after that point. We have selected the weights when the mAP score is the highest which corresponds to around 2500th iteration. Since our dataset was small we could not have the algorithm differentiate between defect types. Therefore, we only wanted the algorithm to detect the anomalies without specifying its type.

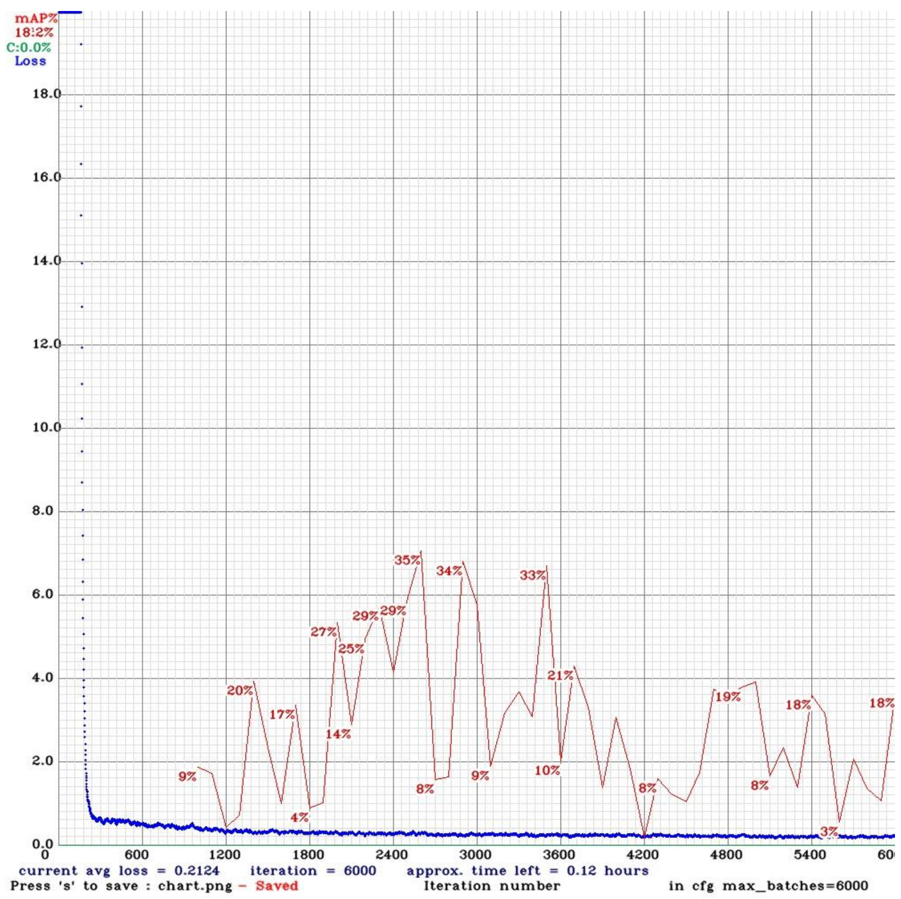


Figure 3: Loss and mAP curves plotted during training

We have conducted the testing using leather images [14] which is also a fabric. In the dataset, they have provided 5 classes of defects which are color, cut, fold, glue and poke anomalies. The dataset also includes defectless images. In Figure 4, example images for each defect type are provided.

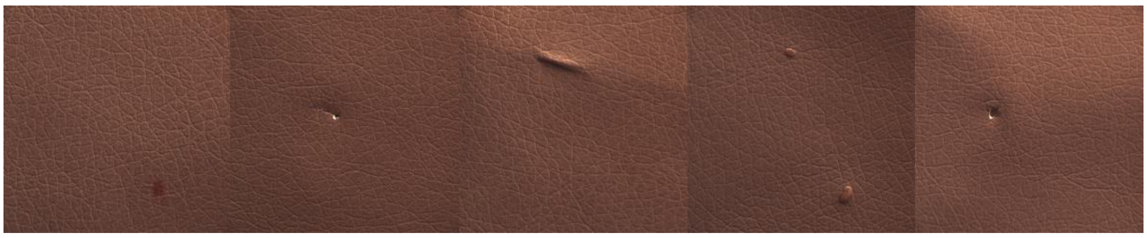


Figure 4: Example images for color, cut, fold, glue and poke anomalies from left-to-right

We used YOLO's test and mAP calculation mechanisms. Figure 5 shows example predictions with our network and weights at 2500th iteration. While doing the tests and calculating the mAP we have selected the IOU threshold as 0.3.

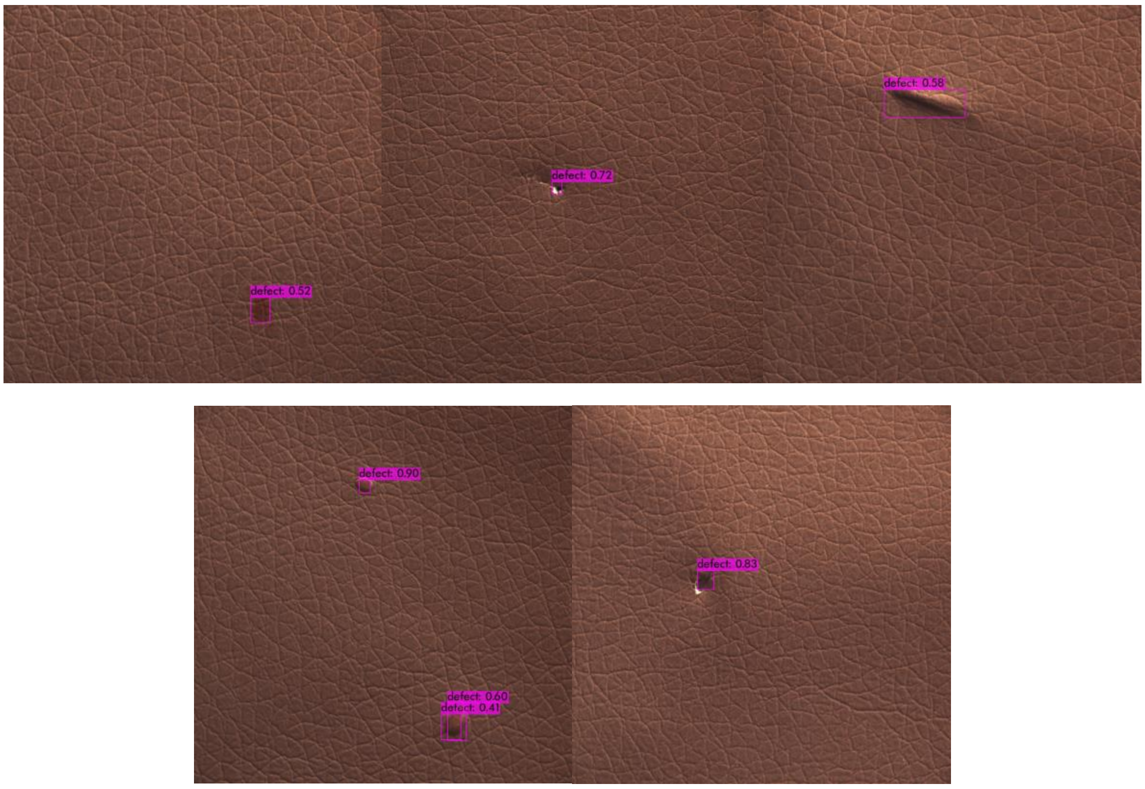


Figure 5: Our algorithm's predictions for color, cut, fold, glue and poke anomalies from left-to-right

Now, we shall provide the mAP, precision-recall-F1 and confusion matrix for all data and mAP and recall for each type in particular. First, we begin with overall data.

All Data

Precision = 0.61, Recall = 0.34, F1-score = 0.43, map@0.3 = %40

	Real Positive	**Real Negative**
Predicted Positive	**31**	**20**
Predicted Negative	**61**	**11**

Our overall confusion matrix suggests that our network has low recall which means we miss some defects. Actually, we have foreseen this situation and added a mechanism to our software so that in the early stages of the application of our project workers can label the defects missed by the network. Still, our precision is better than recall. That is, if an error is detected it is probably a real defect. We attribute these lower scores to the limited amount of data used during training. As seen in Figure 3, it overfits quite early.

