Brendan Neal 1001160236|nealbre1 2018-11-05

CSC420 Assignment 4 Question 1

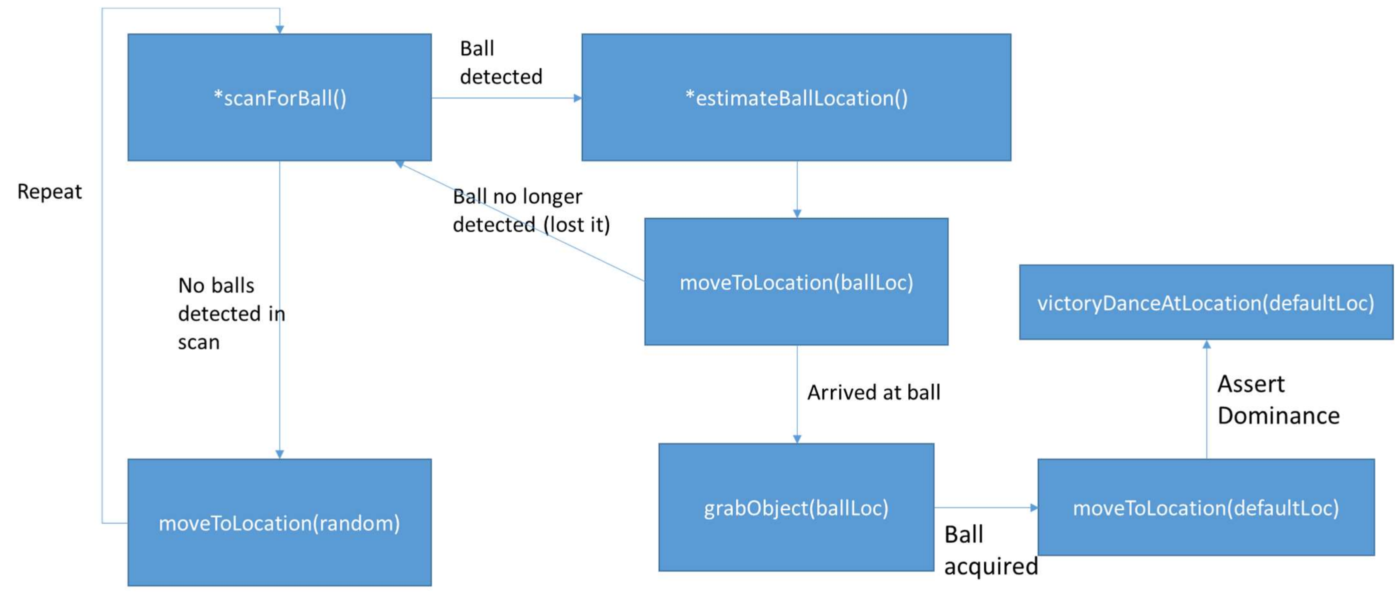
Part A: Brainstorming

- What will the robot encounter?
  - People (likely tennis players)
  - Tennis balls
  - Tennis net
  - Walls of tennis court
  - Floor of tennis court
- Training Data
  - Object Classification / Detection
    - Self-driving-car-esque training data (inputs are raw images, outputs are segmentation masks or bounding boxes)
    - Classes: everything mentioned above ("what will the robot encounter?")
- Robot World Model
  - Collection of (class, location) pairs (keep track of where everything is)
  - Robot's current 3D location
  - Size of tennis court (robot shouldn't try to exit the court / crash into walls)
  - Robot's default 3D location
  - 3D location of net (so robot can get around it)
- Robot Challenges
  - Don't run into anything (walls, people, tennis net, etc.)
  - Avoid moving objects (e.g. people moving around while robot is getting ball)
  - Ball may be out of line of sight from default location
  - Robot needs to find a way around the net

Assumption: the moveToLocation() method automatically avoids collisions. This method would have to use the model of the robot's world to plot paths around objects that the robot is likely to collide with.

Brendan Neal 1001160236|nealbre1 2018-11-05

Part B: Flow Chart



Part C: Pseudocode

```
detectObjects(img):
    Runs a trained object detection neural network on img to detect objects.
    Possible classes:
        People
        Balls
        Tennis Net
        Court Walls
        Court Floor
scanForBall():
    for each slice of 360 degree rotation:
        faceDirection(slice)
        img = takePicture()
        detections = detectObjects(img)
        for det in detections:
            if det.class is TennisBall:
                return img
    return None

estimateBallLocation(i1, i2, detections):
    focal_length, base_width = cameraParams
    disparity = patchMatchDisparity(i1, i2)
    z = focal_length * base_width / disparity
    x = x_coords * z / focal_length
    y = y_coords * z / focal_length

    ball_detection = (find ball detection in detections)

    # Use similar method to question 2D to find center of mass
    ball_location = center of mass of ball points using ball_detection, z, x, y

    return ball_location
```

Brendan Neal 1001160236|nealbre1 2018-11-05

Part D: Robot Eye Loss (BONUS)

Losing an eye/camera means that the robot can no longer use triangulation of two simultaneous images to calculate the 3D locations of objects. However, this can be compensated for using the following method:

1) Capture image I_1
2) Move ϵ meters to the left or right (in x direction)
3) Capture image I_2
4) Run triangulation between I_1 and I_2 using a base width of ϵ meters

This approach effectively captures two images from different perspectives and therefore partially emulates the simultaneous capture of 2 images by dual cameras. This method is very similar to how birds with non-stereo vision use head movement to capture multiple views to emulate depth perception. However, there are a few failure cases for this method:

- If the robot is unable to move a sufficient distance horizontally, then the effective base width will be very small or close to zero. If the base width is small enough, the relative error in the base width will be quite large. Since depth (z) is inversely proportional to base width, a large relative error from a small base width will result in large uncertainties in estimated locations.

- If the two images are very different (e.g. a person walks in front of the robot when it is capturing I_2), matching points to create a disparity image will fail. If a disparity image cannot be created properly, then triangulation cannot be used to estimate 3D locations properly.

- This method performs poorly if the robot has to continually estimate 3D locations while it is moving. Essentially, the method will force the robot to obey a zig-zag like path when moving so it can capture the two perspectives needed to estimate 3D locations. Obviously, such a path is inefficient and the robot will perform its job slower as a result.

- If objects are moving around faster than the robot can capture its pseudo-stereo images, the robot will be unable to accurately estimate their real locations. The robot will be more likely to crash into things if it cannot estimate object locations.