Willburg Outdoor PTY(ltd.)

Smart Image Identifier

Version 1.0

Software User Manual

Project Team:
Stephen Swanepoel u11032091
Dian Veldsman u12081095

Contents

1 General Information . . . 2
1.1 System Overview . . . 2
1.2 System Configuration . . . 2
1.3 Installation . . . 2
1.3.1 Install Eclipse . . . 2
1.3.2 Importing the project . . . 3
1.3.3 Setting system environment variables . . . 3
2 Using The System . . . 6
2.1 OpenCV . . . 6
2.1.1 What is it? . . . 6
2.1.2 Why was it used? . . . 6
2.2 Classes . . . 6
2.2.1 PeopleDetect . . . 6
2.2.2 Writer . . . 7
2.2.3 FourCC . . . 7
2.2.4 Database . . . 7
2.3 Functions . . . 7
2.3.1 greyscale(BufferedImage image, Mat mat) . . . 7
2.3.2 equalization(BufferedImage image, Mat mat) . . . 7
2.3.3 enlargeImage(BufferedImage image, Mat mat) . . . 7
2.3.4 generateMat(BufferedImage image) . . . 8
2.3.5 generateImage(BufferedImage image, Mat mat, byte[] data) 8
2.3.6 processImage(String url) . . . 8
2.3.7 runTests(String url) . . . 8
2.3.8 readDatabase() . . . 8
2.3.9 getImage(byte[] base64String) . . . 8
3 Troubleshooting . . . 9

User Manual

1 General Information

1.1 System Overview

Smart Image Identifier is human recognition software written for Willburg to provide real-time processing of images on their server to assist in the improvement of security. The system is completely autonomous and thus has no user interaction, it is just system-to-system communications.

When an image is pushed to the server, Smart Image Identifier retrieves the image, processes it and sends an appropriate response back to the server

1.2 System Configuration

The system and its network is small and consists of a dummy database produced by the project owner and a server on which Smart Image Identifier will run. All processes will run on the server, it will also be responsible for the retrieval of images from the database and delivering a response based on the result.

1.3 Installation

1.3.1 Install Eclipse

The easiest way to build the dispatcher from source is using Eclipse Neon. Eclipse Neon is a development environment and is open-source software. The latest version of this software can be obtained at https://www.eclipse.org/downloads/. After the download is completed, install the software using the instructions provided by the installer.

1.3.2 Importing the project

In Eclipse, click File, Import, General, finally, select, Existing Projects into Workspace and click Next.

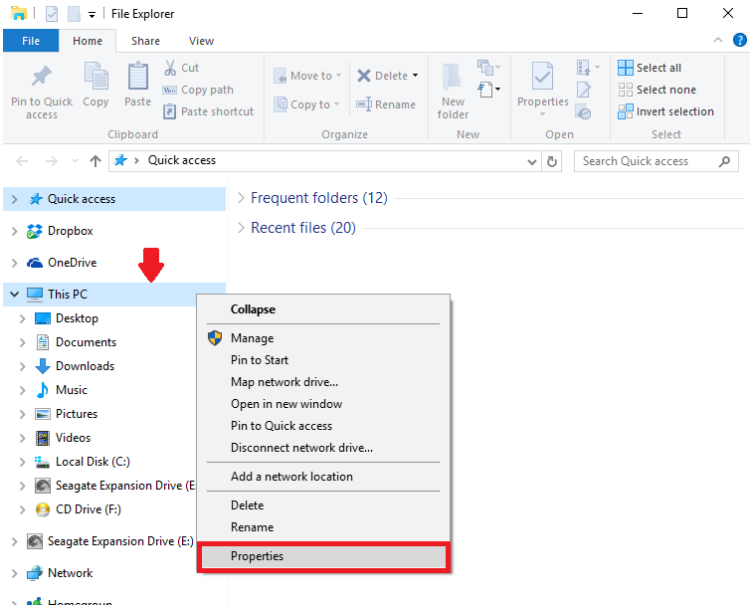
Select root directory, click Browse, locate the given project folder, finally, select Finish.

1.3.3 Setting system environment variables

Setting of environment variables is necessary for your system to detect the OpenCV framework which was used for developing Smart Image Identifier.