Secondly, we shall show the mAP and recall results for each defect type. We did not provide precision or Real Negative part of the confusion matrix because in each experiment we only gave defected images. Our purpose with these tests is to see which anomalies are captured well and which ones are not.

Only Color Anomaly

Recall = 0.11, map@0.3 = %29.5

	Real Positive
Predicted Positive	**2**
Predicted Negative	**17**

Only Cut Anomaly

Recall = 0.26, map@0.3 = %46.7

	Real Positive
Predicted Positive	**5**
Predicted Negative	**14**

Only Fold Anomaly

Recall = 0.29, map@0.3 = %47.25

	Real Positive
Predicted Positive	**5**
Predicted Negative	**12**

Only Glue Anomaly

Recall = 0.58, map@0.3 = %55.23

	Real Positive
Predicted Positive	**11**
Predicted Negative	**8**

Only Poke Anomaly

Recall = 0.44, map@0.3 = %48

	Real Positive
Predicted Positive	**8**
Predicted Negative	**10**

Our network had the highest recall value in the glue anomaly and the lowest in the color anomaly. In fact, color anomaly is very hard to detect because they are relatively small. Also, cut anomalies were not detected with high recall. Their sizes are also small which makes the job of the network harder. Furthermore, the color and cut anomalies do not much disrupt the pattern of the fabric; therefore, the network incorrectly labels them as defectless.

Finally, we want to note that we have used totally different train and test datasets. In that way, we wanted to test the behaviour of our network with the worst case behaviour when the network sees completely different fabric types from what it has learned. We concluded that although the two datasets are completely different, we have reached a reasonable amount of scores with small data.

6. Maintenance Plan and Details

After we ship our product to the textile companies, we need to do some maintenance work to make sure that our product works smoothly.

6.1 Software Maintenance

Although we will ship the software without known bugs, as the people in the textile company begin using it new issues might emerge. There might be some buggy edge cases that we did not consider while testing but discovered by the end-users. To remedy this, we will collect such failures from the users and create patches fixing those issues.

6.2 Database/Storage Maintenance

We will need a database and a storage server to store data about defected fabrics. Therefore, after shipping the product, we need to make sure that those servers are up and running. To this end, we will regularly check those servers and in case of failure we will switch to another server while the previous servers are fixed.

6.3 Hardware Maintenance

Our product also requires properly working line-scan cameras to function perfectly. In case of its failure, we will need to contact the vendor so that the problem is fixed. Since this is a hardware component we will not be able to directly solve the problem without help from the experts.

7. Other Project Elements

In this section, consideration of various factors in engineering design, ethics and professional responsibilities, judgements and impacts to various contexts, teamwork details, meeting objectives, new knowledge acquired and applied are analyzed in respective order.

7.1 Consideration of Various Factors in Engineering Design

The table below presents analysis of the various factors that affected the development, design and usage of the application.

Factor	**Effect Level (Out of 10)**	**Effect**
Information Security	7	During the implementation of data collection tools and labeling mechanisms the storing place of these photos are considered. Since the data will be collected in the factory, and factory management did not want to share their data online we collected the data and saved them in local storage. However we needed cloud storage and a database so that employees will be able to inspect the defected fabric data from everywhere online which decreases the security a bit.
Social Factors	9	Since the application will be used by the factory workers and their age average is relatively high, the user interface and user experience are created as simple and easy as possible. At the end it is tested with some of the workers in the factory.
Factory Production Speed	8	Since our application will be used in the factory, the application running speed should be faster or at least same as the production speed than the

		factory production speed. Therefore, our implementation should be very efficient and fast. Therefore, multi-threading is used. Downloading, machine learning are processed in a different thread than the main thread of the application, which increases the speed of our application.

7.2 Ethics and Professional Responsibilities

Throughout the development process of our project we had to abide by several professional and ethical rules. First of all, we were collaborating with a textile company and we had weekly meetings at some period of time. Therefore, we attended the meetings on time to fulfill our professional responsibility of punctuality. Sometimes, some of our members could not attend due to other occupations. At those times, we have informed the company about that along with the excuse of not joining. Usually, at each meeting with the company, after deciding on what to do next, we have agreed on a deadline to complete the work. We could usually complete the functional requirements assigned by the company before the deadline. For instance, during a meeting people from the company wanted us to add a certain functionality to the Data Labelling Tool and we told them we could complete this in a week and we did so. We were also under the ethical obligation of not sharing the company related secrets and confidential discussions with other 3rd parties who would exploit that.

7.3 Judgements and Impacts to Various Contexts

In the following section, the judgements and its impacts on various contexts (global, economic, environmental, societal) and its impact levels are discussed.

Judgement Description:	Minimizing the extra cost of buying a new machine and employing workers.	
	Impact Level	**Impact Description**

Impact in Global Context	Low	No significant impact.
Impact in Economic Context	High	In the textile industry for gathering the same performance, factories have to change the machines available with expensive ones and most of them are reluctant to make such a change and continue with traditional ways. Therefore our application is an alternative for the machines available in the market which is more affordable. Additionally, in the long term for the factory it will be cheaper to use an automated approach than employing workers.
Impact in Environmental Context	Low	No significant impact.
Impact in Societal Context	High	Since Prelude meets the needs of the factory, the jobs of the worker might be negatively or positively affected from this transformation according to adapting themselves to the change.

Judgement Description:	Providing the best possible accuracy of finding defects.	
	Impact Level	**Impact Description**
Impact in Global Context	Low	No significant impact.
Impact in Economic Context	High	The low price with best accuracy would increase the sale and profit of the factory.
Impact in	Medium	Prelude reduces the waste of fabrics

Environmental Context		with finding the defects best possible accuracy.
Impact in Societal Context	High	Since Prelude meets the needs of the factory, the jobs of the worker might be negatively or positively affected from this transformation according to adapting themselves to the change.

7.4 Teamwork Details

This project is carried out in a collaborative team setting that includes assignments of individual and subgroup tasks according to task specifications and task relationships. Meetings are arranged and held during two week cycles as part of the team setting. The progress of current activities is addressed in these meetings and upcoming tasks are scheduled.

7.4.1 Contributing and functioning effectively on the team

Ziya Erkoç: Trained the YoloV4 algorithm and developed the prediction mechanism. Set-up the Drive API. Contributed to labelling mechanism.

Mehmet Alper Karadağ: Worked on the labelling tool and image collection. Set-up PostgreSQL database and integrated it into the application.

Samet Özcan: Prepared the Report subsection of the program where the user can specify the parameter values of queries as date, production number(s) and defect type(s) and display the results as report with export option.

Osman Burak İntişah: Worked mostly on frontend of the project. Created the streaming page of the application. Streaming the photos, showing the defects on minimap and on the image is done by using kivy library in python.

Osman Orhan Uysal: Worked on DataCollection tool and layouts. Set-up localization with English/Turkish language support.

All group members took part of bug fixing, improving code quality and report preparation. Even though every group member had a specific part to work on, everyone contributed to the other parts.

7.4.2 Helping creating a collaborative and inclusive environment

In order to communicate effectively, following tools are used;

- Online meetings are arranged through **Google Hangouts** at least every two weeks to manage the project plan and to ensure teamwork.
- **GitHub** is used in order to ensure the collaboration and version control of the project [15]. We worked on a private repository since this will be a commercial product at the end.
- **Whatsapp** is used in order to ensure the information flow between project members and factory managers .

The whole team tried to work together throughout the project. Even when Corona hits and working on a project which will be applied to the real world and data collection is needed is hard, the team successfully worked collaboratively.

