Training the Human Detection Pipeline

# Introduction

The human detection pipeline consists of 5 ROS nodes or nodelets.

1. Consistency
2. HaarAda
3. HaarDispAda
4. HaarSvm
5. HogSvm

Each node in the pipeline takes as input, the intensity image, and the disparity image from one of the two 3D sensor systems and a list of regions of interest (rois). The exception to this is that the consistency node which appears first in the pipeline operates without the rois in detection mode. Rather, it generates a list of rois which are consistent with the possibility of there being a person. The job of each node is to pare down the list to include only those rois which are believed to contain a person. Each node is implements a different machine learning algorithm to accomplish this task. Nodes earlier in the pipeline are generally faster, but less accurate. They are designed to quickly weed out the individual rois that do not contain a person thus reducing the work load on the later, more computationally expensive but more accurate classifiers. The training process for each node has been standardized to a single launch file (train.launch) within each classifiers package.

In order to use the train.launch file for any node/nodelet, one must first collect data and manually label it using the labeling GUI. In addition, false alarm rates may be significantly reduced by re-training each node by including a data collected by running data which is known not to include people through the entire system. In this way negative examples are selected that are difficult to classify which is better than the random examples generated by the labeling GUI.

# Collecting and Labeling Training Data

The process for creating training data involves five steps:

1. Running one of the two collection launch files from the System\_Launch package
   1. collect\_stereo.launch
   2. collect\_kinect.launch
2. Labeling the data using the GUI from the labeler package. This is an executable, not a ros node.
3. Training all the nodes/nodelets using their individual train.launch script
4. Running the System\_Launch script (collect\_negs.launch) to generate negative examples
5. Re-training each node/nodelent using the individual train.launch script

Each of these launch files has default arguments indicating the location/source of input and output data. For example, the train.launch scripts all take the following arguments:

1. directory (indicates where to get the labeled data)
2. classifierDir (indicating where to store the trained classifier)

The value of these arguments generally change depending on whether the training data comes from the Kinect or from the stereo pair, and to indicate the difference in location of original training data, and training data which is a combination of manually labeled images, and negative examples.