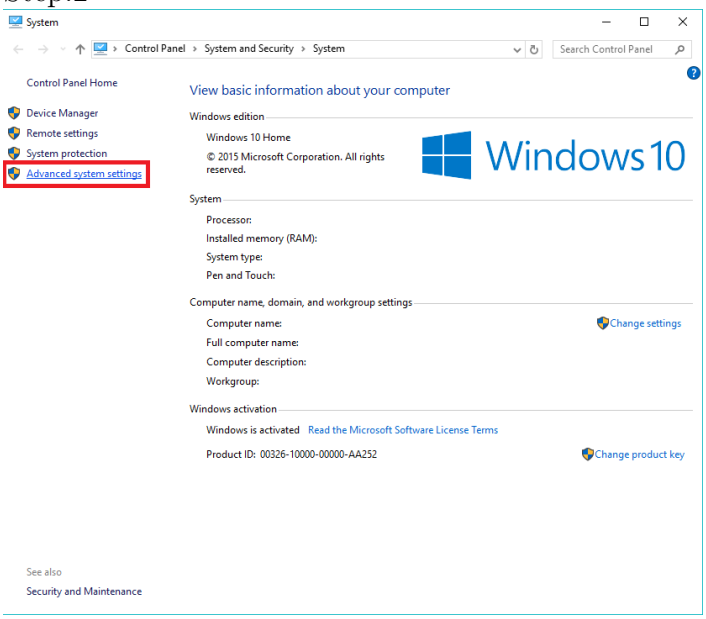
Lets get started

- Step:1



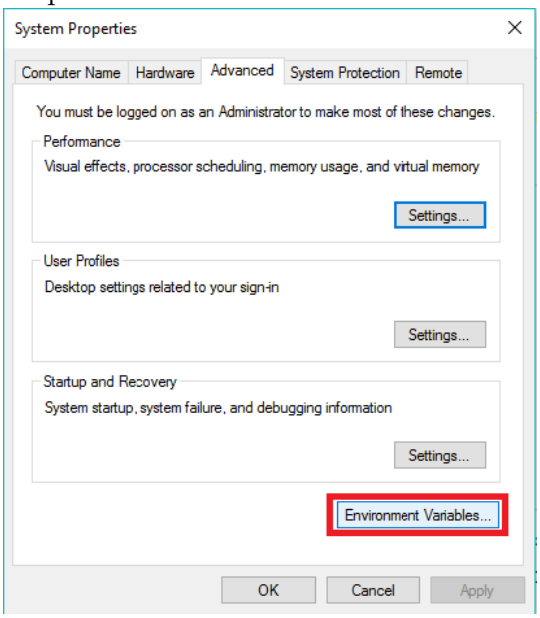
Open File Explorer, right click on This PC (Yours may be named otherwise), click Properties

- Step:2



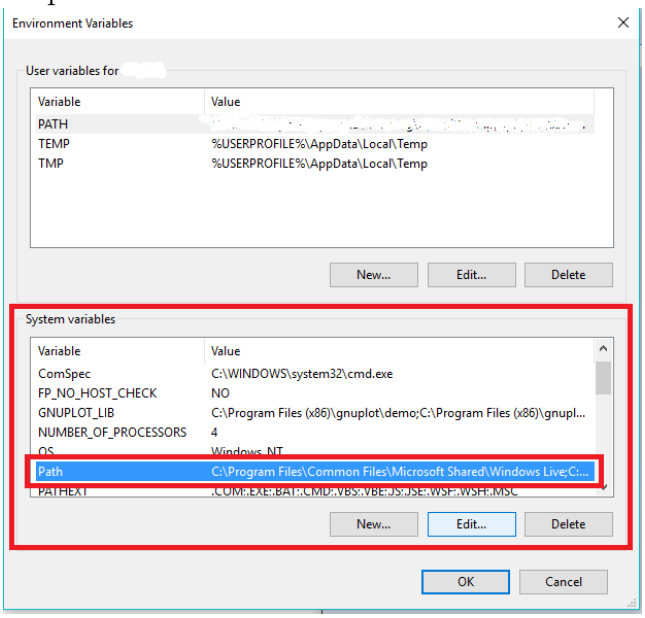
On the left hand panel select Advanced System Settings.

