Vision Calibration

15-491: CMRoboBits

1 Collecting images with "Camera" behavior

The *Camera* behavior allows you to use the robot's tail as a joystick to control which direction the robot's head points. Once you position the head at the desired angle, you record a picture of what the camera sees by pressing the front head button.

- Edit /memstick/config/spout.cfg and ensure that all of the constants in this file are set to zero. These constants are described in the Setup document. They do not come into play here beyond being turned off.
- Edit /memstick/config/run.cfg to run the behavior "Camera" and boot the robot.
- Unpause the robot by pressing its back button.
- *Gently* move the tail and get a feel for how the head responds.
- To take a picture, hold the front head button until all of the face LEDs light. Release the button once the LEDs light.
- After taking all of your pictures, press the back head button. The head will move from side to side until the robot shuts off. *Wait for it to stop making sounds and for the chest light to go off before removing the memory stick*

Your pictures can be found in /memstick/log (once you mount the memory stick).

When taking pictures for calibration, you should take several shots of each object from different distances, different angles, and in different amounts of shadow. You want to have examples from all of the conditions in which your robot will run in the future. (This does need to be balanced against the amount of time that you are willing to spend hand labeling images and the diminishing returns as more and more pictures are added) For this assignment you should label at least 6. Label enough to get good results.

2 Extracting images from the log

The file /memstick/log contains a binary image of several different streams of debugging information. A utility program is needed to extract items that were serialized into the different streams.

- Run "make" from ~/dogs/util/tile and ~/dogs/util/log_processing/log_extractor to build necessary utilities.
- Make a temporary working directory and cd into it.
- Mount the memory stick, copy /memstick/log to your temporary directory, and unmount the memory stick.
- run "$DOGROOT/util/log_processing/log_extract/logextract ./log $DOGROOT/util/log_processing/log_extract/extract_configs/vis_raw.cfg" This command produces several groups of files: two sets of PPM [image] files and a corresponding set of INF files. The PPM files with names that start with "c" are classified images - they have been segmented with the current set of thresholds on the robot. They are useful for evaluating how good those current thresholds are, but not for calibration. For calibration, you need the PPM files that begin with "i". Those are the raw images. The INF files contain information about the head angles of the robot when the image was taken.
- You may look at PPM files by typing "xzgv i*.ppm &". Press "z" to zoom in, space to advance between pictures, and "q" to quit.

3 Labeling images

Labeling each one of the images separately is difficult. We will use the tile utility to generate one large image from all of the separate files and convert from YUV to RGB to make them easier to understand.

- Run "~/dogs/util/tile/tile 4 i*.ppm" This will generate two files: imagergb.ppm and imageyuv.ppm. The first is a tiling of the input images converted to RGB and the second the same tiling left in the original YUV color space. You will label the former with color classes in an image editor. The later file is used when generating thresholds.
- Copy ~/dogs/util/thresh/colors.txt to your working directory. This file contains the RGB values that you will use to label symbolic colors in imagergb.ppm (which you should Save As label.ppm when editing).
- Run "gimp imagergb.ppm &"
- Right click on the image, choose Layers — Layers, Channels & Paths

- Create a new layer (bottom left button) Keep this dialog open. You draw only on the currently selected layer. For calibration, draw on the top one (New Layer) and preserve the RGB information in the Background layer as a reference. It is helpful to turn down the opacity of the top layer while working so that you can see through the layer that you are drawing on and trace objects visible in the Background layer. (Remember to bring Opacity back to 1.0 before exporting)

- Label the pixels in New Layer according to the symbolic colors listed in colors.txt. That is, color all background pixels black (0, 0, 0), all green pixels green (0, 128, 0), etc. (These values are RGB triplets) Color regions that you don't want to use for training grey. As a time saver, you may fill the whole top layer with grey and only label the interesting regions (being sure to label some background as well as the interesting bits).

- To actually draw, use the pencil tool. Select a 1 pixel brush by clicking on the brush shape (initially a large, black circle) in the lower right of the toolbox window. To change the current drawing color, click on the foreground color (initially black) in the lower left corner of the toolbox window. A useful trick when drawing in to make a single dot with the pencil, move the cursor to a different location, and hold shift while left clicking again to draw a line between the initial dot and the new cursor location. Also, if you completely bound a region with lines generated with the pervious technique, you can use the flood fill too (bucket spilling paint icon) to fill in the bounded region.

