RPM MILESTONE 2

David Huang
dhuang316@gatech.edu

***Abstract*— The purpose of this text is to describe my agent's design and performance for solving the Ravens Progressive Matrices questions for set B. Currently I want to include more logic and generalization that will be robust and solve questions more cleanly.**

1 HOW THE AGENT WORKS

My agent uses direct pixel comparison currently with a generate and test method to check for similarity between images. For exact matches I use numpy where I do grey scaling and normalization. I normalize the pictures that I'm comparing and then check to see if they're equal with these normalized values. Some images may not be completely identical, so I use some thresholds to check for close matches with opencv's orb function. With orb I can detect and compute key points with the computeAndDetect function to see if the images after a flip or rotation match to a certain threshold with the possible answers. This would be the generate state, I use many different functions such as cv2.flip between horizontal and vertical with values of 0 for horizontal and 1 for vertical then check if the key points from orb meet a certain threshold which is the testing stage that match close enough with the transformed image which for this set is the image B. The thresholds are very sensitive and I'm not sure if it is a robust solution however, it passes the testcases consistently on gradescope and has correct answers locally so far, so it seems good enough currently. To address deletion, I use shape detection where I store the string of the name of the shapes from image A into a list, I detected the shapes with the findContour and approxPolyDP functions from opencv and then use the approximated contours length from approxPolyDP where 3 is a triangle 4 is square, etc. and then I check to see if any removal is done from image A to image B within the list that is created and I run a loop to check every possible answer to see if the deletion occurs in the possible answers and if it matches with C's shapes list after deletion for example if the list of C has [circle,triangle] then possible answer could be [circle] since triangle

would be deleted the deletion factor is accounted from images A to B where triangle was deleted in B.

1.1 Performance

Performance to me is correctly answering basic problems and tests within a high level of accuracy, I would say high is 90%. Although my code is high accuracy for the basic problems that are solved, the test questions are not as accurate therefore the solution I have now can have major improvement. Currently the performance of my agent solves 7 basic questions and between 5 to 4 tests on set B. I expect the tests to pass 5 to 6 compared with the 7 correct basic questions. This shows that my current code isn't performing as expected and will require better techniques for increased accuracy.

2 STRUGGLES

The agent performs well on questions that have a lot of similarities since I only accounted for flips and exact matches and deletion. The agent I have right now does struggle to get good enough similarities sometimes but I'm not sure how to handle the thresholds well enough to be robust for the test problems. There are some images that don't have complete similarities after a transformation therefore I couldn't use the numpy normalization I mentioned to solve every problem that had a transformation. This led me to use the orb solution mentioned above but it also led to uncertainty in the confidence of my answers. Some answers may be wrong as the threshold isn't the same very every question which is reflected in the performance section, so it will lead me to either improve the current method if possible or find an alternative method. The shape detection function that I used has some issues as well considering that the length of circles varies and it could mix in with other shapes and misidentify a circle as when it should be another shape. I ran into the problem for detecting heart shapes, but I was able to identify it properly after some checks with the length of the approxpolyDP function, however again I'm not sure if this solution is general enough.

3 EFFICIENCY

The runtime of the agent is around 2.2 seconds which shows that it's relatively fast. However, the agent slows down when loops are run to determine which

possible answer for certain transformations match. I noticed the runtime increased from .45 seconds to 2.2 after the addition of the loops mentioned. This generally shows that my agent will take longer to search for the correct answer given more states to check.

4 PLANS TO IMPROVE

The plans I have to improve my current RPM solution consist of thinking of a strategy that can use means end analysis with some weighting heuristic. I plan to apply weights to shapes that match the transformations that are applied to get a higher accuracy than just relying on pixels matching or not. This will be done with the current functions I have but more generalized in the sense that if the shapes match in the list of shapes mentioned before higher weight will be applied. I originally believed that generate and test would be the solution for this project however I realized that there are other strategies that will produce better results in terms of runtime if applied correctly. This weighting heuristic I have in mind is one of them since it will be easy to add high weight to matching shapes and then to pull out the maximum to find the answer with high accuracy. I will most likely store the weights in a dictionary with the key being the possible answer's choice selection and the value being the weights. This will allow me to keep track of the highest weights in a list with a loop that will allow me to use the sum function and pull out the highest weight. I originally stated that I wanted to use memory and store previous answers into a dictionary that can be called in constant time if it runs into a similar or same question and I still want to apply this method and possibly allow my agent to learn from these previous questions however I don't have a lot of ideas on how to actually apply this.

5. 3X3 GENERALIZATION

I plan to generalize my agent by creating a function that can take in an input of nxn matrices that will be able to check if it's a 2 x 2 matrices or 3 x 3 matrices and account for the differences considering 2 x 2 matrices will have 3 images to account for and 3 x 3 will have 8 images to account for. Basically, if problem type n = 2 then it's a 2x2 and if n = 3 then it's a 3 x 3. There may be a more general way to connect the matrices and see if the length of dictionary figures exceeds a certain amount to label it as 3x3 matrices, but it doesn't seem to make much difference. Another way that I have thought of to generalize the 2x2 matrix is by

separating the possible choice answers which are integers and the letters that are the images given. After the separation I can check the length of the letters and the larger one would be 3 x 3 matrix since there are more images to analyze as mentioned before.

6. FEEDBACK

The feedback I hope to receive is if I'm going in the right direction in terms of how my agent works although I'm using these opencv functions I'm wondering if I don't have a general enough approach to succeed in solving the matrices problems efficiently. Although the functions are handy there may be better solutions that utilize them but not fully. If so, should I keep this in mind or should I continue with what I have? Are there better approaches in terms of finding similarity after a transformation? I know that there are some mathematical ways to find contours and pixel shapes but I'm not sure if they would make much of a difference efficiency wise. If so, then I would like to hear how. Is my generalization for 3x3 a proper way to go about it? I'm not too sure if there's a better way to get an input like the 2x2 matrix and be able to solve the problems without an if condition and then use the same function for 3x3 matrices. If there is I would like to hear how one would go about it? For my plans to improve I wonder if this technique of a weighting heuristic with means end analysis that I am considering will achieve the high performance I want? Is there any issue with applying weights to the shapes that match or is the actual implementation I have in mind not general enough to fit for all potential cases that can come up in 3x3 matrices as well? Is it possible to actually apply the memory and learning in this project? Is it worth it to even attempt it?

7. CONCLUSION

Overall, the current results I have are sufficient for me, this milestone let me explore and see how I can utilize the opencv functions to fit for this project. I would have liked to have a more robust and general solution that applied a weighting heuristic, but it took a while for me to understand what I needed to pass the basic and test questions. I plan to revisit set B after the gradescope submission time ends to try to implement the plans to improve. I felt like this was a great learning experience for image processing and computer vision. I'm curious to see how other people decided to create their agents and which methods were used.