T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

MOBILE GLASS SURFACE DEFECT DETECTION

SENA ÖZBELEN

SUPERVISOR
DR. YAKUP GENÇ

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

MOBILE GLASS SURFACE DEFECT DETECTION

SENA ÖZBELEN

SUPERVISOR
DR. YAKUP GENÇ

2024
GEBZE

GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 21/01/2024 by the following jury.

JURY

Member
(Supervisor) : Dr. Yakup GENÇ

Member : Prof. Yusuf Sinan AKGÜL

ABSTRACT

With the improvement of Artificial Intelligence (AI), real-world applications based on AI are being developed more. Since data can be accessed and collected easily, this creates an opportunity to utilize AI more. One of the ever-evolving subfields of AI is Computer Vision. Computer Vision is used to interpret visual resources. This also helps identify objects in these visual resources. There is a lot of ongoing research on this subfield. This project aims to detect defects on a given surface by utilizing the existing pre-trained models and datasets. It is a mobile-based project. Therefore, accuracy and latency are the crucial points of the project.

Keywords: object detection, computer vision, defect detection.

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
YOLO	:	You only look once
IOU	:	Intersection over Union

CONTENTS

LIST OF FIGURES

1. PROJECT DEFINITION

This project is developed for detecting defects on the given surface on a mobile platform. It basically shows the detected defects in a bounding box with its confidence value for the current camera preview. To do that, there is a need for a dataset. A dataset is used to train or fine-tune a deep learning model. For the model part, it utilizes a pre-trained model for object detection. To run the model more accurately, there is a need for preprocessing. Preprocessing has to solve the light imbalance problem. In darker places, it might be difficult to detect defects properly. Lastly, this project is built on a mobile platform. Therefore, the program needs to be efficient while running. Everything should be suitable for a mobile application including the model because the available object detection models are generally heavy. The main target surface was glass, but it was changed to metal since the dataset is based on metal as shown in Figure 1.1. However, it can detect defects on different surfaces if they are not too small and they are similar to the ones in the dataset.

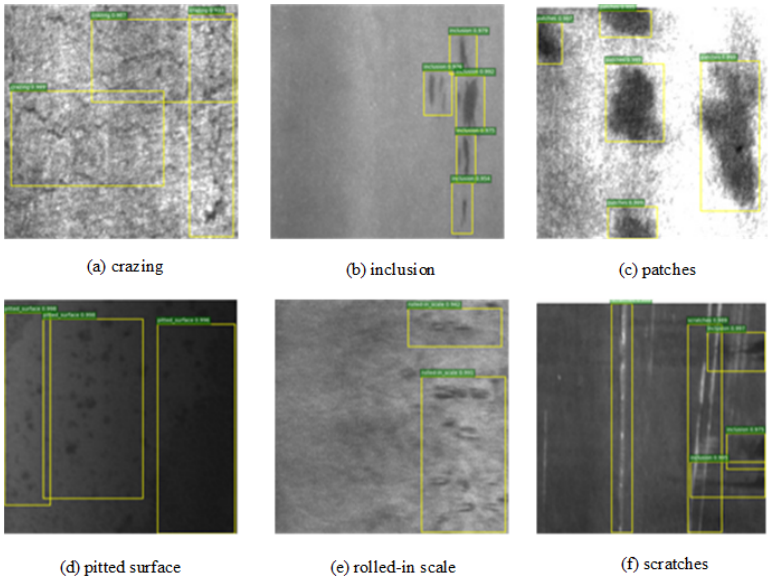


Figure 1.1: An example for a defect detection.[1]

2. PROJECT DETAILS

This project is handled on two sides, the model and the application. The flow diagram is shown in Figure 2.1

