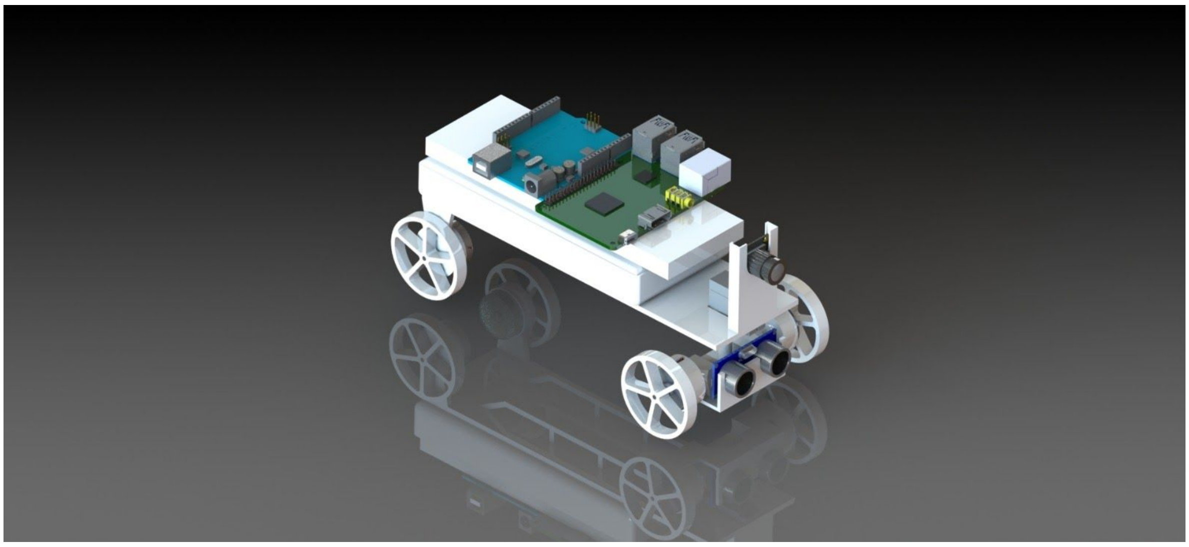
Marco Spanò

Tracking Car

Introduction

A Real-Time application drives the car to chase a blue ball till the ball is not reached or lost from the camera view. Using an ultrasonic sensor on the front, the car is also able to cut off the engine when the object is close to the vehicle or an obstacle is found. Two videos are recorded from the camera during the whole run, one directly from the camera and one after the detection process (showing in white what it has detected).

Below a demonstration video of the final result:
https://youtu.be/kEDyJJeeosM



Building part

The car is completely assembled from scratch. It is composed by:

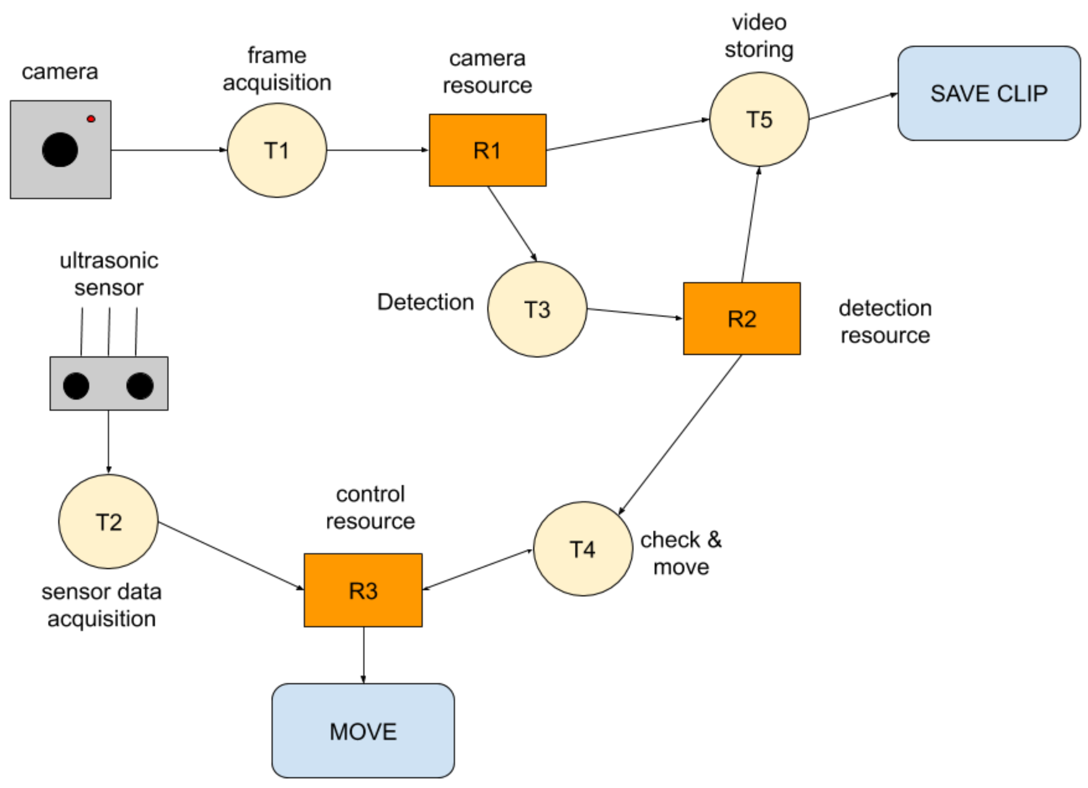
- 3D printed chassis, steering block and wheels
- Raspberry Pi 3 model B
- MicroSD 8 GB
- USB key 128GB
- Arduino UNO
- Camera module for Raspberry
- 4x PS3 DualShock Controller Vibration Motors
- Step up boost converter
- Micro Servo MG90S
- Ultrasonic Sensor HC-SR04
- Relay module
- Capacitor
- Transistor
- Diode
- Resistor
- Breadboard
- Wirings

