hamuchiwa / **AutoRCCar**

OpenCV Python Neural Network Autonomous RC Car

33 commits | **1** branch | **0** releases | **2** contributors | BSD-2-Clause

Branch: **master** | New pull request | Find file | Clone or download

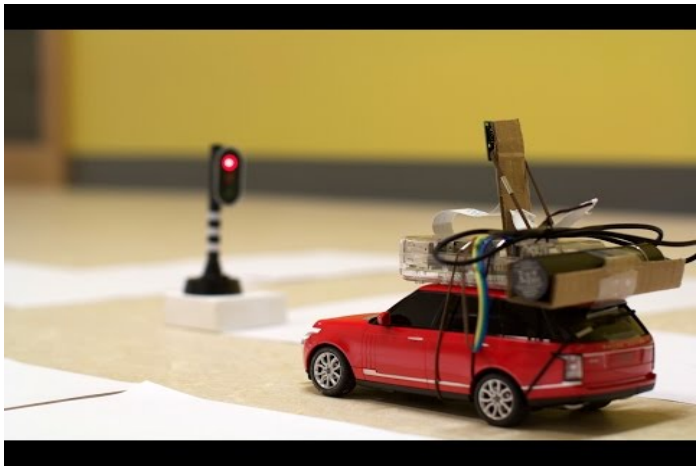
hamuchiwa committed on **GitHub** Update ultrasonic_client.py — 2 — Latest commit `d79e6f7` on Jun 29

Traffic_signal	Traffic_Signal	9 months ago
arduino	Create README.md	2 years ago
computer	Update mlp_training.py	4 months ago
raspberryPi	Update ultrasonic_client.py	4 months ago
test	Update ultrasonic_server_test.py	2 years ago
LICENSE.md	Create LICENSE.md	2 years ago
README.md	update README.md	4 months ago

README.md

AutoRCCar

See self-driving in action



A scaled down version of self-driving system using a RC car, Raspberry Pi, Arduino and open source software. The system uses a Raspberry Pi with a camera and an ultrasonic sensor as inputs, a processing computer that handles steering, object recognition (stop sign and traffic light) and distance measurement, and an Arduino board for RC car control.

Dependencies

- Raspberry Pi:
  - Picamera
- Computer:
  - Numpy
  - OpenCV 2.4.10.1
  - Pygame
  - PiSerial

About

- raspberrt_pi/
  - ***stream_client.py***: stream video frames in jpeg format to the host computer
  - ***ultrasonic_client.py***: send distance data measured by sensor to the host computer
- arduino/
  - ***rc_keyboard_control.ino***: acts as a interface between rc controller and computer and allows user to send command via USB serial interface
- computer/
  - cascade_xml/
    - trained cascade classifiers xml files
  - chess_board/
    - images for calibration, captured by pi camera
  - training_data/
    - training data for neural network in npz format
  - training_images/
    - saved video frames during image training data collection stage (optional)
  - mlp_xml/
    - trained neural network parameters in a xml file
  - ***picam_calibration.py***: pi camera calibration, returns camera matrix
  - ***collect_training_data.py***: receive streamed video frames and label frames for later training
  - ***mlp_training.py***: neural network training
  - ***rc_driver.py***: a multithread server program receives video frames and sensor data, and allows RC car drives by itself with stop sign, traffic light detection and front collision avoidance capabilities
- test/
  - ***rc_control_test.py***: RC car control with keyboard
  - ***stream_server_test.py***: video streaming from Pi to computer
  - ***ultrasonic_server_test.py***: sensor data streaming from Pi to computer
- Traffic_signal/
  - trafic signal sketch contributed by @geek111

How to drive

1. **Flash Arduino**: Flash *"rc_keyboard_control.ino"* to Arduino and run *"rc_control_test.py"* to drive the rc car with keyboard (testing purpose)

2. **Pi Camera calibration:** Take multiple chess board images using pi camera at various angles and put them into "chess_board" folder, run *"picam_calibration.py"* and it returns the camera matrix, those parameters will be used in *"rc_driver.py"*

3. **Collect training data and testing data:** First run *"collect_training_data.py"* and then run *"stream_client.py"* on raspberry pi. User presses keyboard to drive the RC car, frames are saved only when there is a key press action. When finished driving, press "q" to exit, data is saved as a npz file.

4. **Neural network training:** Run *"mlp_training.py"*, depend on the parameters chosen, it will take some time to train. After training, model will be saved in "mlp_xml" folder

5. **Cascade classifiers training (optional):** trained stop sign and traffic light classifiers are included in the "cascade_xml" folder, if you are interested in training your own classifiers, please refer to OpenCV documentation and this great tutorial by Thorsten Ball

6. **Self-driving in action**: First run *"rc_driver.py"* to start the server on the computer and then run *"stream_client.py"* and *"ultrasonic_client.py"* on raspberry pi.