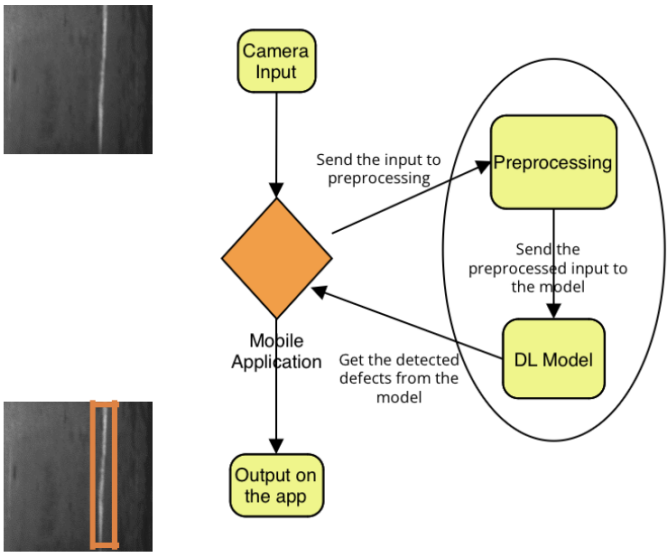


Figure 2.1: Flow diagram of the project.

2.1. The Model

2.1.1. The Dataset

This project uses a dataset that consists of metal surfaces with different types of defects. The dataset is ”NEU Metal Surface Dataset”[1]. This dataset includes six different defect types as shown in Figure 2.2 but this project uses only four of these types which are scratches, inclusions, patches, and rolled-in scale. The images are 200x200.

2.1.2. Preprocessing

The problem with the dataset is the light imbalance problem. Some images are too dark and defects are difficult to observe. This problem is valid for real-world cases as well. For this problem, there is an algorithm called ”Retinex”[2]. This algorithm

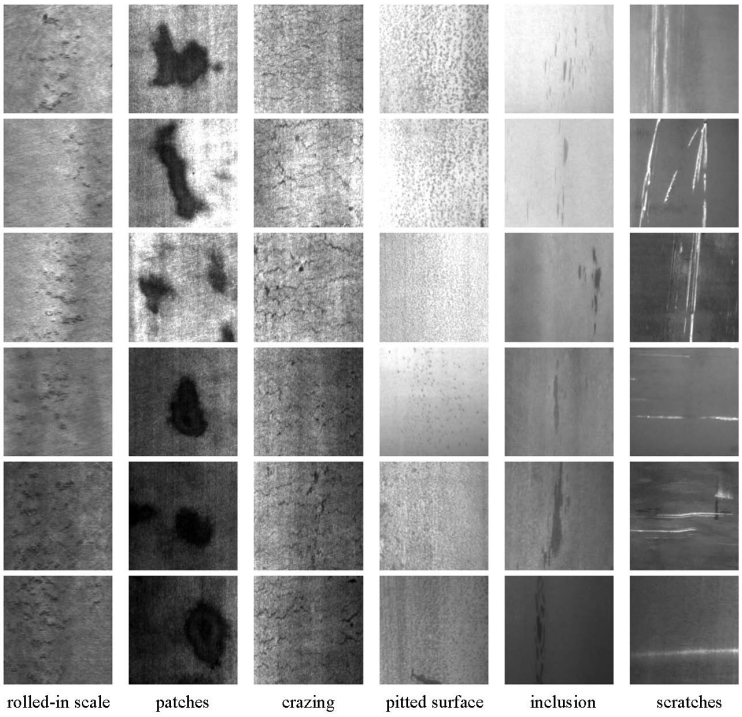
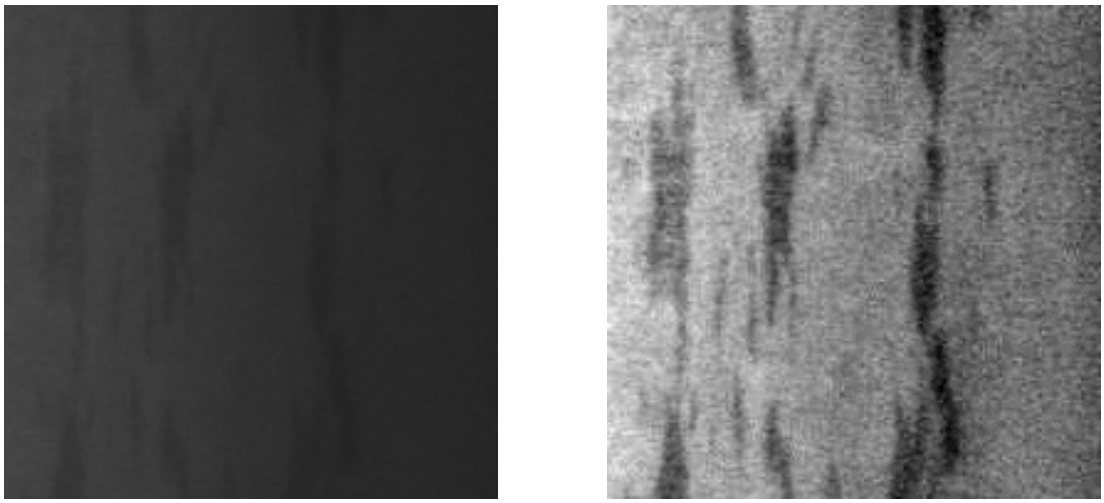


