RASPBERRY PI DRIVEN VISION FOR AUTONOMOUS MOBILITY

Andrei Tumbar

Computer Engineering
Rochester Institute of Technology
at1777@rit.edu

ABSTRACT

Classical computer vision has always been a challenge to implement on an embedded systems because of the requirement of fast computational power and system memory. This paper presents a possible solution to realtime computer vision aboard a Raspberry Pi 3B for the purpose of driving an autonomous vehicle.

1. INTRODUCTION

Vision processing is a fairly well developed field that looks at processing images to extract certain features. This can include features such as object edges, object classification, or 3D mesh reconstruction with stereo correlation. One of the major challenges of computer vision in robotics and other embedded systems arises from vast computational requirements.

Many robots that operate from the input of a camera will use some form of computer vision to drive their navigation logic. For example, the Perseverance rover uses a combination of its 6 engineering cameras as well as an inertial measurement unit for path planning and pose estimation [1]. One of the main challenges with computer vision for robotics and other embedded system arises from the large computational requirements of vision software. While algorithms are parallelizable making graphics processing units (GPUs) highly attractive, power constraints of these robots usually factors in to these robots.

There has been some work in the past done to implement vision on the Raspberry Pi. [3] discusses a Raspberry Pi configuration that uses the Raspberry Pi camera module. OpenCV is used to support much of the vision functionality. This project will use some of the design decisions from [3] however will not used the Python OpenCV packages distributed for the Raspberry Pi. Instead, a custom compiled, statically linked version of OpenCV is built as well as a leaned down version Debian Linux operating system is used instead of the one noted in [3].

This paper introduces a vision and navigation system aboard a Raspberry Pi 3B. The official Raspberry Pi Camera Module V2 is used for its impressive streaming features and hardware noise reduction. The vision system is meant to drive a small battery powered car around a track. The Raspberry Pi's low power requirements of 3.7 W at maximum load of all four internal cores makes it optimal for this application [2]. Unlike robots mentioned previous in [1], this vision system will only operate off of a single camera. Depth estimation is performed using a calibrated camera pose and an affine transform for adjusting camera perspective. This paper will discuss the implementation details of the vision and navigation systems such as the software framework used as well as the physical configuration of the car.

2. BACKGROUND

2.1. TI Cup Car Hardware

The purpose of this project is to drive a TI Cup Car. This car includes two direct current (DC) motors used for throttle as well as a servo motor to control the steering arms. All three motors are driven with a pulse-width modulation (PWM) signal. The DC motors used for the rear wheels have a base frequency of 10 kHz while the steering servo uses a frequency of 50 Hz. Due to the low current limitations of a software drivable general purpose input/output (GPIO) pins, the DC motors cannot be switched directly from the pin headers. A motor driver must be used to switch an external power source controlled by a low drive GPIO. This motor drive H-bridge circuit able to switch the high current requirement of the DC motors from the external battery under the control of a 3.3 V GPIO PWM signal.

One of the challenges the Raspberry Pi board for use with this vehicle is the hardware limitations presented by the board. The Raspberry Pi only includes two hardware PWM channels. Each channel may be configured to drive one of two selectable GPIO pins exposed on the Raspberry Pi header. This leaves one of three options: drive both the DC motors with a single PWM signal, drive the DC motors with the hardware PWM and the steering servo with software PWM, command a microcontroller to generate PWM signals and control all three motors on its own.

The first two options noted above both present problems of their own. The first option is not possible to be implemented on the specific motor driver board as the pins that drive the DC motors only work with one set of the PWM pins on the two channels. Implementing the servo signal using a software driven PWM signal is risky as the response time of a non-realtime kernel like [standard] Linux is too long to generate a reliable and stable square wave. The final option is the most favourable as the PWM signal generation aboard a microcontroller is always hardware driven and the power requirements of the microcontroller are extremely minimal. This option however does add some complexity to the system as the requirement for yet another board is introduced as well as well as a communication protocol between the two boards. The MSP432P401R microcontroller from Texas Instruments is used for the purpose of driving the PWM pins to control the motors. This is the same board used in class and already has PWM libraries written for previous labs as well as testing on these libraries for validity.

2.2. Camera, Vision & Navigation

The software built to control the vehicle is divided in three main stages:

1. Capture camera frame
2. Detect and threshold track edges
3. Compute path and control car to stay on path

The first stage simple involves taking raw image frames from the camera module. The camera will be set up to stream at 30 frames per second (FPS). The official Raspberry Pi Camera Module is a general purpose camera capable of taking high definition video or still single image frames. The camera is driven by a camera library built by Google named libcamera. While libcamera is able to interface with more traditional Linux video streaming drivers such as gstreamer, this project will implement a camera driver using a native libcamera application.

The vision processing step will take raw image frames captured by the camera and apply a series of processing stages. The collection of processing stages is knows as the vision pipeline. To perform the vision, the open-source OpenCV library is used. OpenCV is split into independent modules which are can be built as separate static libraries. This allows a user to compile OpenCV with only a subset of its features to keep the code sizes relatively low. This project uses the image processing module (imgproc) as well as the 3D camera calibration module (calib3d) to support its pipeline.

The final stage in the vehicle control process will take processed images from the vision pipeline and plan a path for the car. It will classify track edges as splines and attempt to find a path within the edges of the track. This stage will keep the car on course using the Proportional-Integral-Derivative (PID) control paradigm.

2.3. Software Framework

3. PROPOSED METHOD

4. RESULTS

5. CONCLUSION

6. REFERENCES

7. REFERENCES

[1] “Perseverance, Mars 2020 rover : Discussion,” NASA. [Online]. Available: https://forum.nasaspaceflight.com/index.php?topic=38208.1460. [Accessed: 18-Apr-2022].

[2] “Power consumption benchmarks,” Power Consumption Benchmarks — Raspberry Pi Dramble. [Online]. Available: https://www.pidramble.com/wiki/benchmarks/power-consumption. [Accessed: 18-Apr-2022].

[3] A. Rosebrock, “Raspberry Pi For Computer Vision,” PyImageSearch, 08-May-2021. [Online]. Available: https://pyimagesearch.com/2019/04/05/table-of-contents-raspberry-pi-for-computer-vision/. [Accessed: 18-Apr-2022].