7.4.3 Taking lead role and sharing leadership on the team

During the execution of the assignments, leadership moved between participants according to the needs of the mission. For instance, the project was divided into several parts and one team member was leader for one part. The person who is responsible for or leader of their part took the responsibility and led the team according to the knowledge he gained from his experiences and research. This leadership sharing continued during the project until all the assignments were completed. And by this approach, most of the team members are included in various parts of the project which is provided under the section 7.4.1.

7.4.4 Meeting objectives

The first task for our project was analyzing and acquiring the camera for which will be used for data collection and prediction. Analysis and research are completed and the best camera selected and proposed to the factory manager around 10th of March. However, because of the cost of the camera and equipment, the factory decided to use the camera on hand. According to the camera specifications a python application is developed for data collection at the beginning of April. However, after this application is developed Coronavirus Pandemic hit and the communication between factory and the team members is disrupted. We have still tried to find out ways to implement the camera and the software to the factory but the factory was not operational enough until September. Until September the team has worked according to the plan, machine learning algorithms are developed however we have used the data which is found online for training purposes [13]. Since we could not implement the whole system in the factory, we have continued to increase the accuracy until the end of October. Still, we are awaiting news from the factory to apply our project which the factory is not currently willing to do because the pandemic has affected it negatively.

Although the project plan was mostly satisfied, there were some delays occurred at several parts due to Covid-19 outbreak. Nevertheless, we ended up with a prototype of our project involving the specifications that we think are must. Finally, it can be said that our objectives are met and it is deliverable as a product.

7.5 New Knowledge Acquired and Applied

We have learned several new skills throughout the project. Firstly, we have learned how to use the Darknet Deep Learning Network and run the YoloV4 algorithm on it [5]. We referred to its Github documentation and followed the steps described there to train our dataset and run prediction routine. It has several parameters including subdivision size, batch size and input layer size. The documentation describes what each parameter is doing. Therefore, based on our needs we have adjusted those parameters. For instance, since the memory of the GPU we were using to do training had 6 GB of memory, we had to decrease batch size so that the batches would fit in the memory. We have also learned how to develop a Desktop Application with the Kivy framework of Python. We usually referred to the documentation of the framework to do the development [6]. At some points, we had to resort to Stack Overflow entries to resolve some issues. For instance, sometimes some aspects of the framework were not working as expected. To solve that, we have applied the workarounds suggested by the community. We also learned how to use Google Drive API in order to download from and upload to a cloud storage. To this end, we have used its official documentations.

8. Conclusion and Future Work

We have implemented many of the functionalities that we have planned. We are looking forward to collecting the data from the company, training it and running the deep learning model on the company with real image stream using our tool. We are also planning to productize our software based on the feedback we will receive from the textile company. We will need to decide how we are going to sell this product, whether we will include the line scan-cameras in the package, which part of the production line we will embed it and so on. Although currently we are planning to embed it into the last stage of the fabric manufacturing which is the quality control, people from the textile company suggested that this system could even be embedded right into the weaving machine. That way, as the machine weaves the fabric, our software can mark defects. This would indeed speed up the process because then at the quality control stage the fabric inspection machine can be fast forwarded to the faulty areas so that the people can fix the problem. As a result of this senior design project, we have developed a software by collaborating with an industrial entity to satisfy their needs. We believe that we have created a quite accurate prototype of what it would turn out to be in the future. We have enjoyed every aspect of the project and look forward to turning it into a full-fledged software ready to be sold to every textile company.

9. Glossary

Manager: Executives of the company

Worker: Textile Expert or Cloth Inspection Expert or Quality Control Worker

Data Collection Tool: The application we developed in order to mark/label the data, show detailed reports, populate a 2D defect map and do predictions.

YOLO: A Deep Learning algorithm which stands for You Only Look Once. [5]

Darknet: A neural network structure that runs YOLO algorithms. [5]

Kivy: GUI library for Python. [6]

mAP: Mean average precision. This indicates how good our model is at performing.

Recall: Sensitivity, fraction of retrieved instances among all relevant instances.

10. User Manual

Prelude is a software tool which aims to enhance the defect detection processes in fabric inspection by minimizing the human factor and resulting deficiencies originating from the insufficiency of the human eye and time-dependent decreasing concentration level. Advanced artificial intelligence detects the location of any defect during flow and warns the user via plain user interface. Continuous development of the algorithm is aimed by the inclusion of features for feeding the artificial intelligence with more images during usage, and statistical reports are ready to be prepared based on user specifications: time interval, defect type and production no.

Once the program is run the following buttons are displayed on the main screen for calling of these named functions and corresponding windows (Figure 2). Mark Defect, Inspect Fabric, View Report and Archive button opens up the pages described in the following sections. Our program is bilingual and can be viewed in Turkish or English based on the preferences. In the English version, "Türkçe" button is shown so that users can switch to Turkish. Similarly, in the Turkish version "English" button is shown. We support Turkish language because initially we are aiming to ship our product to Turkish textile companies.

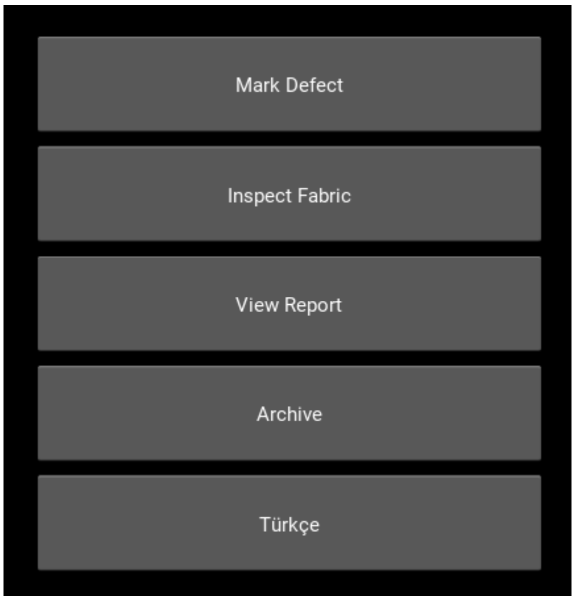
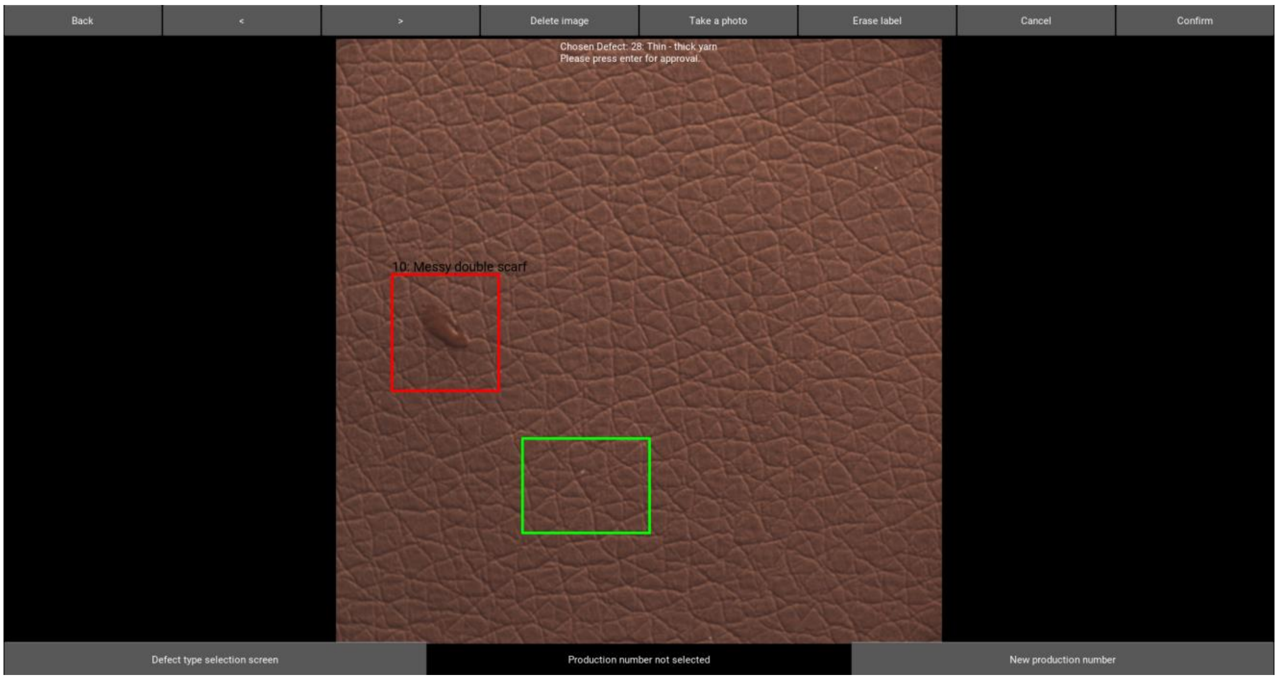


