CS 378 - Autonomous Vehicles in Traffic I

Week 7b - Working with OpenCV
OpenCV

- OpenCV is an open source computer vision library that was originally started by Intel, and is now maintained by people from Willow Garage along with open source contributors in the vision community
- A large part of vision research is typically done in Matlab. As a result, there is a lot more code available through Matlab than OpenCV.
- However, we like OpenCV because it is easier to integrate vision into robotics through it instead of Matlab-based approaches.
  - Additionally, a lot of OpenCV code has been written fairly efficiently
OpenCV

- OpenCV has a lot of implementations of some really cool algorithms. We'll see one of them today.
  - Person Detection

- We'll also see some basic filters and image manipulation
  - Threshold
  - Image Derivatives (Sobel) and Edge Detection (Canny)
  - Dilate and Erode
  - Flood Fill
Working with OpenCV

- OpenCV supports C, C++ and Python through different APIs.
- I think I read somewhere that all new development will be through the C++ and Python APIs.
- All the examples we'll see today are written in C++
- The latest version is 2.3.1.

Documentation
- OpenCV is decently documented now, and the latest documentation is for the v2.2 API
- C++
  - [http://opencv.jp/opencv-2.2_org/cpp/](http://opencv.jp/opencv-2.2_org/cpp/)
- Python
  - [http://opencv.jp/opencv-2.2_org/py/](http://opencv.jp/opencv-2.2_org/py/)
ROS and OpenCV

- It is fairly simple to use OpenCV within the ROS ecosystem.

- We saw an example of this on Monday
  - We performed some basic image manipulations on camera images from the car
  - The slides do a decent job of explaining how to perform image processing in ROS using OpenCV.
  - Take a look at today's code as well.

- Today I will focus on demonstrating as many applications as possible.
  - I'll go through the code at the end of the class.
Getting today's code

- Most examples from today are available through a single package called `intro_to_opencv`
  - It is located in the `art_examples` stack. I placed it there yesterday, so you will have to update your local copy through SVN.

- You should be able to run this code on the image stream available through the bag files using topic remapping
  - `usb_cam/image_raw:=center_front/camera/image_raw`

- To run it using a web camera, you'll have to use your own machine. Also install the following package:
  - `sudo apt-get install ros-electric-bosch-drivers`
  - This stack contains the `usb_cam` package, which supports many different USB cameras out-of-the-box.
Running the code

- Either play back a bag file, or launch the driver for your camera
  - `roslaunch intro_to_opencv usb-camera.launch`
  - This launch file should work in most cases, but it is possible it will not for your setup. Let us know if you face any problems

- Run the following command to run the example
  - `rosrunt ner_to_opencv opencv_example --method <method> [--image <image> _image_transport:=compressed]`
  - Note that you can optionally give the command line argument `_image_transport:=compressed`. This is a better method of doing what I showed you on Monday