Design and control of the program for the operation of the automatic parking lot in VSB-TUO campus – **license plate detection** and connection to parking slot

FEI, VSB-TUO

Abstract

A Python 3 script plus the open source license plate detection OpenAlpr program, installed on a Raspberry Pi 3B detect the license plate number of a car that is entering the 2nd door of the Automatic garage (located North to VSB-TUO Sport Hala, and besides the new CPIT 3 building), and saves it as a string in database. This project can be used in the future to substitute/support the RFID entrance to the garage in the 1st door, or other improvements.

Introduction

Combination of a Python 3 script, Python libraries OpenCV and others; and the programs Motion-project, MariaDB, and OpenAlpr.

Development platform:

* Hardware: Raspberry Pi 3 B plus camera and PIR sensor
* Application front end: Python 3 Raspbian command line calling
* Application back end: Python 3 script calling Raspbian commands
* Development environment: Thonny (<https://thonny.org/>)

Methods and materials

This software is composed of Motion project (command line software -open source-), OpenAlpr (command line software and Python library -open source-), MariaDB Server (relational database -open source-), OpenCV (command line software and Python library -open source-), and a Python 3 script plus Python and Raspberry Pi standard libraries.

Frames: a live video feed using Motion project provides the frames. Motion project can detect motion and call a script. At this first version, a PIR sensor is used to sense the motion.

Python script: The PIR sensor is sensing motion. When motion is detected, the script waits a delay of up to 1 second (this must be optimized in the parking lot) and saves a frame from the live video feed. The script uses OpenAlpr to detect license plate numbers. If there is a number, the license plate number, detection confidence and time are saved as string in the MariaDB database. Then, the frame is dumped. If no license is detected, the frame is dumped. The PIR motion sensors is sensing movement ***concurrently*** to the detection process.

Figure 1 Project structure

This project includes hardware and software components.

Hardware

Hardware list

* Raspberry Pi 3 B with Raspbian (Buster)
* Pi camera V2.1
* PIR sensor
* Resistor 220 Ω
* Cover case protection for the Raspberry Pi

Software

General description

The software is taking frames from Motion project live stream. When PIR sensor senses an object waits a small delay up to 1 second and captures a frame. This frame is analyzed with OpenAlpr to search for license plates. If a license plate (1 or more) is found, the number is inserted on a MariaDB SQL database and the frame is dumped. If no license is found, the frame is directly dumped.

Apps:

* OpenAlpr
* Motion project
* OpenCV
* MariaDB SQL
* Python 3 scripts

Software description

UML analysis

The Unified Modeling Language (UML) is a modeling language, that give the possibility to visualize the design of the system, structure and behavior of it.

USE CASE diagram

The use case diagram is a representation of the interaction between the users and the system. See in the next figure use case diagram for this project.



Figure 2 Use case diagram

User roles

The users are the car and the database.

1. Car: The car will stop in front of the 2nd entrance of the automatic garage. The car will trigger the PIR motion sensor.
2. Database: The SQL database receives/saves the license plate number, detection time and detection confidence, from the detection function if a license plate is detected.

ACTIVITY diagram

An activity diagram is type of behavior diagram that shows the activities occurring in a process. Behavior diagrams show the dynamic behavior of the objects in a system, including their methods, collaborations, activities, and state histories. The dynamic behavior of a system can be described as a series of changes to the system over time.

The activities occurring for detecting movement when the car is approaching the garage inside door.



Figure 3 Activity diagram: detecting car

The activities occurring after movement is detected.



Figure 4 Activity diagram: detecting license plate

Timing diagram

“Timing diagrams focus on conditions changing within and among Lifelines along a linear time axis. Timing diagrams describe behavior of both individual classifiers and interactions of classifiers, focusing attention on time of occurrence of events causing changes in the modeled conditions of the Lifelines”. The primary purpose […] is to show the change in state or condition of a lifeline (representing a Classifier Instance or Classifier Role) over linear time. The most common usage is to show the change in state of an object over time in response to accepted events or stimuli. (OMG Object Management Group, 2017).

The time that the detection takes is 9.5 seconds.



