EC535 Project Report: Team Object Detection

Bharath Ananth,Jingwei Zhang
Dec 13,2017

Abstract

The aim of this project is to detect Stop-sign & Yield signs for autonomous motion of the car. Using linux's inbuilt device drivers, we obtain the image frames, convert them to RGB values, and do thresholding on these pixels to detect red or blue colours. A kernel module on gumstix, then uses flags to determine which GPIO pins to be enabled, and move the car in forward/stop or reverse direction.

1.Introduction

Automatic object recognition has long been an interesting research area in image processing, one specific area with practical importance is automatic traffic sign recognition. A robust traffic sign detection algorithm is an essential part of applications like automatic vehicle control, navigation, etc. Road signs and traffic signals define a visual language that can be interpreted by drivers. They represent the current traffic situation on the road, show danger and difficulties around the drivers,give them warnings, and help them with their navigation by providing useful information that makes driving safe and convenient.
Colors represent an important part of the information provided to the driver to ensure the objectives of the road sign. Therefore, road signs and their colors are selected to be different from the nature or from the surrounding in order to be distinguishable. Detection of these signs in outdoor images from a moving vehicle will help the driver to take the right decision in good time, which means fewer accidents, less pollution, and better safety.

In this embedded systems project, we are able to control a car on the presence of a STOP sign or a YIELD sign. The goal of this project is to detect two different signs: stop sign and yield sign in an image. A usb camera is connected to the gumstix board in order to get the video stream in. The properties of color, shape and size of the sign is used to separate them from other part of the image. We do all of our image processing on the gumstix, and then using device drivers, communicate output of our results to a kernel module which then drives the GPIO pins that moves the car in the appropriate direction. Our method here, illustrates a novel way to use a low-power, space efficient device like gumstix, do image processing without any reliance of open-source software and an application that is capable of performing in near-real time.

2 Design Flow

Components:

1. Camera: Logitech Y270 Webcam

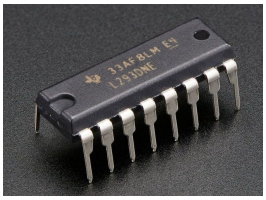
Specifications:

- USB type: High speed USB 2.0
- Video stream format: MJPEG
- Video capture resolution: 640*480

2. Motor driver: L293D

Gumstix operate at low voltages and require a small amount of current to operate while the motors require a relatively higher voltages and current . Thus current cannot be supplied to the motors from the gumstix This is the primary need for the motor driver IC.

A motor driver IC is an integrated circuit chip which is usually used to control motors in autonomous robots. Motor driver ICs act as an interface between microprocessors in robots and the motors in the robot. The most commonly used motor driver IC's are from the L293 series such as L293D, L293NE, etc. We chose the L293D as it was the most popular.



Specifications:

- Provide bidirectional drive currents of up to 600mA at voltages from 4.5 V-36V
- Two full H-bridges
- two DC motors bi-directionally

3. Car: We assembled a simple off-the-shelf 3 wheeled car, since its main purpose was to show movement along a direction. It accepts input voltage from the motor driver and moves accordingly.

Specifications:

- Size: 20 x 14cm (L x W)
- Wheel size: 6.5 x 2.7cm (Dia. x H)
- Motor power supply is 3V - 6V

