CHAPTER 3
RESEARCH METHODOLOGY

This chapter contains the steps that will be taken in working on this project. Starting from entering the dataset into the python notebook folder to the final stage of the research. The steps are as follows:

3.1. Put the dataset into the python notebook project folder

The datasets used in this research are haarcascade_frontalface_default.xml and haarcascade_eye.xml files. Here is the process of getting the haarcascade dataset into the python notebook project folder:

1. The first step is to retrieve the dataset. The dataset used in this project is taken from Github[1], and to run this project the dataset must be entered into the python notebook project folder so that the dataset can function to detect a person's face or eyes.
2. The first dataset is the Haarcascade_frontalface_default.xml file which contains code to detect faces such as using a subwindow with 24x24 dimensions, has a maximum number of 211, has 25 stage numbers, has a process threshold for face detection, has an internal node value, and has a leaf value. The haarcascade_frontalface_default.xml file also contains classification steps that go through the num and tree stages, in the tree there is one node that contains the haar (rects) feature. The first and second column numbers in the box indicate the position of the pixel being classified, while the third and fourth column numbers indicate the width and height of the feature, and for the last column number, the constant is multiplied to each side of the feature.
3. Meanwhile, the contents of the haarcascade_eye.xml file also contain codes to detect eyes such as using a 20x20 subwindow, having 93 weak counts at most, having 24 stage numbers, having a threshold value for eye detection, and having an internal node value. The haarcascade_eye.xml file also contains a classification step that goes through the numand tree; in the tree there is one node that contains the haar (rects) feature. Each number in rects serves the same purpose as the haarcascade_frontalface_default.xml file. The purpose of the rects for the first and

1 https://github.com/opencv/opencv/tree/master/data/haarcascades

second column numbers is to indicate the position of the pixel being classified, while the third and fourth column numbers indicate the width and height of the feature, and for the last column number, which serves to multiply the constants on each side of the feature.

Below is an image of the contents of the haarcascade_frontalface_default.xml dataset and the haarcascade_eye.xml dataset used in this project:

```
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          <leafValues>
            2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
```

Gambar 3.1: Dataset Haarcascade_frontalface_default.xml

```
<features>
  <_>
    <rects>
      <_>
        6 4 12 9 -1.</_>
      <_>
        6 7 12 3 3.</ ></rects></ >
```

Gambar 3.2: Dataset Haarcascade_frontalface_default.xml rects

```
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>20</height>
  <width>20</width>
  <stageParams>
    <maxWeakCount>93</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>24</stageNum>
  <stages>
    <_>
      <maxWeakCount>6</maxWeakCount>
      <stageThreshold>-1.4562760591506958e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 1.2963959574699402e-01</internalNodes>
          <leafValues>
            -7.7304208278656006e-01 6.8350148200988770e-01</leafValues></_>
```

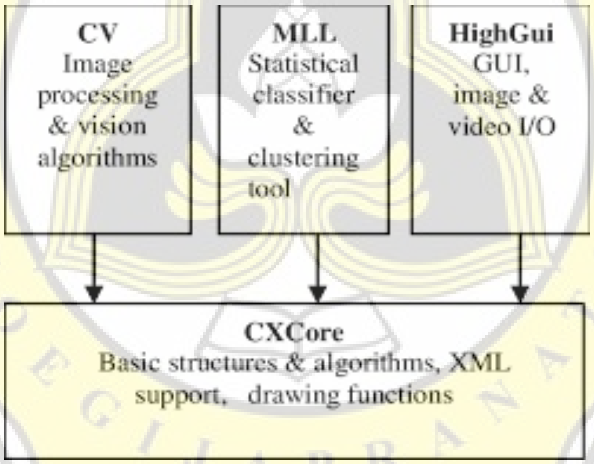
Gambar 3.3: Dataset Haarcascade_eye.xml

```
<features>
  <_>
    <rects>
      <_>
        0 8 20 12 -1.</_>
      <_>
        0 14 20 6 2.</_></rects></_>
```

Gambar 3.4: Dataset Haarcascade_eye.xml rects

3.2. Importing CV2

Before importing CV2, it is necessary to install the OpenCV library on the python notebook used in this project. OpenCV is used to process data in the form of images and videos to read all the information. OpenCV presents the haarcascade algorithm data source that is in the OpenCV data. In this project, face detection uses CV2, and a cascade classifier that reads the xml dataset file according to its use. The haarcascade_frontalface_default.xml and haarcascade_eye.xml datasets are implemented using OpenCV.



Gambar 3.5: OpenCV Basic Structure

Figure 3.5 contains the basic structure of OpenCV. The structure of OpenCV is broadly divided into five main components, as shown above:

1. The CV component contains basic image processing and advanced computer vision algorithms.
2. MLL is a machine learning library, which includes statistical classifier and clustering tools.

