Baxter Image Recognition Class Summary

**Setup for Block Recognition Project**

Follow the steps from the Rethink Robotics wiki to setup your computer for use with Baxter: LINK

Download the files from the github repository: LINK

Move all files from the “src” folder to a catkin workspace. For example ~/ros\_ws/src/baxter\_tools/scripts/.

Place blocks on the edge of the table near the desired box. PICTURE (maybe)

Run “./baxter.sh” to initialize the connection.

Run “rosrun baxter\_tools enable\_robot.py –e” to enable the robot.

Run “rosrun baxter\_tools master\_controller.py config” to setup the configuration file.

Run “rosrun baxter\_tools master\_controller.py” to begin detecting and sorting blocks.

**Master Controller Class**

**Attributes**

right\_camera : image\_receiver() class initialized for the right arm’s camera. Enabled

left\_camera : image\_receiver() class initialized for the left arm’s camera. Disabled

head\_camera : image\_receiver() class initialized for the head camera. Disabled

move : move\_controller() class initialized for the right arm.

image\_processor : image\_processor() class used to process images.

block\_list : A dictionary where each key is a block color and the value is a list of Block objects of blocks of that color. For example the purple block list is block\_list[‘PURPLE’].

box\_pose : A dictionary where each color is a key and the value is the Pose used for the box of that color

**Methods**

\_\_init\_\_(self, setconfig=False)

* (optional) setconfig : Boolean, set to true for configuration mode (configure box locations and table height)
* Initializes all attributes and sets up the robot so that the right arm gripper / camera is used to sort blocks.

update\_home\_pose(self, pose)

* pose: New home position that is inputted into the move controller and image processor.

get\_home\_pose(self)

* Takes an image from the home position and inputs it into the image processor class

find\_blocks(self)

* Calls get\_home\_pose(), then adds a list of block objects of each color in block\_list

are\_blocks\_near(self, block)

* block: The block object that is being targeted
* This method checks if any other blocks are nearby the targeted block. Could be used to avoid block clusters. Currently unused.

get\_blocks(self, trials=2, miss\_per\_trial=2)

* (optional)trials : number of pictures to be taken from the home position before failure
* (optional)miss\_per\_trial : number of blocks missed before a new trial is started (by taking a new picture from the same home location)
* This is the main function for finding blocks. Set the home position using update\_home\_pose() then call this function to find blocks from the current home position.

Main method in file accessed with no arguments i.e. “rosrun baxter\_tools master\_controller.py”

* A while loop that switches between 6 different home positions and calls get\_blocks() from each position. This while loop runs continuously until killed by the user.

Main method in file accessed with arguments i.e. “rosrun baxter\_tools master\_controller.py -c”

* Sets up the config.txt file for the current environment. Finds box locations and configures the table height.

Commented out in Main method

* Test that can be used to get block classification color range accuracy.

**Move Controller Class**

**Attributes**

home\_pose : Pose object of the current home position

ns : String for the Kinematics solver subscriber topic

iksvc : rospy.ServiceProxy(self.ns, SolvePositionIK) object for the kinematics solver

ikreq : SolvePositionIKRequest() class.

hdr : header used for the PoseStamped object

arm : baxter\_interface limb controller initialized for the specified arm.

gripper : baxter\_interface gripper controller initialized for the specified arm.

table\_height : height of the table, this is set by the configuration infrared callback.

infrared\_topic : topic for the infrared subscriber.

infrared\_sub : variable to hold the infrared subscriber.

**Methods**

\_\_init\_\_(self, arm)

* arm : “right” or “left” used to specify the arm being initialized.
* This function initializes all of the attributes of the object for the desired arm, sets the movement speed to the maximum, and calibrates the gripper.

move\_to\_pose(self, poselist, move=True)

* Poselist : a list of Pose objects to be inputted into the IKSolver.
* (optional)move: Boolean that indicates if you actually want to move to that point. This can be set to false to check if there exists a valid joint position without actually moving to it.
* This is the main movement function that takes a list of Pose objects and solves for the joint position of each Pose and moves to each joint position in order. This function is called by several other move\_controller functions.

**Image Receiver Class**

**Attributes**

**Methods**

**Image Processor Class**

**Attributes**

**Methods**

**Block Class**

**Attributes**

**Methods**