Figure 6: Main Menu

10.1 Mark Defect

This section is used mainly for data labelling. Once the user clicks on the button the following window is popped up (Figure 3);

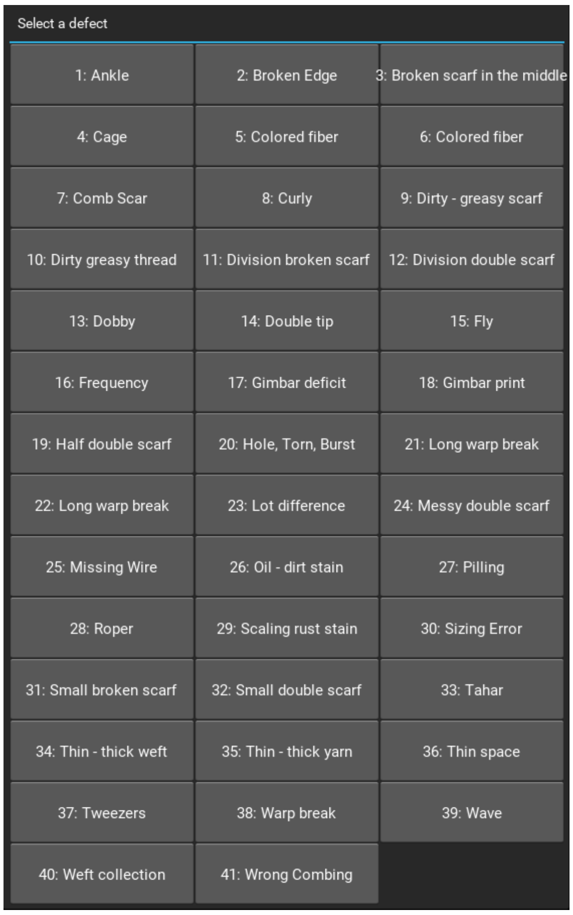


Chosen Defect: 28: Thin - thick yarn
Please press enter for approval.

Production number not selected

Figure 7: Defect Labeling/Marking page

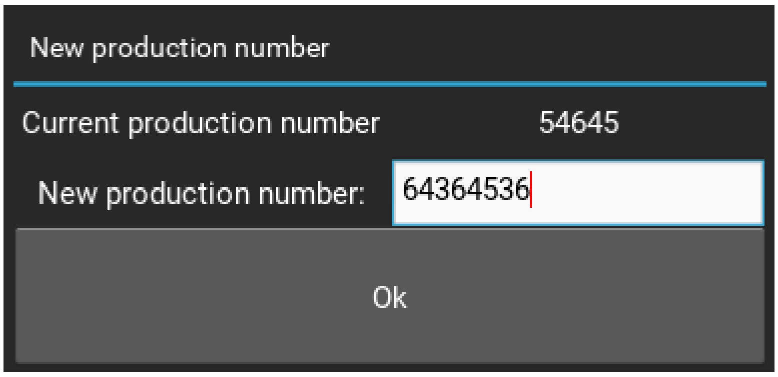
The user can move forward or backward between the photo records using the arrows and < and > buttons, delete the taken photo currently displayed by clicking on the **Delete Image** button, cancel a rectangle he drew by clicking on the **Cancel**. Furthermore, users can delete the previously drawn bounding box by clicking on it and pressing the DEL button on the keyboard or **Erase label** button on the top menu. Lastly, the user can leave the **Label Defect** window clicking on the **Back** button, but the content of the window is preserved all the time. The photo on the camera appears spontaneously once the user clicks on the **Take Picture** button on which the user can specify the location of defect by drawing a rectangle via the mouse as in the figure, enter the defect number with keyboard and press the Enter button on the keyboard or **Confirm** button on the screen finally, which can be done many times depending on the defect count on the currently displayed fabric. Alternatively, the user can click on the **Defect Type Selection Screen** to select the defect type from the list on the subsequently appearing pop-up as shown in the following image;


Select a defect

1: Ankle	2: Broken Edge	3: Broken scarf in the middle
4: Cage	5: Colored fiber	6: Colored fiber
7: Comb Scar	8: Curly	9: Dirty - greasy scarf
10: Dirty greasy thread	11: Division broken scarf	12: Division double scarf
13: Dobby	14: Double tip	15: Fly
16: Frequency	17: Gimbar deficit	18: Gimbar print
19: Half double scarf	20: Hole, Torn, Burst	21: Long warp break
22: Long warp break	23: Lot difference	24: Messy double scarf
25: Missing Wire	26: Oil - dirt stain	27: Pilling
28: Roper	29: Scaling rust stain	30: Sizing Error
31: Small broken scarf	32: Small double scarf	33: Tahar
34: Thin - thick weft	35: Thin - thick yarn	36: Thin space
37: Tweezers	38: Warp break	39: Wave
40: Weft collection	41: Wrong Combing	

Figure 8: Defect selection pop-up

Users can pan the image with the right mouse click and zoom in/out using the mouse wheel. In addition, by pressing the R key on the keyboard the user can reset the position and zoom level of the image. The user must enter a production number for a photo(s) and change it each time the photo corresponds to a newly being inspected fabric in the pop-up appearing after clicking on the **New Production Number** button as follows;


New production number

Current production number 54645

New production number: 64364536

Ok

Figure 9: Change production number pop-up

Once the user clicks on the **Ok** button, the pop-up is closed and the updated production number is displayed on the bottom-center of the main window.

10.2 Inspect Fabric

The most important functionality and main purpose of the software is the detection of defects in the currently flowing fabrics, which is conducted by the artificial intelligence on prediction level with sufficient accuracy and speed compared to the human-eye in addition to the very less amount of maintenance cost. Once the user clicks on the **Inspect Fabric** button the below window pops up where the real-time display of the fabric flow with the occurring defects with the coordinate information next to it -if exist- is at the center, and a minimap representing the entire fabric starting from the beginning up to the lastly inspected point is at the right side. The minimap enables the user to overview the entire inspection on the same fabric and find any defect distribution more in a straightforward and time-saving manner, where the crosses represent the proportionally exact location of that defect in the yellow board representing the fabric. The stream (and the flow after the possible application of the project) can be stopped clicking the **Stop** button and started again clicking the **Start** button. The **Mark Defect** button opens up a pop-up so that users can mark the defects missed by the Deep Learning algorithm as shown in Figure 7. Similar to the process in section 10.1, users can mark the defect. Lastly, the user can turn back to the Main Window by clicking the **Back** button after which the Stream window still continues to run. **Current production number** shows the lastly entered production number by clicking the **new production number** button as explained in the previous section. **Data Communication Status** field in the upper right corner shows if the images are uploaded to the storage or not.