The main device of the project is the Raspberry, that takes the information from the camera and the sensor and coordinates the car thanks to the real-time application. While the camera is directly connected to the Raspberry Pi, all the other sensors and actuators are connected to the Arduino Uno, that behaves as a slave and communicates with the master through a serial connection. The arduino code is designed to be easily extendable and is divided in two parts: the first one takes the data from the sensors (in this project only an ultrasonic sensor is present) and sends it to the master, while the latter one takes a packet from the master, parses it and executes the function related to it (e.g. moving the steering).
Onboard is mounted an ultrasonic sensor to check the distance from the closest obstacle, a servo motor that moves the steering block and 4 motors, one for each wheel. The engines, that were vibration motors of an old PS3 controller, are controlled from the Arduino by a small relay. Moreover, a capacitor is mounted in parallel with the motors in order to have a small boost during the start, while the electricity from the batteries passes through a step up boost converter that enhances the voltage directed to the motors, to improve their performance.

The 3D models were completely designed and printed by **Michelangelo Menghi**, that helped me also during the construction of the car.

Project structure

The application is divided in 5 periodic tasks, each one designed to address a different feature of the car.



Frame acquisition

This task handles the communication between the application and the camera module, using the OpenCV library features. Its goal is to capture the current frame from the camera and saving it inside the 'Camera resource', so that it is publicly available to the other tasks. Since the frame availability is mandatory to pursue the detection and finally the movement of the car, this is a max priority task.

Sensor bridge

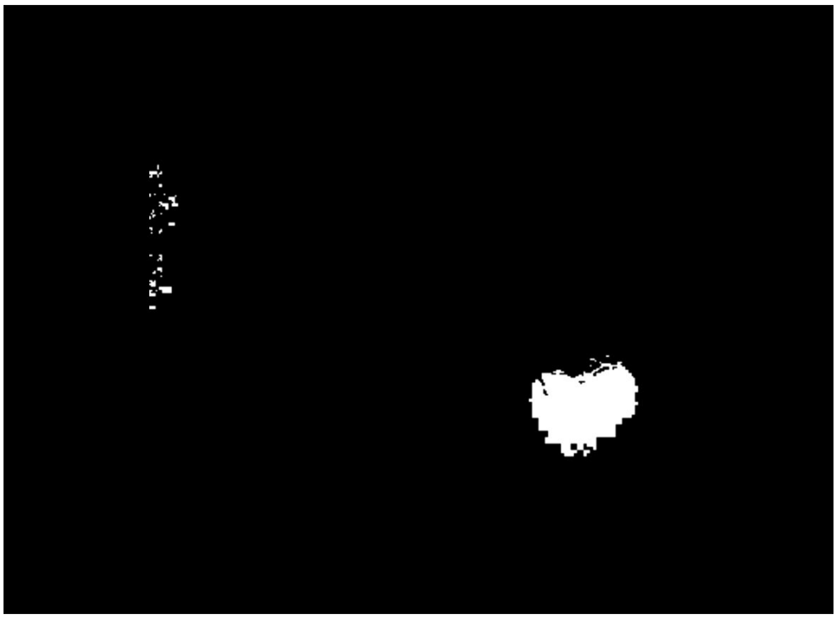
To avoid bumping into obstacles, I mounted an ultrasonic sensor on the front of the car that can measure the distance to the closest obstacle. To get those informations inside the application, this task reads all the upcoming packets from the serial, parsing them one by one and evaluating if the car should be stopped or not. Given its huge importance to avoid physical consequences to the car, this task is set to max priority too.

Detection

The detection functionality is divided in two main parts: color filtering and object positioning. This task addresses the first part, taking the frame acquired from the camera and returning back a filtered black&white one. First of all, it converts the frame to the HSV format, to avoid as much as possible that a different light exposure could lead to an improper detection of the object colour. Then, it filters the color using a threshold in the ranges of each H, S and V values, acquiring a bit matrix corresponding to the frame, where 1 bits are detected pixels and 0 bits are background. Finally it converts this matrix to a 3-channel frame, so that it can be stored inside a video.

Movement

The second part of the detection process, the object positioning, is achieved by the fourth task. After the black&white frame is ready, it starts looking for the target object in the frame: firstly it finds the contours of each object inside the image and then it checks object by object which has the largest area, saving its X and Y coordinates. Now that the detection part is done, it can finally calculating and sending the values to move the car, provided that the ultrasonic sensor hasn't found any near obstacle.



Video storing

The whole application execution is filmed and stored by the system into two different videos. This goal is reached by the last task, that continuously stores the current frame from the camera and the filtered black&white image, the latter in order to have also a view of what the car really sees during its run. Since we prefer to have maximum car performance instead of a perfect video acquisition, I setted the priority of this task as the lowest among the five.

Important parameters

The main parameters that can be tuned are all contained within ‘car_tracker.h’ and ‘car_tracker.cpp’ files. Here a list of the most important ones:

Color filter - min and max values of the H, S and V values and all detection related parameters can be changed, in order to achieve a better detection of the object or to change the aim to a different color (the object used in this project is a blue ball).
Control system - since the control system is designed to be extendable, new sensors and actuators can be easily added to the application (but remember, you have to change the Arduino code too).
Device settings - camera, serial communication and other parameters are setted based on each external device needs.
Task parameters - Task parameters (e.g. the period of each one) are defined to fit the Raspberry Pi 3 processor inside that actual car structure. If you are using for instance a different type of master device, you should tweak those parameters to reach the best performance setup.