- One time saver is to make a custom palette in Gimp with all of the appropriate colors. To do this using the wonderfully intuitive Gimp interface (this is sarcasm), go to the Tool Box window, choose File — Dialogs — Palette. Click the Edit button. Choose New from the right side of the edit dialog that appears. Pick a name (I like dogs). Right click on the grey area to the left. Choose New to add a color. A bar should appear. This bar contains a single color box at the far left side. However, since the default color is black and the background of the bar is black, you can't actually tell by looking. If you right click on the far left side of the black bar, you can choose edit to set the color. Add other colors by right clicking on the black bar and selecting new. Then right click on each new entry to edit it and set it to the appropriate color. Hit Save when you are done. You can now click on each entry of your new palette to set the drawing color to the appropriate value.

- Once you have a labeled image, make sure the top layer is selected and the opacity is set to 1.0. Right click on the image and choose File — Save As — label.ppm. Choose export and raw when it prompts you.

4 Running thresh on your labeled image

Now that you have color class labels for each of the pixels in the sample images, you will use the thresh program to generate a threshold map file from the labeled data.

- *There is one big caveat when running thresh - it saves intermediate results in a file called cache.dat. You must erase this file any time you change the examples in any way. When in doubt, erase it - it only takes a few seconds to regenerate with a fast processor.*
- Edit colors.txt. Change the path and filenames at the bottom to reflect the location of your imageyuv.ppm and label.ppm files. Also give it the path of a file to test the thresholds against (generally imageyuv again, although good ML researchers test on examples the algorithm was not trained on). In addition, in the test section, give it an output file to write the results of segmenting the test file using the new thresholds.
- Run "$DOGROOT/util/thresh/thresh ./colors.txt"
- Scroll up in the output and find the table labeled "NORMALIZED". This is a break down of how pixels were classified. Each row of this table represents all of the pixels of a given class. Row 0 represents background. Each column represents the percentage of examples that were classified with a given color. So, if the classifier was perfect, the table would be a diagonal row of 1's as every instance of every color would be labeled with the correct symbolic class. (Read down the rows of colors.txt to figure out which row corresponds to which color - they are in the same order)
- Scroll up to the very top, just below the thresh command and check the line that begins with Adding ???x??? image. This line tells you how many pixels were added for training, how many of those are being ignored, and how many errors there were. Errors occur when pixels in label.ppm are colored with something other than a valid symbolic color. This could occur if anti-aliasing is turned on with your drawing tools, the opacity of the image was not 1.0 when you exported, or because you used an incorrect color when labeling the image.
- Also view the output file you listed in the testing section of colors.txt using gimp or xzgv. You can visually pick out trouble areas that are being misclassified so you know what to take suplemental photos of.

After you run thresh, you have the actual threshold map. It is located in a file called *out.tmap* Copy this file to /memstick/config/thresh.tm for testing on the robot or to ~/dogs/agent/config/config/thresh.tm and do a "stickit -a" to copy the file to the memstick if you're changing binaries at the same time.

5 Logging vision frames to test your new thresholds

You can periodically write vision frames or send them over the wireless network to ChokeChain to test how well your new thresholds are working.

- Edit /memstick/config/spout.cfg again. This time set the *dump_vision_rle* parameter to a positive value. This option controls how often the robot records a segmented image. A value of 1 causes it to save a picture every frame. A value of 25 causes it to save every 25th frame (about one picture per second). Zero turns this option off. When sending over Wavelan, it does not matter if you send too frequently. However, if you intend to log be aware that there is limited space on the memory stick and you will fill up the log quickly if you take pictures too rapidly.
- Remember to unmount the memory stick after editing the file.
- Edit ~/dogs/util/choke_chain/Makefile and set GRAPHICS := 1. Do a make to rebuild chokechain2 with the GUI on. Connect to the robot exactly as in the last assignment to see RLE images from the robot as it runs.
- If you wish to save the images in a log, edit /memstick/config/run.cfg and add the behavior LogWrite with a higher priority than your behavior. LogWrite allows your behavior to run normally and waits for a back head button press. When the back head button is pressed, LogWrite will save the log to the memory stick and shut off the robot just like Camera did. Extract the pictures from the log just like with Camera, except use vis_rle.cfg instead of vis_raw.cfg as the extraction configuration file. Also, you can access raw vision frames this same way, except you must activate dump_vision_raw in spout.cfg beforehand (and use vis_raw.cfg when extracting).