3. High GUI contains routines and I/O functions for storing and reading videos and images.
4. CXCORE contains the basic data structure and content.
5. CV Aux contains about unused areas and experimental algorithms.

```
import cv2

faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
eyeCascade = cv2.CascadeClassifier("haarcascade_eye.xml")
```

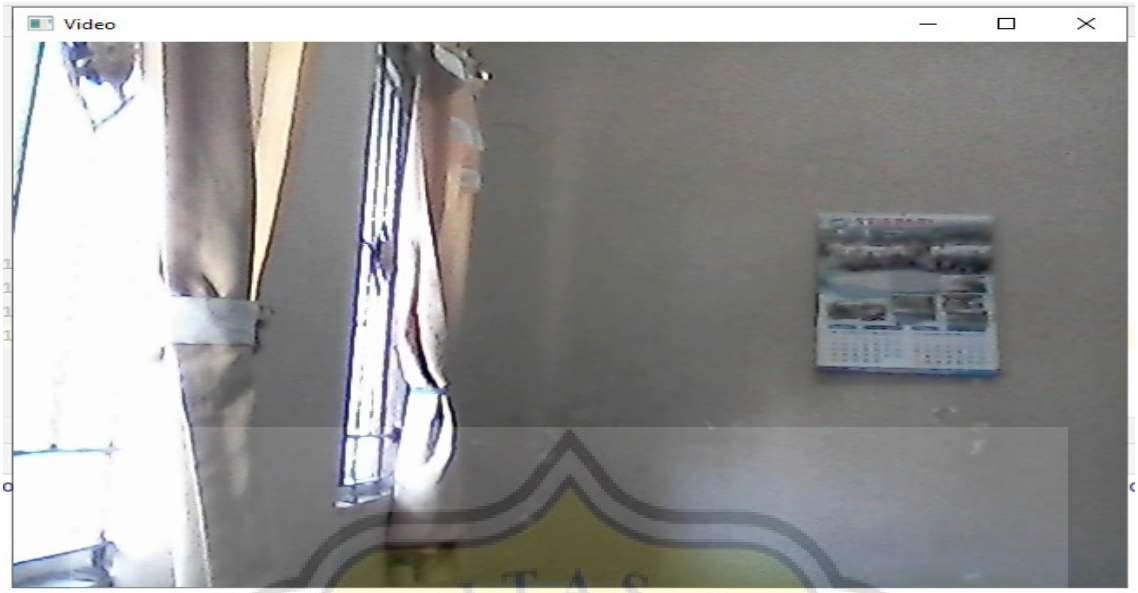
Gambar 3.6: Dataset Implementation On OpenCV

Figure 3.6 contains the implementation of the haarcascade classifier dataset which will be processed using CV2. The haarcascade classifier dataset used consists of two files, including the following:

1. Haarcascade_frontalface_default.xml which is processed using CV2 as face detection and cascade classification to read the xml dataset file according to its use. In the haarcascade dataset, it is stored in the faceCascade variable.
2. Haarcascade_eye.xml which is processed using CV2 as eye detection and cascade classification to read the xml dataset file according to its use. In the haarcascade dataset, it is stored in the eyeCascade variable.

3.3. Turn on the webcam

To detect faces and eyes by processing a dataset of xml files using a cascade classifier, I used the laptop's built-in webcam as well as live video connected to the program. Webcams also make it easy to process messages processed by the cascade classifier against xml file datasets quickly.



Gambar 3.7: The Process Of Turning On The Webcam

3.4. Change each frame to gray

Live video on the webcam camera is converted every frame into grayscale, so that the face and eye variables that are processed using the cascade classifier will be converted to grayscale.

3.5. Implementation of algorithms for face and eye detection

In detecting faces in video, each frame will be processed directly on CV2, and the cascade classifier functions to read the haarcascade_frontalface_default.xml dataset quickly and accurately so that it can detect images of objects on faces. And by providing the code cv2.rectangle to provide a box image around the face area, and cv2.putText code to provide a description of the face above the box image. As for eye detection, each frame will be processed on CV2, and the cascade classifier functions to read the haarcascade_eye.xml dataset file so that it can detect images of objects in the eye. And by providing the code cv2.rectangle to provide a box image around the eye area, and cv2.putText code to provide an eye description above the box image.

3.6. Implementation of the results of detecting someone using glasses

By detecting a person's face and eyes in a live video on the webcam, it can be tested whether someone is wearing glasses or not. This way, if face and eyes are detected, it ensures that a person is not wearing glasses, as well as the code cv2.rectangle to provide a square image around the red face area and the code cv2.putText to provide information that the person is not wearing glasses. Meanwhile, if only the face is detected while the eyes are not detected, then it is certain that the person is wearing glasses, as well as the cv2.rectangle code to provide an image of the box around the green and cv2 face area. putText code to provide information about a person wearing glasses. The project also counts the number of faces and eyes of a person in a live video.