Figure 2.2: Examples from the dataset.[1]

is designed for low-light imbalance problems. It has two parts. Firstly, it takes the logarithm of the original values and the Gaussian blurred values. Then it subtracts these two results. Secondly, it normalizes the values. This gives a result as shown in Figure 2.3.

2.1.3. Choosing the Pre-trained Model

For the model side, this project utilizes available pre-trained models for object detection. It should be light and mobile-friendly. Also, it should achieve high accuracy for real-time input since the project takes the input from the phone's camera. Therefore, there are limited options for this project. The chosen model is YOLOv8 Nano. YOLO is a popular real-time object detection algorithm. It is basically used for the identification and localization of the objects. Localization is done by using bounding boxes for each detection. YOLO has different versions. The latest version at the end of 2023 is YOLOv8 [3]. YOLOv8 has different model sizes. For this project, YOLOv8 Nano was chosen because it is the lightest version.



(a) Original Image (b) Preprocessed Image

Figure 2.3: Preprocessing results for a low-light dataset image.

2.1.4. Fine-tuning

The preprocessed dataset and the chosen model are used for this part. It was fine-tuned on the Kaggle environment using the GPU available on the Kaggle. The epoch number is 200 and the batch size is 16.

2.2. Mobile Application

For the mobile application, Android Studio is used to build an Android application. To get the camera preview and analyze it, the library ”CameraX” is used. It basically takes the camera input in YUV format. In YUV format, a color is described as a Y component (luma) and two chroma components U and V. To feed the model, the Y component is used. This component is converted to a byte array and a simplified preprocessing is applied. This step includes a partial Retinex algorithm because of the efficiency of the application. It only takes the logarithm of the original values and normalizes the results. Then, it is converted to a bit map and it is reshaped since it is 200x200 but the model takes 224x224. The final result is converted into a byte buffer and given to the model. The output is stored in a 3D array. Its dimensions are the shape of the output of the model. After that, non max suppression is applied to remove the redundant detects. In non max suppression, IOU is used. Finally, it shows the boundaries for each detection with the confidence value if the value is above 0.5.

3. CONCLUSIONS

3.1. The Model

The current model gives the results as shown in Figure 3.1.