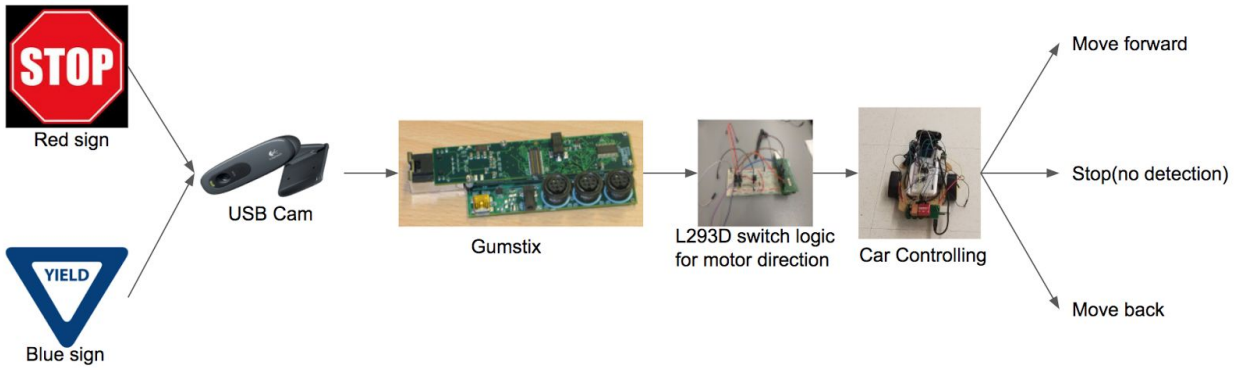
4. Battery Specifications

- For Gumstix: 5V, 1A
- For Motor driver: 9V battery stepped down to 5V using regulator for driving the motors

System Diagram

Various hardware components used in this project.A kernel level program was used to specify the voltage values for each movement of the car. A 9-V battery was used as voltage supply for the car. A power bank was used to power up the Gumstix.

The motors were operated at 3.3V for 0.5 seconds for one straight forward movement. To perform a turn, the opposite motor was operated at 3.3V, while the other motor was turned off



System Diagram

Team Contribution

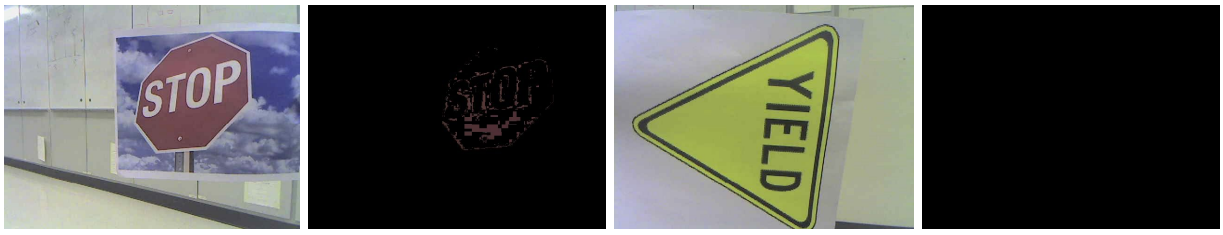
- Gumstix kernel module programming: Jingwei 20% Bharath 80%
- Image Processing: Jingwei 80% Bharath 20%
- Gumstix-car interface/Motor drivers: Bharath 80% Jingwei 20%
- Car Assembly: Jingwei 50 % Bharath 50%

3 Project Details

Image Processing:

The camera can be connected to gumstix USB port through a USB gender bender. Our goal is to get the image frame from the camera video stream every second, and use a C code to get the pixel numbers. The pixel format should be converted to RGB format. We find the thresholding values for red and blue color, and extract only the red and blue pixels out for every image. Then we can detect the aiming color depends on the numbers of extracted pixels.
As opencv library is too big to gumstix board, we have to use a v4l2(video for linux 2) driver module to get the video source in. Therefore, our initial idea is saving a jpeg image from the camera to gumstix, and read the jpeg file in order to do the image processing. The default camera image source format is MJPEG format. Using the sample code provided by professor,

we are able to save the MJPEG format image to a jpeg file. However, jpeg image is a compressed image format. We were not able to read the image using any C library. We have to use C++ code to read the jpeg image, but we are not able to compile C++ code on gumstix. Therefore, we decide to read and convert the input image straight to RGB values instead of saving it to jpeg file. As the camera do not accept RGB format input, we have to use YUYV format to get the image in. Using the function “mmap”, we can save the frame buffer to a local variable called “buffer”. The datatype of buffer is uint8_t, which is a C type 1d array. The buffer image type is



