RPM Milestone 4 Journal

Muhammad Uzair Shahid Syed
msyed46@gatech.edu

1 HIGH LEVEL AGENT FUNCTION

In Milestone 3, I wrote "At a high level, the RPM agent attempts to use dark pixel ratios, percentage of pixel changes between two images and mirror transformation to identify a solution for the 3 by 3 matrix. The agent simply picks the answer from the answer set which is closest to a particular transformation or change in pixels given a certain threshold." This mostly holds true for Milestone 4 except for the addition of frame generations, bit-wise operations and shape detection. The agent uses the same methods but more refined to get a better performance.

In this refined version, agent attempts to understand the relationships by generating frames for images in rows and diagonally. This gives the agent an understanding of how pixels, dark pixel ratio (DPR), image transformation has been affected between two images. Then the agent goes through the following functions one at a time:

- Finding if images in first row are exact and images in the second row are exact. If that is the case then the answer will be an exact image of the images in the third row
- Checking to see if images A and C are mirrored, and if images D and F are mirrored the same as A and C. If that is the case then the answer will be a similar mirroring of image G.
- Using DPR and pixel relationships between images in a row
- Using DPR and pixel relationships between images diagonally
- Using bitwise operations between images - this can also be checked for images in a row or diagonally but kept as a separate function to distinguish agent operations
- Lastly, determining number of shapes, type of shapes and patterns

The first two bullet points are self explanatory, I will use this opportunity to explain the other inner workings of the agent at a high level.

1.1 Row relationship

At a high level, the agent uses the pixels and dark pixel ratios with a threshold of 5 percent similarity to find an answer. For example, the agent would attempt to see if the average pixel change in both row 1 and row 2 are very similar with a threshold of 5 percent. If that is the case, the agent attempts to find an answer and tests it by ensuring the average pixel change between row 1, row 2 and row 3 are all within 5 percent.

Another example would be of the agent finding the pixel changes between AB and BC within an 8 percent similarity. This means that there is a consistent increase or decrease from A to B and then B to C. If the same holds true for second row images, then the agent finds an answer and tests it to ensure the same relationship is seen in the third row.

Further to the previous example, if an agent identifies the absolute value of the pixel changes between AB and BC is within an 8 percent similarity, but their raw values are opposite signs then it is possible the images are moving inward or outward in a consistent manner. No additional pixels are being added rather the movement and intersection of pixels is causing a consistent change in pixels. If the same relationship is seen in the second row, then the agent finds an answer and tests it to ensure the same relationship is detected in the third row.

Lastly, the agent uses DPRs with a 5 percent similarity threshold and attempts to find relationships in each row. If A to B DPR is same as D to E DPR and G to H DPR, and B to C DPR is same as E to F DPR, then the agent can understand there is similarity in each row and there is DPRs that are consistent. Using these relationships, the agent finds an answer and tests it to ensure the same relationship is seen in the third row.

1.2 Diagonal relationships

Similar to row relationships, the diagonal relationship uses same principles and thresholds to find the answer. However, this time the agent checks DPR and pixel changes in a diagonal fashion.

For example, if pixel change or DPR relationships between A to E and B to F is similar, and D to H and F to G is similar then we can deduce that the E to answer will be similar to D to H. The agent would find an answer and test it to ensure

the diagonal relationship identified is present.

Similarly, if there is no pixel change between A to E, B to F and D to H, then we can deduce that all diagonal images are the same. Therefore the agent finds an answer and tests it to ensure answer image has no pixel change with images A or E.

Other diagonal relationships are present as part of the next two subsections.

1.3 Bit-wise operations

In the bit-wise operation function, the agent finds an answer using logical operators AND, OR, XOR, or pixels combinations within the rows. Some of the combinations that are used by the agent are:

- Pixels in (A - B) = pixels in C, and pixels in (D - E) = pixels in F. Therefore, agent finds an answer by using pixels in (G - H) to equal pixels in answer
- Pixels in (A + B) = pixels in C, and pixels in (D + E) = pixels in F. Therefore, agent finds an answer by using pixels in (G + H) to equal pixels in answer
- Pixels in (B + C) = pixels in A, and pixels in (E + F) = D pixels. Therefore, agent finds an answer by using pixels in (H + answer) to equal pixels in G
- Image A AND/OR/XOR image B equals image C, image D AND/OR/XOR image E equals image F. Therefore, agent finds an answer by using image G AND/OR/XOR image H to equal answer

Other combinations between the images and logical operators have also being applied to capture precision in the agent's answer.

