1. **Go through this description and create a flowchart of your expected project.**
2. **Use this site for some help with Python**
   http://www.programcreek.com/python/index/572/rospy

Detailed Design: Baxter Sorts Colored Balls Anjana, et. Al. Capstone 2015

Original From Boyce Mullins Code – modified

On our 6838 Web Site Read: Helping BAXTER See CS 543 Final Report

Methodology (in detail) –

1. Detect the ball :
   (a) Move BAXTER's arm to some initial position.
   (b) Look for the ball by scanning the environment through the motion of BAXTER's right hand.
   (c) Get images from the camera located on BAXTER's right hand.
   (d) After getting the image, convert the RGB (Red, Green, and Blue) image into an HSV (Hue, Saturation, Value) image.
   (e) Next, convert the HSV image to a binary image using the OpenCV inRange () function. Using this function allows us to specify which color to look for in the image.
   (f) Using our binary image, find the largest contour, or white areas, in the image. Our desired object is the largest contour.
   (g) The center of the largest contour gives us the location (x, y) of the object.
2. Center the ball in the image:
   (a) Choose an alignment point within the image as the point where the center of the ball should be aligned.
   (b) Using the alignment point described in the previous step, the image was divided in a grid-like fashion into four quadrants. The first quadrant was the top-right portion of the image and subsequent quadrants were numbered in a counterclockwise manner. After determining which quadrant the center of the ball was currently in, move BAXTER's right hand closer to the alignment point.
   (c) Repeat the previous step until the (x, y) location of the center of the ball was within some tolerance of the (x, y) location of the alignment point.
3. When the ball is at the alignment point in the image, move BAXTER's hand appropriately, and grab the ball.
4. Change the desired color to the color of the appropriate bucket.
5. Repeat Steps 1 and 2 to find the appropriately colored bucket.
6. Using the algorithm (mentioned in reference), calculate the distance to the appropriate bucket.
   (a) Calibrate the camera using the known size of the target object (s) and a preset distance (z) from the camera. After placing the target object at z, take a picture of the target and find the apparent width in pixels.

(b) At run time, get the image of the target and compare the current apparent width of pixels to the calibrated width of pixels.

7. After BAXTER has determined the distance to appropriate bucket, we navigate his arm over the object, and have him drop the ball.
8. Repeat the above steps until all balls are sorted.

Working of code in detail:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

So initially python imports the libraries, creates objects to the functions, and then checks for if__ name__=='__ main__': and starts execution from there.

In **if__ name__=='__ main__':** block:

- BM_node is initialized.
- Baxter is disabled and then enabled to ensure no confusion.
- Interface cameras, close left hand camera and right hand camera.
- Set resolutions.
- Calibrate grippers.
- Redirect it to main block.

In **def main (args):** block:

- Redirect it to BM_project class.
- Set path for inverse kinematics portion.
- Initialize inverse kinematics service, request, header, and initial positions of arms.
- Set color count to zero.
- Assign the colors that are to be detected to an array named ball colors.
- For the colors in ball color, set the initial parameters as zeros and false, and increment the color count.
- When Baxter is in initial starting position(for search i.e. in safe position)
    - Hasn't found any object yet.
        - If didn't grab any ball.
            - Keep searching for ball.
        - Else if grabbed a ball.

            - Search for bucket.
        - If found any exceptions, report them.
        - If response is valid, set those positions to limb joints and get the current position of the arm, position error as well as orientation error.
        - Get the total position and orientation error and move arm to the position specified in limb joints.
    - Else if mycount (variable assigned to set a movement depending upon its result to search for the ball) ==0, move the arms to final position where we found the ball and increment mycount to 1.
    - Else if the object is not in center of view, move the arms until the object is in center of view by adding or subtracting the corresponding step size with respect to the quadrant.
    - Else if the ball is not grabbed yet set the z position of the arm to the grab height.
    - Else if there is no safe lift after ball is grabbed, set the z position of the arm to a value and make the safe lift after grabbing ball true.
    - Else if looking for bucket, set the blue ball to be dunked in orange bucket and move to the bucket position from the assigned values.
    - Else if depth of the object from the camera is not yet calculated, calculate the focal length and add it to the x position of arms to reduce the offset between the actual position of the object and the position of the camera is assuming.
    - Else if not in starting position, modify the x, y, z positions of arms with respect to the mycount variable.
- Append the inverse kinematics request positions to the right arm and handle any exceptions.
- If the response is valid, move the limb to the limb_joints position and get the current position of the arm.
- Calculate the position difference, orientation difference, and total difference.
- If the ball is in center of view, grab it.
- If the depth is calculated, dunk the ball in bucket and start looking for next ball.

In **class BM_project:** block:

- Define a function self to define our initial parameters using its object.
- Create windows for display.
- Set the grab height.
- Subscribe to the right hand camera.
- Save the initial image i.e. RGB.
- Convert it into HSV.
    - Hue range is from 0 to 180 degrees.
        - Divide it by 2 to place it in code.
    - Saturation and value range is from 0 to 255.
        - So convert it into percentage by diving the saturation or value by 255.
- And then to binary using inrange () function.
- If the ball is not yet grabbed find the contours of binary image.

    - If the area of contour is greater than contour tolerance specified, draw contours to object.
        - Set found object to true and draw bounding rectangles to the image
        - And if the bounding rectangle correctly fits the image, make the center of view to true
- After the subscription to the right hand camera is successful, the BM_project class and rest of functions in main will be executed concurrently.
- And the main accesses, the parameters that we defined using self-object.

Conclusion and Results:

Object detection on the basis of color is fast and easy to implement which requires no training of data. But however there are some difficulties in doing so such as including the variability in success based on lighting, difficulty in searching for multiple balls, depth detection is preset, all the background objects had to be covered in white (in order to not confuse Baxter with the surrounding objects like wires, tube light shadow etc.).

Baxter was able to successfully dunk blue ball in orange bucket in first trail and later modified the code and made Baxter sort orange ball from different multiple colored balls and dunk in the respective orange colored bucket.