RPM Project: Milestone 2
CS7637

Josh Adams
jadams334@gatech.edu

Abstract—The RPM Project: Milestone 2 journal is a place to reflect on the obstacles faced while improving the agent's performance. In the first milestone I discussed the approaches I would take to make an agent which would be able to solve the RPM problem. Some of those methods were utilized and others were just attempted.

1 PROBLEM SOLVING

My agent combines many approaches to come up with an answer for a given problem. What I have found works best is to implement small checks which make large differences. The agent processes every image and creates a dictionary of data, such as the contours and lines in the image as well as many other things. The first thing my agent does is to run a simple but quick check. An example of this would be to check the pixels in image A image B and image C. If each of those images have approximately the same number of black pixels, it would be expected that you could remove any answer choice which does not have a similar number of black pixels. While this does not generalize terribly well, it has been a method that works for reducing the possible answer for a problem. Even reducing one possible answer, increases my chances of reaching the correct one. The next thing my agent does is create a tracker for the possible answer choices which are currently available. As it progresses through many filters it takes votes for which answer choices should be removed. Once an answer reaches 3 votes total, with at least one strong vote, that answer choice will be removed from possible answer choices. These filters do have thresholds which reduce false positives, where it incorrectly labels an answer choice as a candidate to be removed. Filtering of images based on some weaker comparisons would require other stronger comparisons to be considered. An example would be trying to filter images based on the angles of the lines, this would require at least some number of angles before that comparison is considered valid.

Over each filter a relational heuristic is being tracked for the images. For example, if image A and image B both have detected corners which are similar, the heuristic associated with the AB relationship will be incremented. Conversely if image A and C do not share relational corners the AC heuristic will be decremented. The next filter uses 'HoughLines' to find lines in the images and then checks the angles within the images and votes for removal of possible answers which do not show a strong relationship. The agent then looks for circles in the image using 'HoughCircles'. The last set of filters checks for lines within an image to create an idea of what shapes could be in that image. An example of having both a 0-degree and a 90-degree line means this image could contain a square, but also could contain an octagon or many other shapes. This coupled with the number of those angles within the image help to isolate the possible shapes in the image. My agent is not incredibly efficient but is able to solve all the problems within a few seconds. There are many places for improvements as I do have a few nested for-loops, but my focus has not been efficiency, rather on being correct and robust. I also have many redundant checks and rechecks which I will be able to consolidate later for the agent. For example, I verify all previous relationships between images for each new image tested. Luckily, I was able to make those checks very fast, but they are still needlessly tested many times. My agent currently works well on problems where there are clear relationships between images.

2 CHALLENGES

I have encountered many challenges mostly with trying to determine relationships of shapes and how they change between states. One problem dealt with association of certain angles of lines with certain shapes. Such as a ~30-degree or 45-degree angle with triangles. I have found that this is not a robust solution but has helped in reducing answer choices. An example of this working but not as intended. I thought I detected a triangle based on the specific angles found in the image. I removed answer choices which did not share those angles. I thought it was due to a triangle being in those images, but it was a different shape entirely. My agent currently can apply its simplistic problem-solving methods to 3x3 problems effectively. The reason they work for 3x3 is that they break down the comparisons between the two images into small easier to manage checks. It then expands those results for successive comparisons. In a 2x2 matrix it would compare image A and image B then image A and image C. For a 3x3 it would compare A to B,D and E and follow the path most similar based on a relational heuristic I created. This heuristic is based on the connections between images which is described in the Problem-Solving section above.

3 NEXT STEPS

Going forward I will be implementing better template matching coupled with the previously used contours. This will start at the smallest contours found in the image and create a template off that and try to detect that within the various images. I will also try and implement more simplistic methods of testing various image transformations. Such as applying reflections or translations and just comparing the resulting image.

4 FUTURE HELP

Places where I could use help would be more robust methods for determining shapes and relationships of those shapes. My current methods are not terribly robust as they cannot account for situations which I have not encountered and put in place methods for handling. I am more than willing to take any suggestions offered to improve my agent.

5 REFERENCES

1. OpenCV https://opencv.org/
2. HoughLines https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html
3. HoughCircles https://docs.opencv.org/master/da/d53/tutorial_py_houghcircles.html
4. FindContours https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga95f5b48d01abc7c2e0732db24689837b