1.4 Shape detection

Lastly, if none of the above attempts find the agent an answer, the agent starts to detect shapes to understand relationship of shapes in each rows. The agent creates a frame for each image which tells the agent how many squares, triangle, circle, stars or number of shapes exist in the image.

Then, the agent checks for repeating patterns of shapes and finds if there is a repeating pattern then what shape should the answer be to continue this pattern. For example, if a given problem has the first row as [triangle, square, circle], the second row as [square, circle, triangle] and the third row as [circle, triangle]. The agent is able to identify that the third image in the third row should be square. The agent will then attempt to find if the number of shapes in each image in a

row also has a pattern and based on this information, the agent is able to find an answer.

1.5 Finding the answer and voting

Based on the explanations given above, the agent first understands relationships between images in a row or diagonal approach using DPR/pixel density/bit-wise operation/shape detection. It stores this information in a knowledge frame and uses that information to find a possible candidate for an answer. Once a candidate is found, it tests it to ensure if the answer conforms to the given pattern from the knowledge frame. There are various methods to perform the test and it depends on the relationship or pattern the agent is checking for. For example if the agent is checking for an exact match then the agent uses the `cv2.matchShape` function or if it is simply checking for the number of shapes then it will attempt to match the number of shapes in the answer to the knowledge frame.

In each of the above methods of finding an answer, if the agent finds more than one answer it stores it in an array. The agent will clean up the array right after each method is performed by removing duplicates. It would then attempt to find the best answer based off the operation that caused the array to fill up more than one potential answer. For example, the agent could try to select an answer which has a closest match to A and E images in the diagonal relationships, or the agent could try to make sure the possible answer is not similar to any of the images from A to H. The voting system in my agent is not perfect, but is able to help with some cases where the agent is unable to best identify one answer.

2 PERFORMANCE AND EFFICIENCY

When running locally, the agent is successful in identifying the correct answer for 10 out of 12 for both Basic Problem set D and set E. When running it against Gradescope, the agent is successful in passing 10 out of 12 for both Basic Problem set D and set E , and 3 out of 12 for Test Problem set D and 8 out of 12 for Test Problem set E resulting in a score of 100/100 for Milestone 4.

2.1 Agent performance

Agent performance can be defined as how reliable the agent is in using the same methods described in section 1 to find the correct answers. For example, if the agent can perform well on 3 known problems it was programmed against, can

it then use the same methods to come to the correct answer with unknown test cases?

In my case, the agent performed very well for problem set E since it scored a 10/12 on the known Basic problem set locally and on Gradescope as well as 8/12 on Test problem set E which the agent has not seen before.

However, the agent performed poorly for problem set D. It was able to perform well on the known Basic problem set by scoring the same locally and on Gradescope. But for the unknown Test problem set D the agent performed poorly by scoring only 3/12. Therefore, I would say my agent has a decent performance. I hope to increase performance for the final project.

2.2 Performance and struggles

The agent has performed really well in cases where the shapes have similar dark pixels, consistent increase or decrease in pixel change or when there is a very good match, or bit-wise operations performed on the images. A good reason why the agent scored high in problem set E is because of good performance using pixels and bit-wise operations. Also, the agent is good at identifying the number of shapes and type of shape which helps its performance when there is is a pattern of shapes in each rows such as problem D-12.

The agent seems to struggle when there can be two methods applied to solve a problem. For example, in problem E-01, the agent can find the correct answer using bit-wise AND operator, but the agent attempts to solve the answer using pixel calculations in each row because the function to check row calculations comes before bit-wise operation function.

Furthermore, the agent struggles in cases such as D-10 where the shapes cannot be identified by separating them. They are unusual shapes and the agent is unable to see a pattern using DPR or pixels in each row or diagonally either. To be honest, I am unsure how to approach this problem as well.

Lastly, the agent also struggles in cases such as E-09 where there are two shapes in each image and one is colored black and the other is not. The pattern here is not something the agent is currently programmed to understand and this is something I aim to fix in my final project submission.

2.3 Agent efficiency

Efficiency can be described as how quickly the agent can solve a problem. The logic and methods used can be major factors in how efficient an agent is.