Stop/yield sign and thresholding result

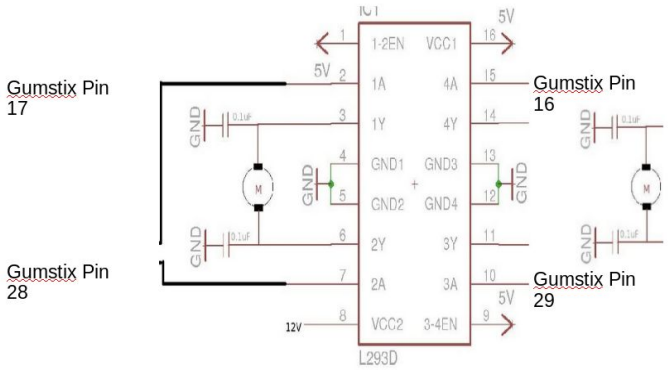
“V4L2_BUF_TYPE_VIDEO_CAPTURE”, and memory type is “V4L2_MEMORY_MMAP”. Then, we can convert every YUYV pixel inside the buffer to RGB pixel number. YUYV pixel format contains 4 bytes for every 2 pixels. Using this function: B=0.7*Y+2*(U-128), G=0.7*Y+2*(U-128), R=0.7*Y+2*(V-128), we can convert it to RGB values. As the lighting condition in lab 307 is very good, we use a small Y parameter to decrease the image luminance.

After getting the RGB values, we can apply the threshold values to every pixel. The threshold bound we are using is: 95<R<255, 0<B/R<0.7, 0<G/R<0.7 for red, and 95<B<255, 50<R<95, 50<B<95 for blue. We wrote a python code to show the images after the threshold. The pictures above show the image processing result. If we input a picture to the thresholding, the red pixels will be extracted and all the other color pixels will be removed.

Circuitry:

We followed the below circuitry to interface gumstix with the motor’s/motor driver.

To avoid inrush currents and regulate the current/voltage input/output to gumstix, we also used a standard Voltage regulator.



Circuit diagram

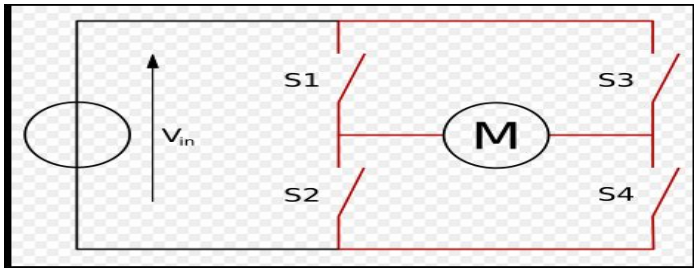
Gumstix Programming:

We wrote a kernel level module to read the status of the flag sent by the Image processing algorithm and populating the output of the GPIO pins appropriately.

The kernel level module first registers a device file with the GPIO pins sets appropriately to LOW. This device file is shared by the Image processing algorithm which then writes a status into the device file with these 3 options

- Status of 0 - to indicate no presence of Blue/red colour
- Status of 1 - to indicate presence of Red
- Status of 2 - to indicate presence of Blue.

An H bridge is built with four switches (solid-state or mechanical). When the switches S1and S4 are closed (and S2 and S3 are open) a positive voltage will be applied across the motor. By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed, allowing reverse operation of the motor.



TRUTH TABLE (one channel)

Input	Enable (*)	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance
(*) Relative to the considered channel

Kernel module, then reads the device file and depending on the flag sets the GPIO pins appropriately

- Status of 0 - Dont move
- Status of 1 - Move forward
- Status of 2 - Move in the opposite direction.

4 Summary

Realtime feature
Through our method and initial tests, we are in a position to detect red and blue colors in a continuous video stream. This is achieved without the use of any open source software.
In addition, we are in a position to do image processing on each frame and produce the results within a fraction of 1 second. This is because there is only one parent while loop(a loop that spans across width*height*2) and we do all the calculation within this loop. This ensures that our system confines within RTOS (Real time operating system) guidelines.

Accuracy
For the color detection, the red color detection is very accurate. With a good lighting condition, the accuracy can reach close to 90%. The blue color is a little bit lower than red, which can reach around 70%.

Cost effectiveness
Coming to hardware results, since we have used the motor driver L293D, we are in a position to not only control forward/backward/stop movements of the motor, but also make turns either way using switching logic.
For ex: if we intend doing a left turn, just by motor dynamics (like moving only either wheel) we could perform turns without the need for a separate servo motor.
Also, a car which was bare minimum in terms of assembly and hardware was used to communicate our proof of concept. This brings down our cost significantly holding the principles of embedded system design.

5 Reference

1. https://linuxtv.org/downloads/v4l-dvb-apis/media_uapi.html
2. http://ai.stanford.edu/~kosecka/FinalReport_5_T2.pdf
3. http://www.jayrambhia.com/blog/capture-v4l2
4. http://www.ti.com/lit/ds/symlink/l293.pdf