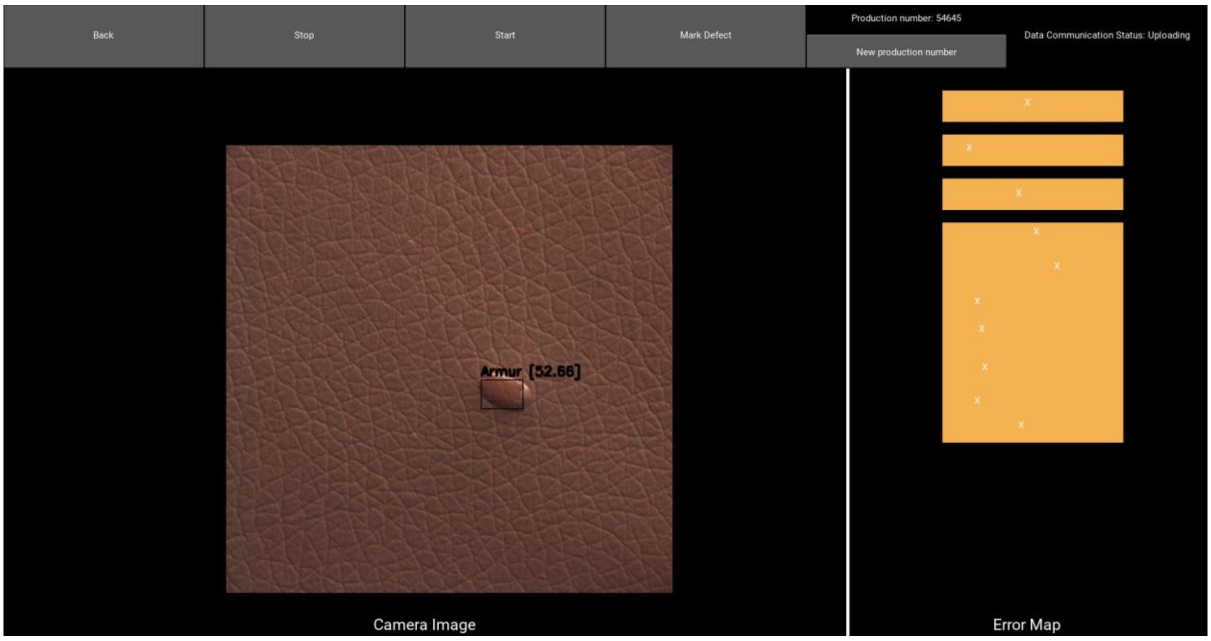


Figure 10: StreamView page

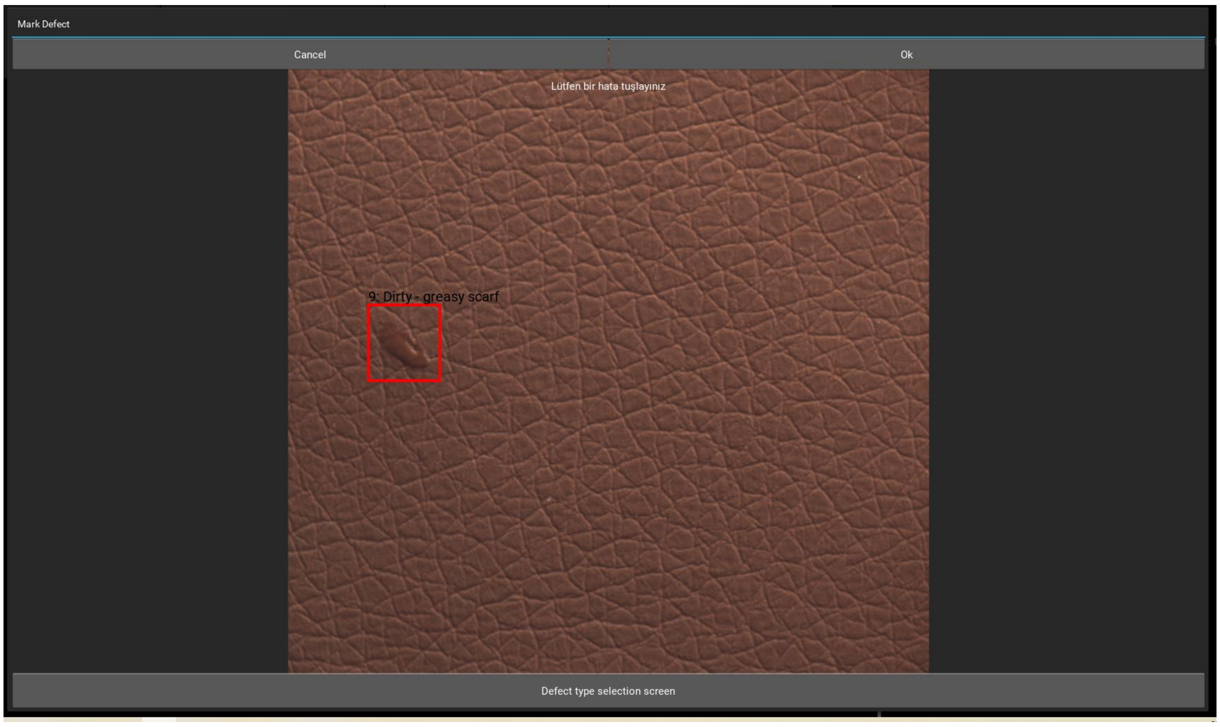


Figure 11: Mark defect pop-up

10.3 View Report

Even though there is no exact solution for the defects to occur during production of fabrics, still, there might be applied some approaches for reducing the amount of defects partially by the analysis of defect records. For instance, if a type of defect is faced in extraordinary amounts, actions such as checking the fabric production stages or some conditional factors (e.g. deficiency in the preservation of fabric material) can be taken. In addition, based on the cost of a small portion of defects which are not detected fabric selection for production can be arranged and related industrial actions can be taken. Depending on the success of such analysis the number of employees for handling defects and also the time loss during handling can be reduced for efficiency and profit.

There are two type of options for making queries to meet the pre-discussed needs of the company once the View Report button is pressed;

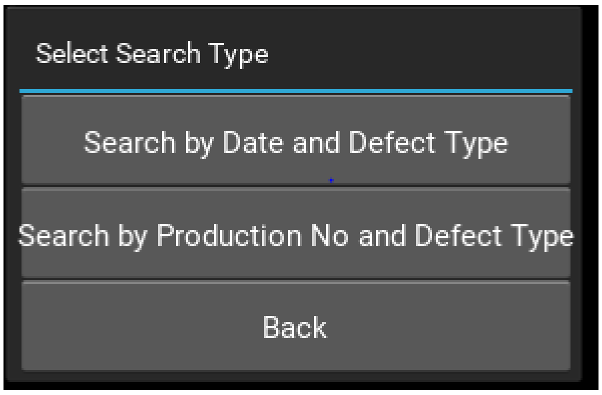

Select Search Type
Search by Date and Defect Type
Search by Production No and Defect Type
Back

Figure 12: Report query type selection page

The **Search by Date and Defect Type** option enables the user to specify the time interval and defect types for limiting the outcome of the search and being able to extract only needed/desired data whereas the **Search by Production Number and Defect Type** option provides the user to specify production no's of the fabric that s/he wants to extract information and also to limit the defect types similarly. The **Back** button closes the current window and pops the main menu window.

