

Software Engineering Project – COS 301

Ambitious Designs

Software User Manual



Project Team: Dian Veldsman –

Project Manager: Stephen Swanepoel – 11032091

Project Owner: Willem Burger

Abstract

This document is the Software User Manual for Smart Image Identifier. The Software User Manual instructs how to install and use the Smart Image Identifier software.

This project is part of the Software Engineering course COS 301 at the University of Pretoria.

Contents

Abstract.....	2
Overview	4
Tutorial.....	5
1. Installing the Software	5
1.1. Install Eclipse.....	5
1.2. Importing the project.....	5
1.3. Setting system environment variables.....	5
2. Understand your software.....	9
2.1. OpenCV	9
2.2. Classes.....	9
2.3. Functions.....	10

Overview

The software implements human detection within images as well as videos. Images are retrieved from the server and processed by Smart Image Identifier which returns a response and outputs an example of the detection which gets saved into your projects source folder titled “output”.

The PeopleDetect.java file has been Doxygen commented for your convenience, detailing every method and giving a brief description of its purpose.

Tutorial

1. Installing the Software

1.1. 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.

1.2. Importing the project

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

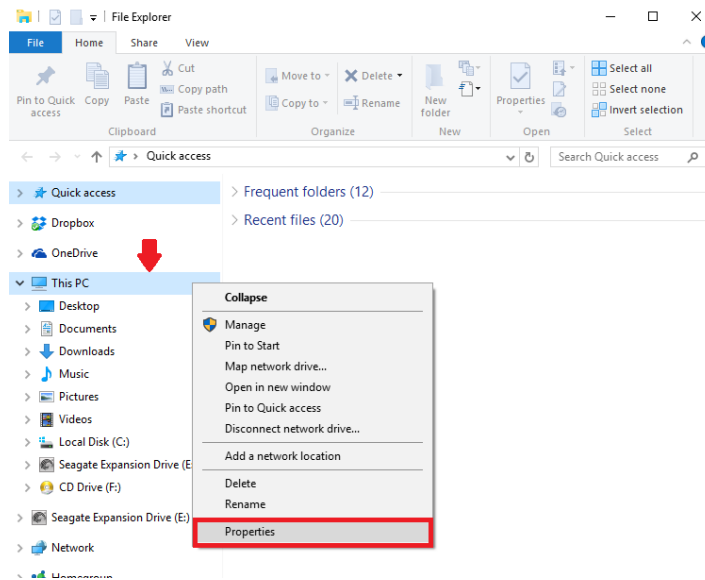
Select root directory > click browse > locate the given project folder > finally, select Finish.

1.3. 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.

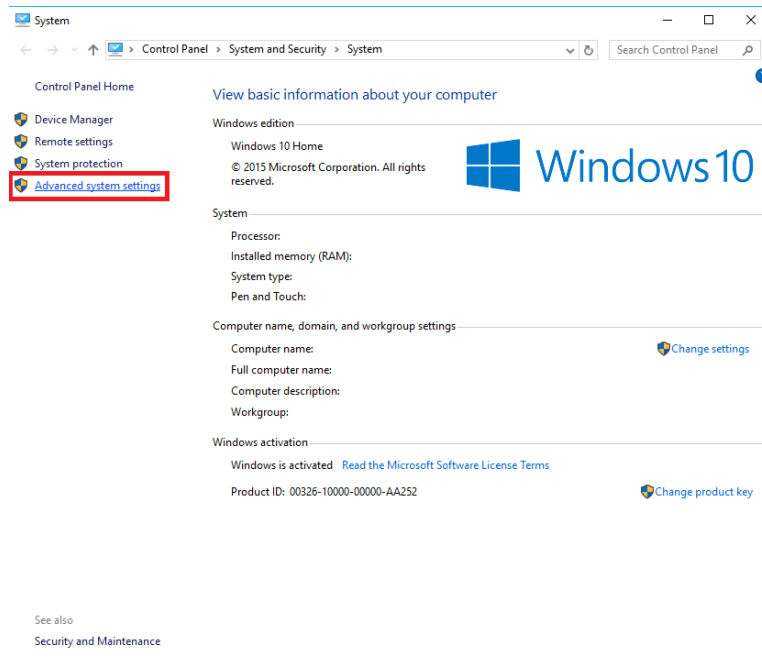
Let's get started.

1.3.1. Step 1



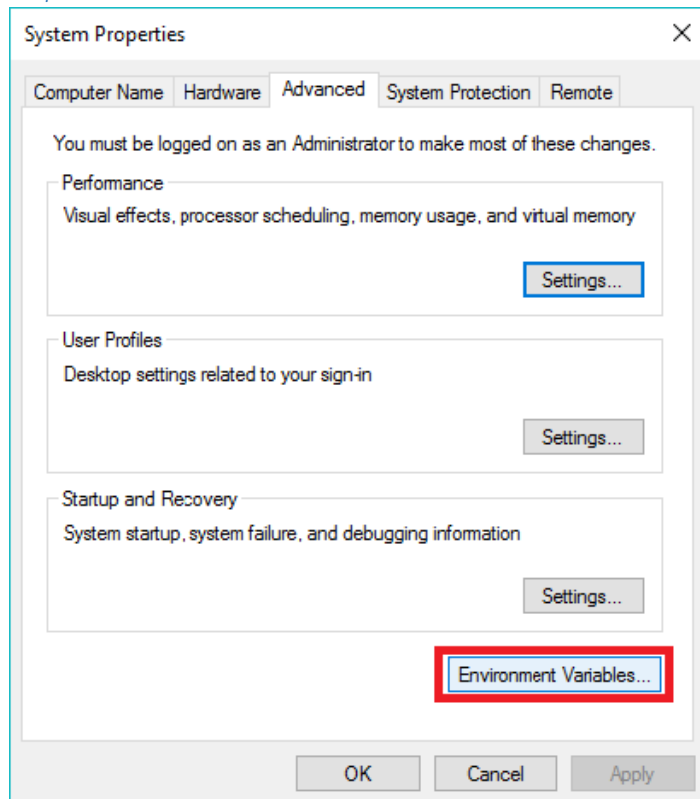
Open *File Explorer* > right click on *This PC* (Yours may be named otherwise) > click *Properties*