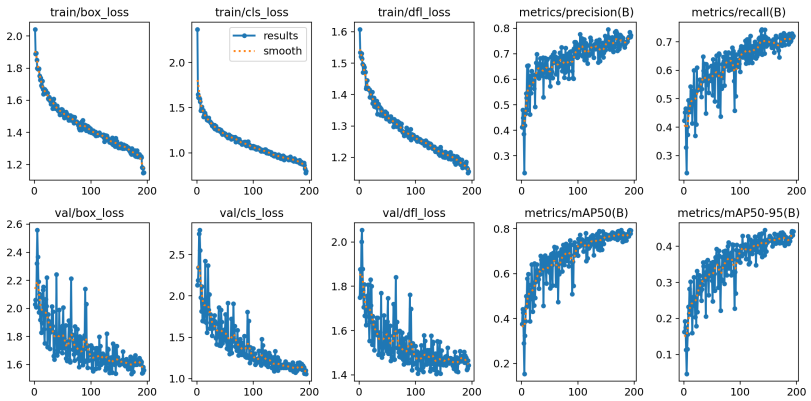


Figure 3.1: The results of the fine-tuning.

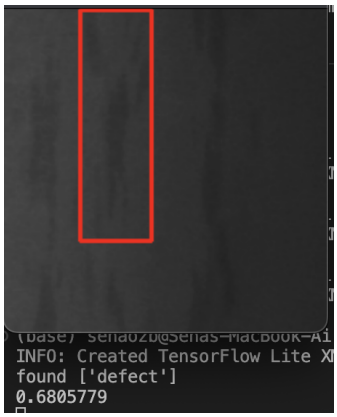
As for both recall and precision values, it is almost 0.75 which is the success criteria of the project. It is most probably higher than this value because there are some missing annotations in the dataset as shown in Figure 3.2 and this decreases the metric values of the model.

As for the effect of Retinex, there are small differences between the original image-based model and the preprocessed image-based model. In Figure 3.3, the confidence value of the detection is higher for the image with Retinex but in Figure 3.4, there is no big difference. However, Retinex gives better results in general so it is used.

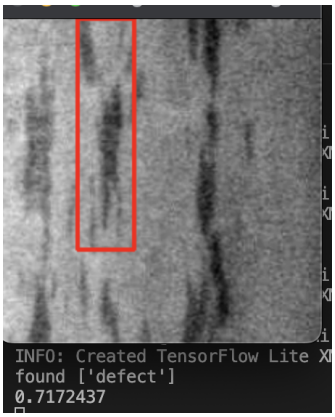
(a) The labels

(b) The predictions

Figure 3.2: The problem in the dataset.

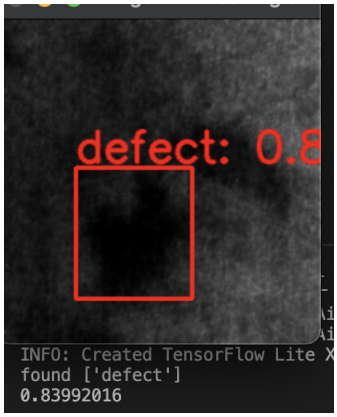


(a) Without Retinex



(b) With Retinex

Figure 3.3: An example for the effect of Retinex.



(a) Without Retinex

(b) With Retinex

Figure 3.4: Another example for the effect of Retinex.

3.2. Mobile Application

For the mobile application part, latency is an important factor. The latency is generally around 0.4 seconds. The success criteria say it should be below 0.5. Therefore, it performs a decent job. However, it uses too many resources. For some cases and conditions, this might produce some problems. The project is tested on an Android phone with 8 GB RAM, Mali-G68 MP5 GPU, and 4x 2.4 GHz ARM Cortex-A78 Main CPU as shown in Figure 3.5. It shows latency as milliseconds so it should be lower than 500 to meet the criteria.

Figure 3.5: The result on the application.

REFERENCES

[1] : http://faculty.neu.edu.cn/songkechen/zh_CN/zdylm/263270/list/

[2] : Liu,S.;Long,W.;He,L.;Li, Y.; Ding, W. Retinex-Based Fast Algorithm for Low-Light Image Enhancement. Entropy 2021, 23, 746. https://doi.org/10.3390/e23060746

[3] : https://docs.ultralytics.com