10.3.1 Querying with date and defect type

The user can specify the starting and ending date of the query via the calendar as followed;

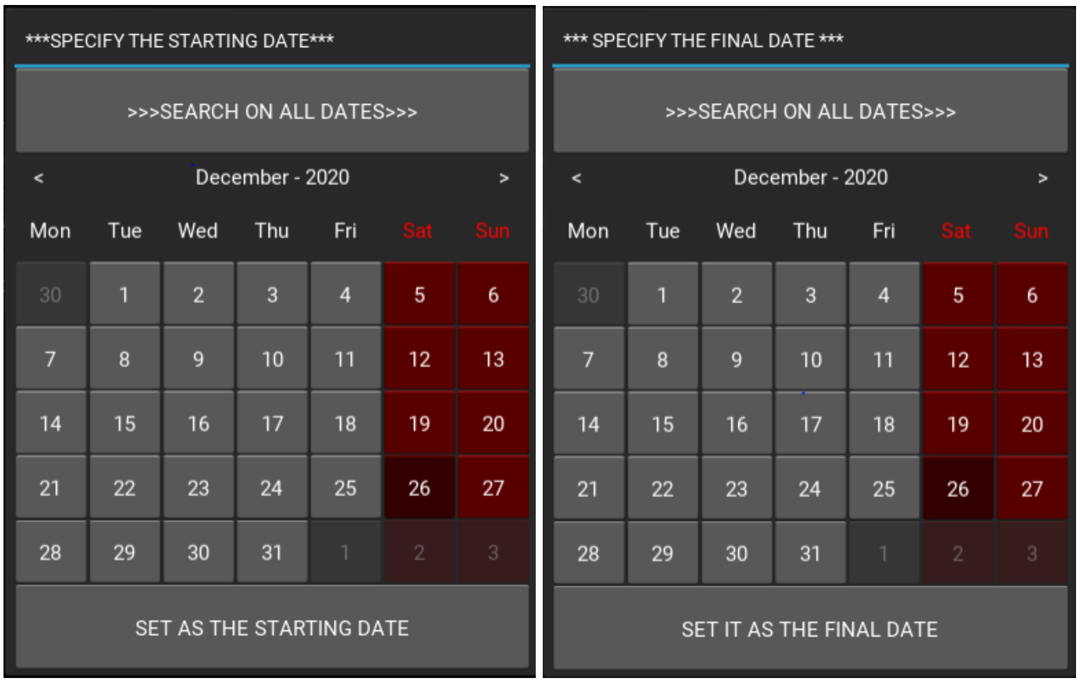


Figure 13: Selection of start and end dates of report page

The user presses the **Specify the starting date** button once decides on the starting date and selects on the calendar accordingly, and similar popup appears with the **Specify the final date** this time for ending time, or else the user can include all times and skip the date selection part any time by clicking on the >>>**SEARCH ON ALL DATES**>>> button. The month displayed can be shifted forward or backward via the arrows around the month - year specification.

Once the date selection part is completed defect type selection pop appears as;

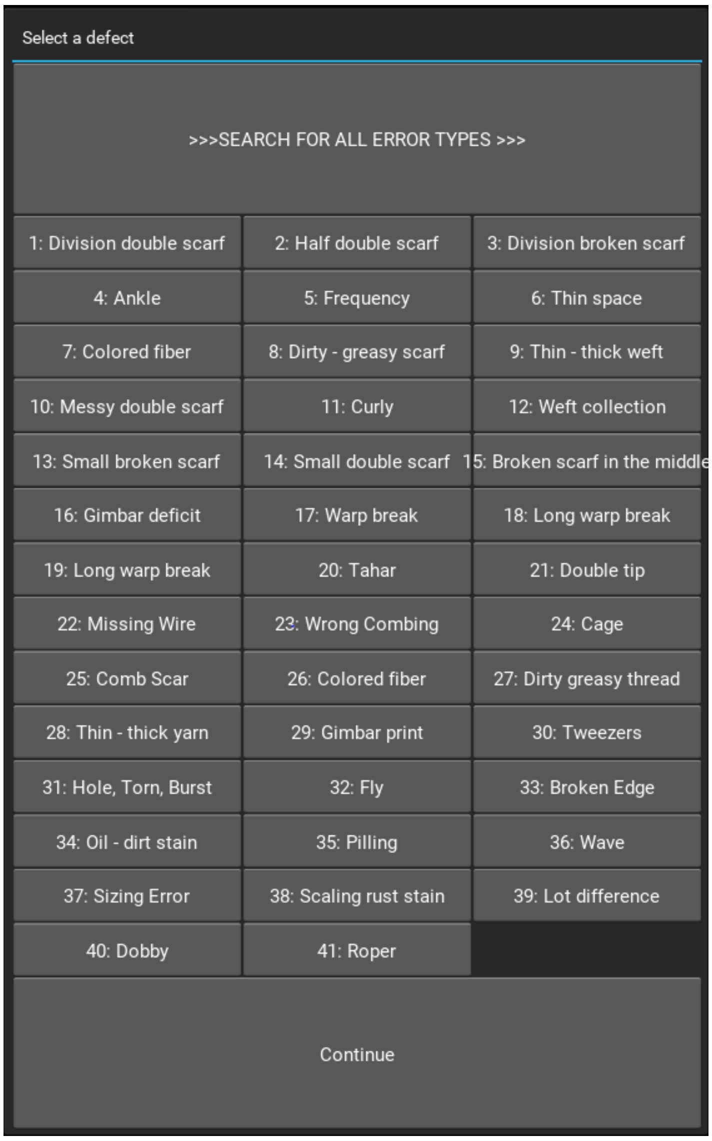


Figure 14: Defect type selection page

The user can click on the defect type(s) that s/he aims to include in the query one by one between the **Search in all defect types** button for inclusion of all fabric types and the **Next** button for querying, both of which directs the user to the results window.

10.3.2 Querying with production number and defect type

The user firstly specifies the production numbers of the fabrics that s/he wants to query by texting them into the white text box by putting either comma or blank between them. The user can add the production numbers in several times by pressing on the **Insert for search** button each time after entries, then or directly at once, the user clicks on the **Continue** button for passing into defect type selection popup which is exactly the same with the one in the query with date and defect type option in 10.3.1.

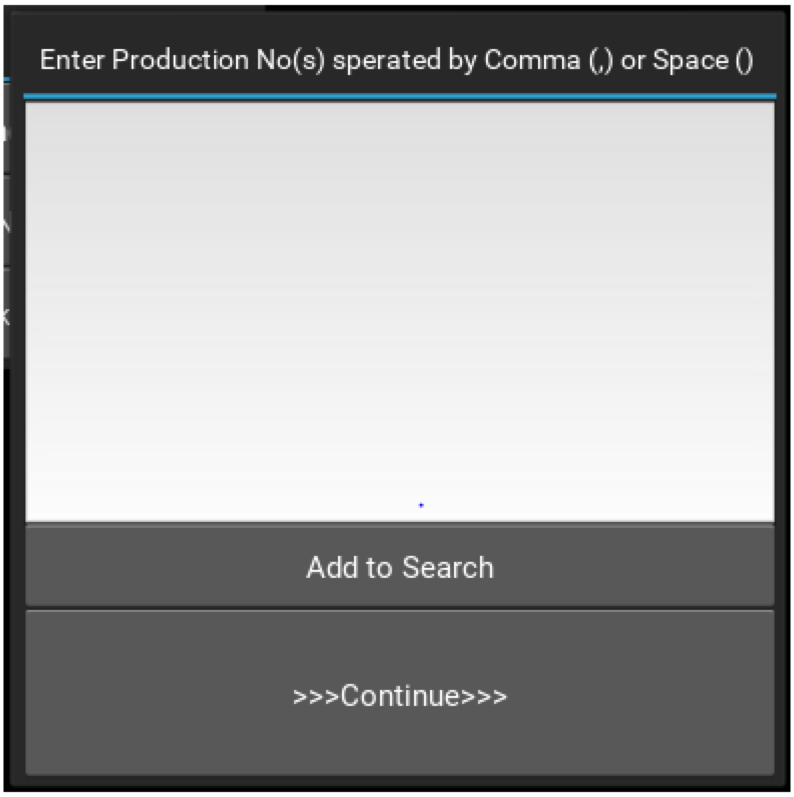
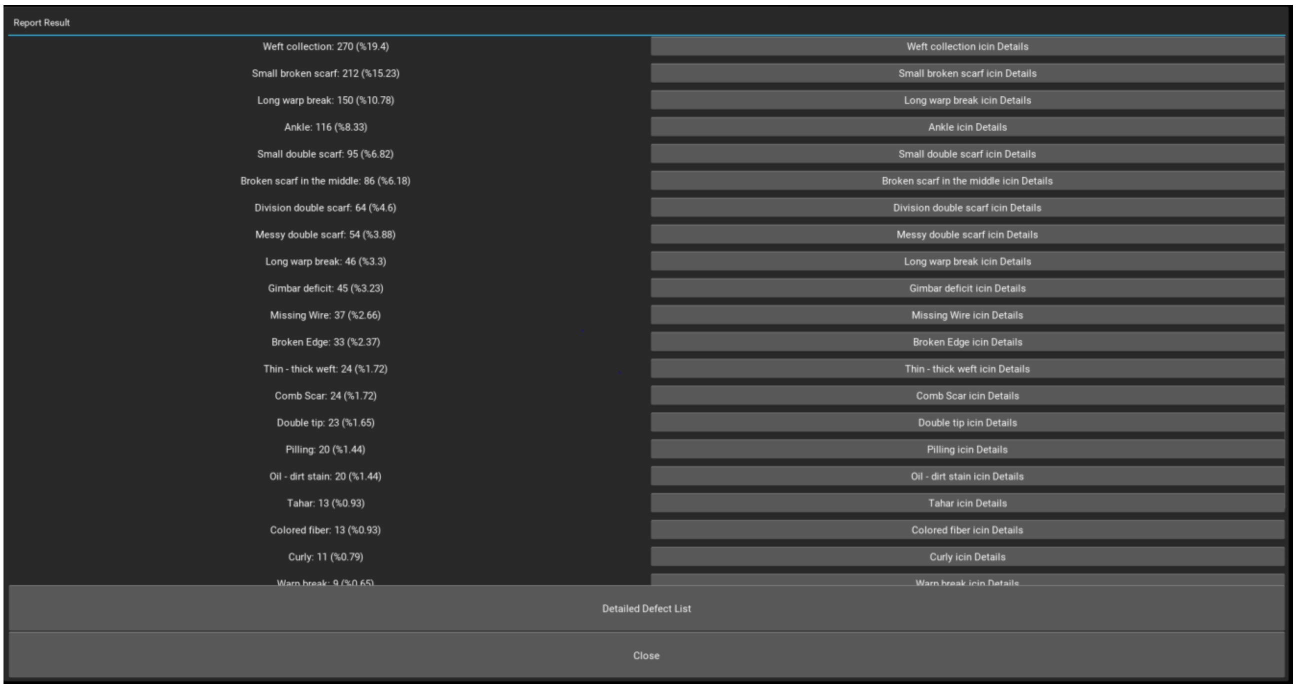


