CS 378 - Autonomous Vehicles in Traffic I
Week 7a - SVN, Camera Processing

OK, back to business!

- As some of you may have experienced, programming assignment 2 submission was a bit of a mess on Wednesday night.
  - Not entirely unexpected
  - I realized that not going through a basic SVN tutorial was a mistake. We will go through one today.
  - We gave extensions when we were unable to solve the problem. Solving problems over email is pretty hard!
  - If you still feel uncomfortable with SVN, try to finish the assignment a day early and get help in office hours to submit it. It is unlikely that extensions will be given for this reason in the future.

Announcements

- Car visit on Friday (3/2)
  - We'll be testing some code we have written for the Velodyne
  - Not the best day to visit us, as we'll be working on the suburban roads outside PRC. We'll also be a bit busy.

- Explore UT on Saturday (3/3)
  - I heard back from 3 students - Thanks!
  - Let me know if you want to attend
  - We'll try and do autonomous runs

Common problems

- ROS_PACKAGE_PATH issues
  - Let's take a look at the instructions.
  - Your ROS_PACKAGE_PATH environment variable needs to point to wherever you checked out the repository
  - Make sure you don't simply overwrite the variable, but add your directories properly
Common problems

- Copy and pasting commands directly from the assignment
  - Be careful with this, as the code that you are trying to copy may not have been encrypted properly - I will try to make sure that this does not happen in future.
  - Programming assignment 2 example
  - When in doubt, type the command by hand

Troubleshooting

- Reporting errors to us
  - Whenever reporting errors, cut and paste directly from the terminal window itself.
  - Try to give as much information as possible.
  - The context for your commands is also extremely useful.
- Reporting for SVN issues
  - By default, send us the output of the following commands
    - `cd ~/svn/sandbox`
    - `svn status`

Basic SVN Usage

- SVN is a software versioning system that is extremely popular.
- Learning such a system will be useful to all of you, whether you go into research or not.
- The basic commands that you'll need to know to use are
  - `svn add`
  - `svn rm`
  - `svn status`
  - `svn info`
  - `svn commit` (OR `svn ci`)
  - `svn checkout` (OR `svn co`)
- An easy to understand overview is here:
  - [http://www.abbeyworkshop.com/howto/misc/svn01/](http://www.abbeyworkshop.com/howto/misc/svn01/)

SVN checkout

- This is used to checkout a SVN repository or part of it. It is pretty easy to use.
- You used this command when you downloaded the code. (the instructions were [here](http://www.abbeyworkshop.com/howto/misc/svn01/))
- Examples
  - `svn co https://utexas-art-ros-pkg.googlecode.com/svn/trunk/utexas-art-ros-pkg`
  - `svn co https://utexas-art-ros-pkg.googlecode.com/svn/trunk/stacks/art_vehicle`
- You can also provide the `--username` flag to indicate your username while checking out or committing in.
### SVN add/SVN rm

- **svn add** adds a new file or folder (recursively) so that it can be committed to the repository.
  - If you create a new file or folder inside a directory that has been added, then you need to add this file/folder manually.
- **svn rm** remove files from your local machine, and deletes them from the repository whenever the next commit is made.
  - If you are unsure about deleting your local copy, always use `svn rm --keep-local`.
- If you use `svn add` and `svn rm` on a file without committing it, then it is lost!

### SVN status/SVN info

- **svn status** gives you information about the current files.
- This is extremely useful for us to debug problems you might be facing. As time progresses, you'll figure out ways of dealing with problems based on the status.
- **svn info** gives some basic information about the directory.
- It is useful for looking up the path of a given directory on your local machine inside the repository.

### SVN commit

- Commits any added/modified/deleted files, folders or properties to the repository.
- You can optionally give the `--username` and `--password` flag when you commit.
- As Jack suggested, use single quotes instead of double quotes while committing.

### Common SVN problems

- Being prompted for a 'null' GNOME keyring.
- Copying folders obtained from `svn`.
  - Some of you may have copied the sample folders provided.
  - SVN stores information in `.svn` folders that also get copied, and may cause problems.
  - Once you copy the folder, run the following command inside the copied folder to remove old `svn` files:
    - `find -name .svn -print0 | xargs -0 rm -rf`
Programming Assignment 3

- Let's take a look at [programming assignment 3](#).
- A screenshot from my visualization is also available for guidance.
- This is a fairly easy assignment, especially if you understood the previous ones well.
- Live Demo (if time is available)

Troubleshooting SVN problems

- If something has gone wrong and you can't fix the problem:
  - Backup your locally edited/added files
  - Checkout a fresh copy of the repository
  - Paste those files there and try to commit again.
  - If this still does not work, then get help!!