- Step:3



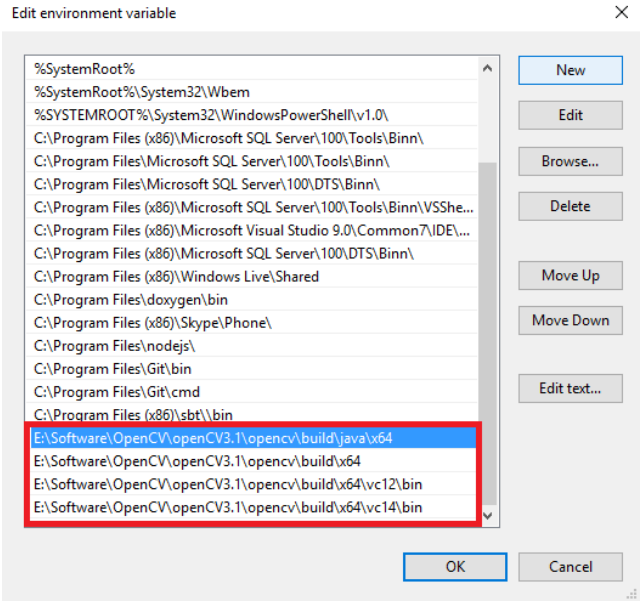
Click Environment Variables located at the bottom of the window.

- Step:4



Select Path, click Edit

- Step:5



Select New, enter the paths to the above folders which may be located in the given OpenCV files, restart computer to take effect.

2 Using The System

2.1 OpenCV

2.1.1 What is it?

OpenCV is the open source software which was selected and provides the framework utilized by the Smart Image Identifier software.

2.1.2 Why was it used?

Just one of the things many things OpenCV is capable of is that it provides libraries and functionality to assist in the detection of humans within images.

2.2 Classes

2.2.1 PeopleDetect

This is the class which Smart Image Identifier was implemented. Full Doxygen commenting is provided in this class, describing functions, their parameters and a brief description on its purpose.

2.2.2 Writer

This class provides us with video writing capabilities, it was used to create a copy of the video in question.

2.2.3 FourCC

This class provides us with a four character code (4CC) which is used by AVI files. It wraps a 32-bit value to be used as a 4CC inside an AVI file, so that it is guaranteed to be valid, and it incurs no overhead if used repeatedly.

2.2.4 Database

This class is responsible for all database related processing: retrieving an image, decoding an image as well as any other required data handling regarding the database.

2.3 Functions

2.3.1 greyscale(BufferedImage image, Mat mat)

This method receives two parameters, image and mat. The image variable is used to create a temporary mat object with the same dimensions whereas the mat object is used to transfer the data within the image after which is grey-scaled and returned. Grey-scaling assists in image processing by removing all the colour the image holds less data therefor becomes easier to process.

2.3.2 equalization(BufferedImage image, Mat mat)

This method receives two parameters: image and mat. The image variable is used to create a temporary mat object with the same dimensions whereas the mat object is used to transfer the data within the image which is required to be grey-scaled. This increased the contrast within the image helping objects to appear more defined in the image.

2.3.3 enlargeImage(BufferedImage image, Mat mat)

This method receives two parameters: image and mat. The image variable is used to create a temporary mat object with the same dimensions whereas the mat object is used to transfer the data within the image. This method enlarges an image, it helps with identification of objects as the windows of pixels processed making it easier to detect.

2.3.4 generateMat(BufferedImage image)

This method receives one parameter: image. The image variable is used to create a mat object with the same dimensions which is returned to the system and stored for processing.

2.3.5 generateImage(BufferedImage image, Mat mat, byte[] data)

This method receives three parameters: image, mat and byte. The image variable is used for identifying sizes, the Mat object holds the data which is to be saved into an image and data which holds additional data containing the location of the boxes which are drawn onto the image after a human is detected. This method outputs the results of the Smart Image Identifier and saves it as a JPG file which is stored in the output folder within the project.

2.3.6 processImage(String url)

This method receives one parameter: url. The url is the location of an image which is required to be processed. This method is the heart of Smart Image Identifier, all the processing required for human detection is done here.

2.3.7 runTests(String url)

This method receives one parameter: url. The url is the location of an image which is required to be processed. This method runs the image through various tests.

2.3.8 readDatabase()

This method connects to the Willburg database, retrieving images for processing

2.3.9 getImage(byte[] base64String)

This method connects receives one parameter: base64String. This string contains base64 encryption used to store the image in the database which is then used to create an image which in turn is pushed forward for processing.

3 Troubleshooting

The system has no human interaction, it is simply a stable communication line between two systems which is required to be maintained.