Figure 15: Production no selection page

10.3.3 Results

After the user determines the specification of the query, the results are displayed in a new window as below;



Report Result

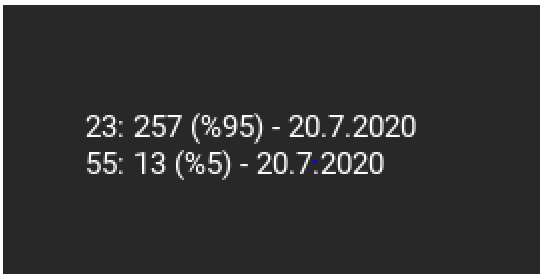
Weft collection: 270 (%19.4)	Weft collection icin Details
Small broken scarf: 212 (%15.23)	Small broken scarf icin Details
Long warp break: 150 (%10.78)	Long warp break icin Details
Ankle: 116 (%8.33)	Ankle icin Details
Small double scarf: 95 (%6.82)	Small double scarf icin Details
Broken scarf in the middle: 86 (%6.18)	Broken scarf in the middle icin Details
Division double scarf: 64 (%4.6)	Division double scarf icin Details
Messy double scarf: 54 (%3.88)	Messy double scarf icin Details
Long warp break: 46 (%3.3)	Long warp break icin Details
Gimbar deficit: 45 (%3.23)	Gimbar deficit icin Details
Missing Wire: 37 (%2.66)	Missing Wire icin Details
Broken Edge: 33 (%2.37)	Broken Edge icin Details
Thin - thick weft: 24 (%1.72)	Thin - thick weft icin Details
Comb Scar: 24 (%1.72)	Comb Scar icin Details
Double tip: 23 (%1.65)	Double tip icin Details
Pilling: 20 (%1.44)	Pilling icin Details
Oil - dirt stain: 20 (%1.44)	Oil - dirt stain icin Details
Tahar: 13 (%0.93)	Tahar icin Details
Colored fiber: 13 (%0.93)	Colored fiber icin Details
Curly: 11 (%0.79)	Curly icin Details
Warp break: 9 (%0.65)	Warp break icin Details

Detailed Defect List

Close

Figure 16: Resulting report based on the query

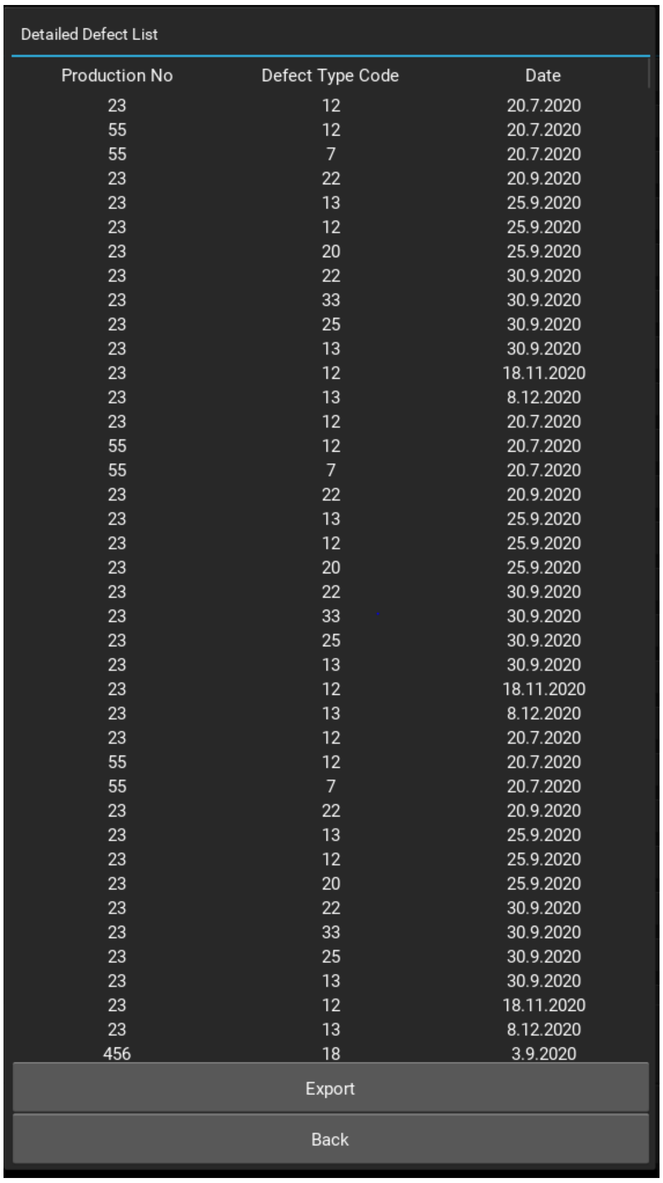
The defect types in the window are arranged in decreasing order based on the numbers they occured, in addition, the corresponding percentages relative to the sum of all occured defects are displayed next to them. For each defect type, the user can click on the corresponding button on the right side of the window and see more corresponding detailed results where defect counts with corresponding percentage and the inspection date are displayed for each fabric with unique production number are displayed as follows;



```
23: 257 (%95) - 20.7.2020
55: 13 (%5) - 20.7.2020
```

Figure 17: The number of defects in each fabric with percentage to overall of the query and with the start date of inspection

Analysing the details production number, the user turns back to previous page where s/he can view the all recorded results of the query row by row by pressing on the **Detailed Defect List** button as followed;