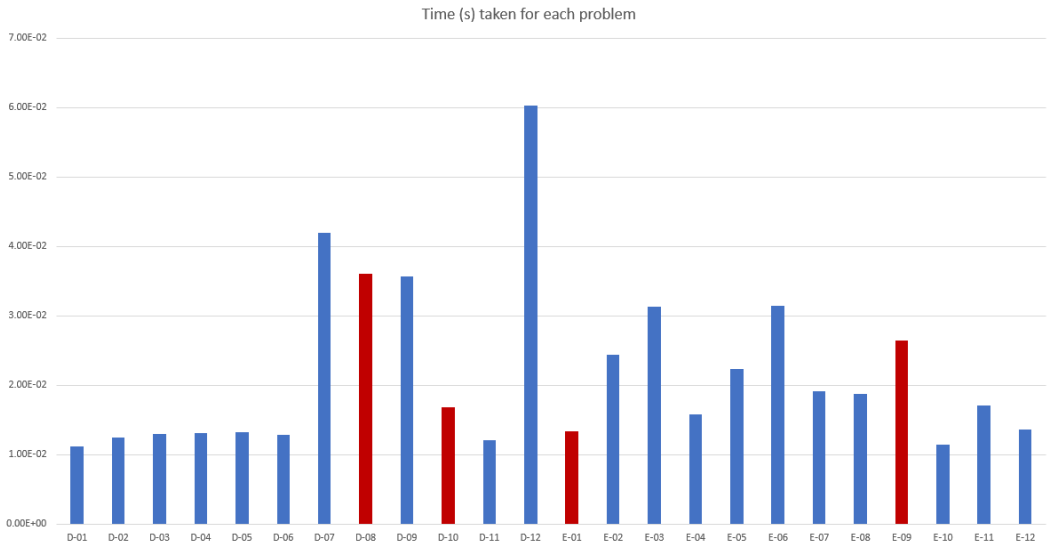

Figure 1—The individual run-time statistics for Set D and Set E. Blue bars indicate agent was successful and red bars indicate agent had failed

The agent performs the entire set D in a total of 0.27894 seconds and the entire set E in a total of 0.25070 seconds. This is something I expected because set D had problems where the agent had to determine shapes and contours compared to set E where it was mostly comparisons using generated frame knowledge. See figure 1 for the graph which shows individual run time statistics of each problem set.

The agent performs fairly well overall however there are cases as it can be seen in figure 1 where the agent takes longer than expected. To me, an agent performing under 0.02 seconds would be considered quite efficient. Therefore, it can be deduced from figure 1 that the agent is mostly efficient except some corner cases.

The problems that took under 0.02 seconds are the cases where its a simple case of finding an exact match in the row or diagonal relationships and a bit-wise operation, or adding or subtracting pixels to find the answer.

The problems that took longer are cases where the agent probably had more than

one answer and had to utilize the voting system methods to finalize an answer. In the case of D12 which took the longest time to execute, it is because the agent had to identify shapes, number of shapes and the patterns that exist and then find an answer.

Finally, the red bars in figure 1 that took longer than 0.02 seconds is because the agent exhausted all attempts and all methods and was unable to find an answer or found an incorrect answer. The red bars that took less than 0.02 seconds is probably because the agent was able to find an answer using the right approach, but the answer was incorrect due to inaccuracies in measurements or contour detection.

3 PERFORMANCE IMPROVEMENTS

To improve performance for the final submission, I have to take the 3 by 3 code and ensure the simplified logic is applied for 2 by 2 to tackle the Problem Set B. I think now that my agent is getting better at determining patterns it should be able to handle the Set B.

Next, I would want to try and attempt to add in any corner cases for each bit-wise logical operator and the combinations that can be present (for example A XOR B = C, A XOR C = B, B XOR C = A). This could potentially improve the test cases as well.

I also plan to improve and generalize the agent in shape detection so that it can solve cases such as E-09 which have shapes that are colored and not colored. I want to make sure the agent understands when it sees something similar.

Then, I aim to run all the test at once and see if the agent is getting any problem incorrect when I know it should be able to solve the problem. This would help me determine if there are multiple methods to solve a problem and one is simply more accurate than the other - this also helps me clean up my agent's code and increase performance and efficiency.

Lastly, to make slight improvements overall, I want to continue testing my agent against all the Challenge Problems to see if I can improve my agent's performance and efficiency. Currently, the Test problems for problem set B is very poor (5/12) and I think my agent is now equipped with more tools to increase performance for set B.

4 FEEDBACK REQUEST

I would like to get feedback in the areas where I identified my agent is struggling in section 2.2 Performance and struggles. I would like to know what methods is best to approach problems such as D-08, D-10 and E-09. There are multiple shapes in these problems where the difference is colored/uncolored shapes, merged shapes and shapes within each other. How does one best approach the problems without writing an over-complicated code?