Cameras

- Sony XCD-SX90CR
  - 1280 x 960 resolution
  - 15 fps

Viewing images

- Start rosrun
- Play back a bag file
  - `rosbag play -l <bag_file>`
- Start the default image viewer - provided through the ROS image pipeline
  - `roslaunch image_view image_view image:=/center_front/camera/image_raw`
Camera Image Examples

- There are examples for using the camera images
  - These examples build upon each other incrementally

- These examples are in the `vision_examples` package.
  - `rosdev art_examples`
  - `svn up`
  - `rosmake vision_examples`

- For the following examples, you should have a `roscore`
  running with an appropriate bag file
- A bag file is available here:
  - You can get this bag file using the `wget` command

get_cam - Subscribe and Get Images

- File: `src/get_cam.cc`
- To run:
  - `rosl-run vision_examples get_cam compressed`

- Line 86: This node subscribes to `center_front/camera/image_raw`
- Line 18: The subscription calls the call-back function `processRightImage`
  whenever an image is received
- Line 27: This function prints out the message "Image Received"

- We store compressed images in our bags, so you have to use
  compressed. On the car you can also use raw.
- The uncompressing is done by a ROS package called `image_transport`
  (Lines 83 and 86)
- `image_transport` is not really necessary, but it automatically
  compresses/decompresses as necessary.

What happened here? - Master

![Diagram showing the master setup with nodes roscore, rosbag_play, get_cam, and their connections]

What happened here? - Communication

![Diagram showing the communication setup with nodes roscore, rosbag_play, get_cam, and their connections]
**OpenCV**


- **OpenCV (Open Source Computer Vision)** is an open source implementation of a number of popular computer vision algorithms
  - Face Detection
  - Pedestrian Detection
  - Local Feature Extraction
  - ...

---

**cv_bridge**


- Converts between ROS Image message formats and OpenCV image formats
  - ROS -> OpenCV
    - `toCvCopy()`
    - `toCvShare()`
  - OpenCV -> ROS
    - `toImageMsg()`
**view_cam** - View Images (*highgui*)

- File: `src/view_cam.cc`
- To run:
  - rosrun vision_examples view_cam compressed
  - meld src/view_cam.cc src/get_cam.cc
- Line 21: use `cv_bridge` to convert between `sensor_msgs::ImageConstPtr` and `cv::CvImageConstPtr`
- Line 25: `cv::Mat` is the OpenCV image format

**gray_cam** - Convert to gray-scale

- File: `src/gray_cam.cc`
- To run:
  - rosrun vision_examples gray_cam compressed
  - meld gray_cam.cc view_cam.cc
- Many vision algorithms use gray scale images
- Line 28: Create image to store the gray-scale version
- Line 29: Convert to gray-scale
- Line 31: Display gray-scale image instead

**edge_cam** - Find edges (*canny*)

- File: `src/edge_cam.cc`
- To run:
  - rosrun vision_examples edge_cam compressed
  - meld edge_cam.cc gray_cam.cc
- Line 32: Create image to store edge version
- Lines 33-35: Parameters for the canny algorithm
- Lines 37-40: Comment trying to explain these parameters
  - [http://opencv.jp/opencv-2_2_org/cpp/imgproc_feature_detection.html#Canny](http://opencv.jp/opencv-2_2_org/cpp/imgproc_feature_detection.html#Canny)
- Line 41: Call to canny algorithm
- Line 43: Show `edgelmage` instead of `grayImage`

**mult_right_cam** - Display multiple images

- File: `src/mult_cam.cc`
- To run:
  - rosrun vision_examples mult_cam compressed
  - meld mult_cam.cc edge_cam.cc
- Lines 101-107
  - Create, resize and move 2 windows
  - namedWindow - [http://opencv.jp/opencv-2_2_org/cpp/highgui_user_interface.html#namedWindow](http://opencv.jp/opencv-2_2_org/cpp/highgui_user_interface.html#namedWindow)
  - cvResizeWindow and cvMoveWindow do not have highgui c++ alternatives - so using the old C API.
  - You can use `cvDestroyWindow` to kill the window as well
- Line 44
  - Display the second image
**publish_right_cam - Publish the edge Image**

- File: src/publish_right_cam.cc
- To run:
  - rosrun vision_examples publish_cam compressed
  - rosrun image_view image_view image:
    - /center_front/camera/image_edge
  - meld publish_right_cam.cc edge_right_cam.cc

- Line 16: Add a publisher
- Lines 44-49:
  - Line 44-47: Create a new cv_bridge::CvImage object. Copy header, set encoding and copy image (image header only - the data is shared)
  - Line 49: Publish the image
- Line 110: Advertise the published image