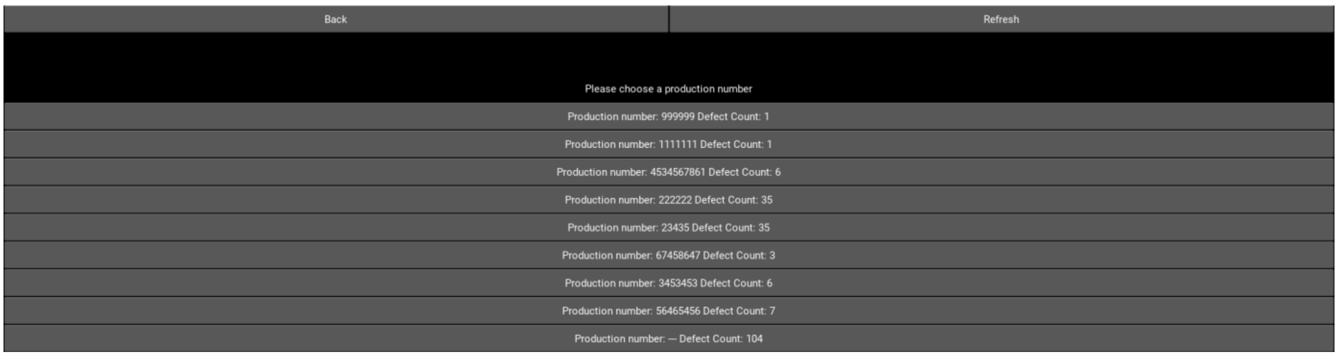
Detailed Defect List

Production No	Defect Type Code	Date
23	12	20.7.2020
55	12	20.7.2020
55	7	20.7.2020
23	22	20.9.2020
23	13	25.9.2020
23	12	25.9.2020
23	20	25.9.2020
23	22	30.9.2020
23	33	30.9.2020
23	25	30.9.2020
23	13	30.9.2020
23	12	18.11.2020
23	13	8.12.2020
23	12	20.7.2020
55	12	20.7.2020
55	7	20.7.2020
23	22	20.9.2020
23	13	25.9.2020
23	12	25.9.2020
23	20	25.9.2020
23	22	30.9.2020
23	33	30.9.2020
23	25	30.9.2020
23	13	30.9.2020
23	12	18.11.2020
23	13	8.12.2020
23	12	20.7.2020
55	12	20.7.2020
55	7	20.7.2020
23	22	20.9.2020
23	13	25.9.2020
23	12	25.9.2020
23	20	25.9.2020
23	22	30.9.2020
23	33	30.9.2020
23	25	30.9.2020
23	13	30.9.2020
23	12	18.11.2020
23	13	8.12.2020
456	18	3.9.2020

Figure 18: All the defect records for the query

By clicking on the **Export** button the user can export the list into a local directory in Excel file format in the similar fashion. Finally, the user can press the **Back** button to turn back to the results window where s/he can turn back into the query type selection window where s/he can either start a new query or turn back to the Main Prelude window as indicated at the beginning.

10.4 Archive



Please choose a production number

Production number: 999999 Defect Count: 1

Production number: 1111111 Defect Count: 1

Production number: 4534567861 Defect Count: 6

Production number: 222222 Defect Count: 35

Production number: 23435 Defect Count: 35

Production number: 67458647 Defect Count: 3

Production number: 3453453 Defect Count: 6

Production number: 56465456 Defect Count: 7

Production number: — Defect Count: 104

Figure 19: Fabric Defect Archive page

In this page, the user views Production No's and the number of defects associated with that production number. **Back** button returns the user to the main menu, the **Refresh** button fetches the latest update from the database. After clicking the button of a production number, the user can view the defect map and images of defected fabrics as seen in the below image.

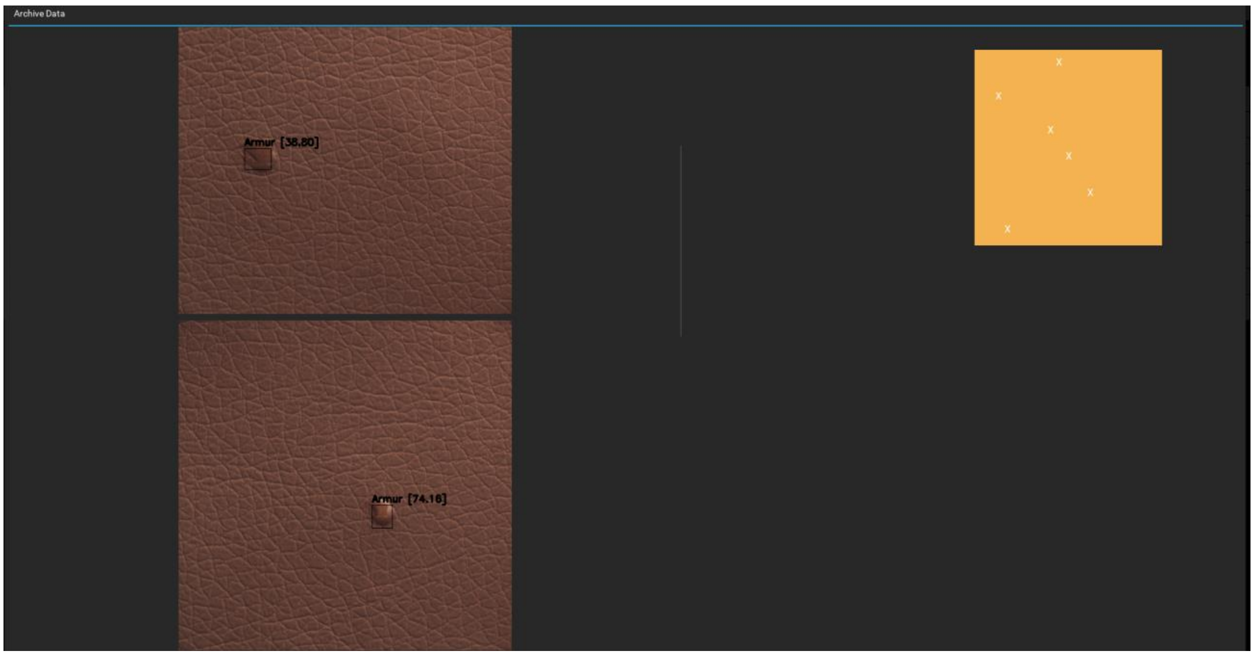


Figure 20: Defected fabrics and the corresponding 2D Map shown side-by-side

References

[1] M. S. Biradar, B. G. Sheeparmatti, P. M. Patil and S. Ganapati Naik, "Patterned Fabric Defect Detection Using Regular Band and Distance Matching Function," 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, 2017, pp. 1-6.

[2] K. Srinivasan, P. H. Dastor, P. Radhakrishnaihan, S. Jayaraman, "FDAS: A knowledge-based frame detection work for analysis of defects in woven textile structures", J. Text. Inst., vol. 83, no. 3, pp. 431-44 7, 1992.

[3] CEO of Textile Company A, Interview, Nov 2019

[4] Ö. Kısaoğlu, "Kumaş kalite kontrol sistemleri", Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 12(2), 233-241, 2006.

[5] "Darknet Convolutional Neural Network", https://github.com/pjreddie/darknet last accessed 26.12.2020

[6] Hanbay, K., Talu, M. and Özgüven, Ö., 2020. Fabric Defect Detection Systems And Methods—A Systematic Literature Review.

[7] "Kivy GUI Library", https://kivy.org/ last accessed 26.12.2020

[8] "PostgreSQL", https://www.postgresql.org/about/ last accessed 26.12.2020

[9] "Google Drive API", https://developers.google.com/drive last accessed 26.12.2020

[10] "Python PostgreSQL Tutorial Using Psycopg2", https://pynative.com/python-postgresql-tutorial/ last accessed 26.12.2020

[11] "Cuda Toolkit", https://developer.nvidia.com/cuda-toolkit last accessed 26.12.2020

[12] "AITEX FABRIC IMAGE DATABASE", https://www.aitex.es/afid/ last accessed 26.12.2020

[13] "AnyDesk", https://anydesk.com/en last accessed 26.12.2020

[14] P. Bergmann, M. Fauser, D. Sattlegger and C. Steger, "MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 9584-9592, doi: 10.1109/CVPR.2019.00982.

[15] “Github”, https://github.com/ last accessed 26.12.2020