Mini Project 2 Journal

Brandon Davis
bdavis380@gatech.edu

***Abstract*—**In this report, I will cover the implementation details of my AI agent , and how it solves some of the Ravens Progressive Matrices problems. In milestone 1, I discussed some initial strategies in which I believed would suffice for an AI agent to perform successfully, "Generate and Test, and Brute Force Matching. Therefore, after the first implementation attempt, that is indeed my approach, for now. I am looking forward to the reader providing any insights and ideas, that would help my agent not only be more efficient, but robust.

1 HOW IT WORKS

1.1 How does your agent currently function?

At a high level, my agent reads in the problem figures, and separates the figures labeled with letters from the ones labeled with numbers, then converts figures A through C to images, using the opencv library. Note that it only supports 2x2 problem sizes for now so I did not bother checking for larger problem sizes. It then compares the images by determining if image A is similar to image B. If so, the answer has to be similar to image C, therefore compare each answer to image C until a close match has bee found. Otherwise, compare Image A to C, and do the same for Image B. This is the strategy I called Brute force Matching.

1.1.1 *Does it perform shape recognition or direct pixel comparison?*

To be able to compare images for matching, the agent utilizes the be following opencv methods:

cv2.threshold(...) - to convert the image to binary image for simplicity

cv2.matchTemplate(..) - to match objects form template image to source image

cv2.flip(..) - to rotate or flip images then compare

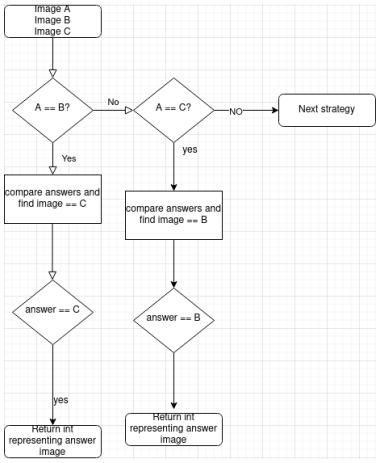

Figure 1—Brute Force Matching

1.1.2 *Does it generate a candidate solution and compare it to the options*

There is a case where Brute Force Matching does not solve the problem, therefore this posed as a need to utilize more image processing. For this case, I rotated image B and C then compared to A. This would tell me that, the answer choices will match a flipped copy of either B or C. As a result, if B or C, flipped, matches A, then flip B r C then compare that image to the answer choices using matchTemplate(), and return the result

2 HOW WELL DOES YOUR AGENT CURRENTLY PERFORM?

According to gradescope the agent completes at about 0.5 seconds, but only passes eight Basic and eight Test Problems. I susect the issues arise when the problem figures have nested shapes. Though I have experimented with detecting nested shapes using image contour features in opencv, the agent is not able to solve those problems just yet.

3 HOW DO YOU PLAN TO IMPROVE YOUR AGENT'S PERFORMANCE ON THESE PROBLEMS BEFORE THE FINAL PROJECT SUBMISSION

I plan to investigate and take advantage of more features available in opencv to help cover the many cases from any of the RPM figures. One case to cover would be nested shapes. As of right now, the plan is to count and detect the image contours and count the number of parents and children in the image. Moreover, the agent needs to know if an object within the image is either filled or not, this could be a use case for counting the amount of nonzero pixels in the

nparray representing the image. Also, I believe it would help if the images and transformations were somehow represented. Some initial ideas include construct objects that represent information about the images, such as, number of contours, area, shapes, location/position of shapes within the image, and its relation ship with the other images.

4 HOW DO YOU PLAN TO GENERALIZE YOUR AGENT'S DESIGN TO COVER 3X3 PROBLEMS INSTEAD OF JUST 2X2 PROBLEMS?

For all problem sizes to be supported, the it will have to check all of the images listed with letters and compare to the image at the head of each row. The agent, needs to understand the relationship between the images from each row, as well as each column. I will enhance the initial processes to store images with labels A-H and iteratively construct an object to store basic information about the image and its relationships to its neighbors.