Figure 5Timing diagram

**How to install the system**

For this installation tutorial, it is assumed that the Raspbian is install in the Raspberry Pi 3B.

Installation (Vinczejanos, 2017) (MMattiii, 2019)

In case, the memory is limited in the Raspberry Pi, run the following commands to uninstall LibreOffice and Wolfram which will not be used.

sudo apt-get purge wolfram-engine libreoffice\* scratch -y

sudo apt-get clean & autoremove

First of all you should update your Pi with the following two commands:

sudo apt-get update

sudo apt-get upgrade

The next command installs all packages needed for all the other steps:

sudo apt-get install autoconf automake libtool libleptonica-dev libicu-dev libpango1.0-dev libcairo2-dev cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev python-dev python-numpy libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev virtualenvwrapper liblog4cplus-dev libcurl4-openssl-dev libtiff5-dev libpng12-dev gcc make ca-certificates autoconf-archive

In some cases the package can throw errors, in such a case run separately the command:

sudo apt-get install libpng12-dev

The following commands will download, make and install Leptonica. Leptonica is one of the dependencies of OpenALPR.

The making process takes some time. To speed things a little up the -j2 is added.

cd /usr/src

sudo wget http://www.leptonica.org/source/leptonica-1.76.tar.gz

sudo tar xf leptonica-1.76.tar.gz

cd leptonica-1.76

sudo ./configure

sudo make -j2

sudo make install

The next dependency is Tesseract. Again, -j2 speeds up the making.

cd /usr/src

sudo git clone https://github.com/tesseract-ocr/tesseract.git

cd tesseract

sudo git tag

sudo git checkout 3.04.01

sudo ./autogen.sh

sudo ./configure --enable-debug

sudo make -j2

sudo make install

sudo ldconfig

To test if it Tesseract was installed correctly, type the following command. This will output the installed version.

tesseract -v

The last dependency is OpenCV.

The making process of OpenCV took the most time for me, so make sure to use -j2 for the making. It stopped two times, but I just repeated the making command, and everything worked out just fine.

I had some issues with the stdlib library, but I solved them by using -D ENABLE\_PRECOMPILED\_HEADERS=OFF as cmake argument.

cd /usr/src

sudo wget https://github.com/opencv/opencv/archive/4.0.1.zip

sudo mv 4.0.1.zip OpenCV-4.0.1.zip

sudo unzip -q OpenCV-4.0.1.zip

cd opencv-4.0.1

sudo mkdir release

cd release

sudo cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local -D ENABLE\_PRECOMPILED\_HEADERS=OFF ..

sudo make -j2

sudo make install

After installing all dependencies, you can finally install OpenALPR itself:

cd /usr/src

sudo git clone https://github.com/openalpr/openalpr.git

cd openalpr/src

sudo mkdir build

cd build

sudo cmake -D CMAKE\_INSTALL\_PREFIX:PATH=/usr -D CMAKE\_INSTALL\_SYSCONFDIR:PATH=/etc ..

sudo make -j2

sudo make install

sudo ldconfig

To test if OpenALPR is working, you can download one of their licence plate pictures and run the licence plate recognition command:

cd ~

wget http://plates.openalpr.com/h786poj.jpg -O lp.jpg

alpr lp.jpg

Python 3 Script

We will use Python 3 script to fetch frames from the Motion project live video feed, and to control OpenAlpr and MariaDB, to detect license plates and save the information detected.

# Credits and references

**MMattiii. 2019.** How To: Install OpenALPR on Raspberry Pi. *Reddit.com.* [Online] April 8, 2019. https://www.reddit.com/r/raspberry\_pi/comments/baxwz5/how\_to\_install\_openalpr\_on\_raspberry\_pi/.

**OMG Object Management Group. 2017.** UML Unified Modeling Language Specification Version 2.5.1 . *omg.org OMG Object Management Group.* [Online] December 5, 2017. https://www.omg.org/spec/UML/2.5.1/PDF.

**Vinczejanos. 2017.** Install OpenALPR on Raspberry PI 3 (Part 2). *vinczejanos.info.* [Online] May 1, 2017. https://blog.vinczejanos.info/2017/05/01/install-openalpr-on-raspberry-pi-3-part-2/.