System Integration Lab

Design of Embedded Intelligent Systems, DT8007, HT19, LP1

The purpose of the lab is to, by solving an example, experience some techniques for integration of heterogeneous software and hardware components. It will be done by solving an autonomous driving scenario.

Scenario

The autonomous vehicle can drive in different modes; single vehicle driving (driving mode 0), head in convoy driving (driving mode 1) or in the convoy driving (driving mode 2). The system behaves differently depending on the mode. Important components in the system are:

- A collision detection module based on a camera,
- A control module,
- A brake light and in-convoy communication module.

Mode/sub-scenario	Stimuli	Action
Mode 0	None	None
Mode 0	Object in camera scene	Brake lights on
Mode 1	None	No brake light, send heartbeat message with time info every second
Mode 1	Object in camera scene	Brake lights on, send warning message of danger ahead to convoy (include time)
Mode 2	None	No brake light, send heartbeat message with time info every second
Mode 2	Warning message received	Brake lights on, repeat warning message (include time)

Task

- Implement (in Python on the Raspberry Pi) the collision detection with the camera, by e.g.
    - Take a background image and subtract that from current image, if the difference is large enough alert:
        - http://docs.opencv.org/master/d1/dc5/tutorial_background_subtraction.html
    - Motion detection:
        - https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/

- Implement the brake light simulator (in Matlab or C on a PC)
    - When a message is received from the controller make an action, by e.g.
        - Simulate by text messages
        - Make a simple GUI
- Implement (your own choice of platform) a controller which will:
    - Integrate the two modules described above
    - Capture local time on reception of message
    - Log and analyze time of the messages

Use LCM (https://lcm-proj.github.io/index.html) for implementation of the messages. A special message is used to indicate the mode.

Questions to address

1. How do the messages look like in LCM?
2. How do you represent time?
3. How well do the time match between the different parts of the systems (i.e. the different computing platforms)?
4. What type of jitter do you get on the heartbeat signal (express in variance in seconds)?
5. How does the jitter change with multiple vehicles (1, 2, 5, 10, and 20)? (To check this, simulate multiple vehicles with multiple processes)
6. What do you think is the cause of the jitter? Can it be reduced, if so how?

Submission

Send the solution to: wojciech.mostowski@hh.se, the deadline is **18 October 2019.** In the solution provide the group members, the answers to the questions and the implementation.

Useful links

- LCM:
    - https://lcm-proj.github.io/index.html
    - http://dspace.mit.edu/bitstream/handle/1721.1/46708/MIT-CSAIL-TR-2009-041.pdf
- OpenCV: http://opencv.org/

RaspberryPI Practicalities

- OpenCV and Python:
  http://www.pyimagesearch.com/2015/02/23/install-opencv-and-python-on-your-raspberry-pi-2-and-b/
  Use "`make -j4`" instead of "`make`" to utilise Raspberry PI 4 cores.
- Picamera:
  http://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/
- LCM, installation:

```
$ sudo apt-get install autoconf libtool
$ git clone --depth=1 https://github.com/lcm-proj/lcm lcm
$ cd lcm
$ cmake .
$ make -j4
$ sudo make install
```

- Additionally you may need this to make it work with Python and OpenCV, when in the CV environment, if there is a problem here ask Google or lab supervisor. But please do not invoke this command before checking if things work, they may not be necessary with the current versions of the software involved:

```
$ sudo ln -s /usr/local/lib/liblcm.so.1 /usr/lib/liblcm.so.1
$ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
$ ln -s /usr/local/lib/python2.7/site-packages/lcm/
```