1.3.2. Step 2



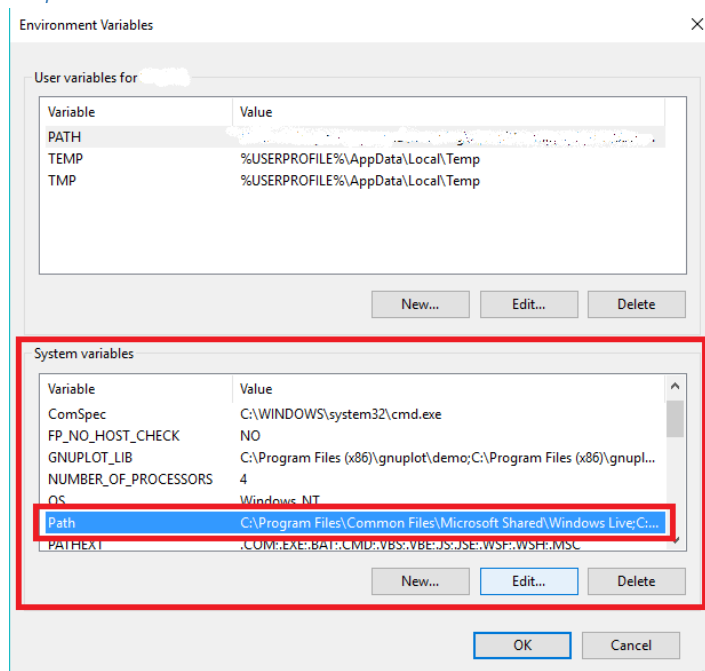
On the left hand panel select *Advanced System Settings*

1.3.3. Step 3



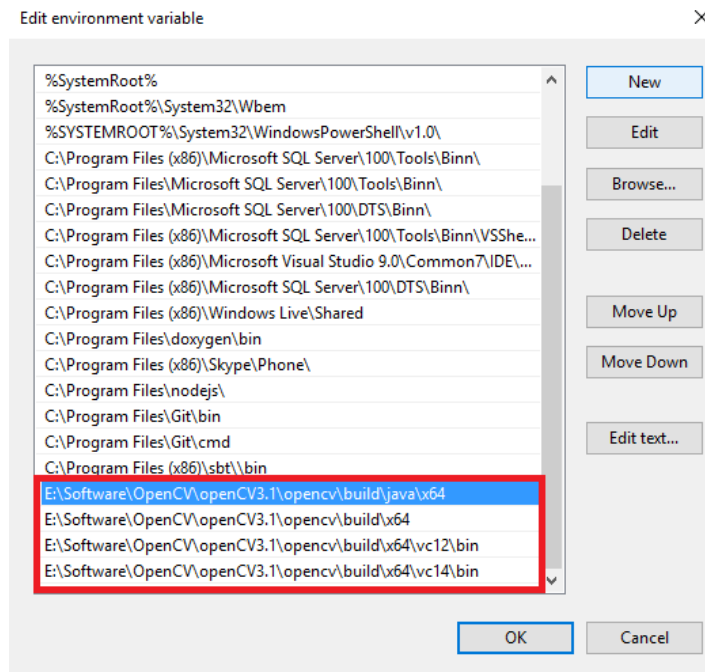
Click *Environment Variables* located at the bottom of the window

1.3.4. Step 4



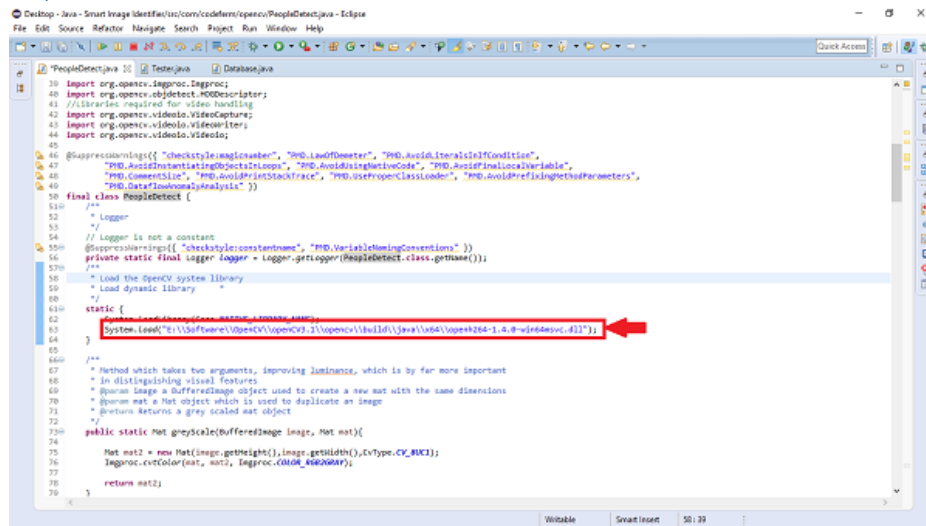
Select *Path* > click *Edit*

1.3.5. 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.

1.3.6. Step 6



Launch the project in Eclipse > Locate the line above and replace it with the location of the .dll on your system

2. Understand your software

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.

2.3.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.