RPM Milestone Journal B

Michael Lukacsko
mlukacsko3@gatech.edu

1 AGENT FUNCTIONALITY

Before implementing my agent, I first categorized the Basic Problems B into distinct groups based on the types of visual relationships observed in the figures. These categories helped guide the development of my agent's problem-solving strategy.

2D grid relationships where A, B, and C are the same: In these cases, my agent looks for identical or nearly identical figures in positions A, B, and C using a combination of direct pixel comparison and contour analysis. My agent compares the images by calculating the pixel-wise differences between the images and detects similarities using a threshold of 0.95. Where the threshold is > .95, the images are considered highly similar. As such, my agent selects the answer option that most closely matches A, B, and C by applying the same pixel comparison process to each of the answer options.

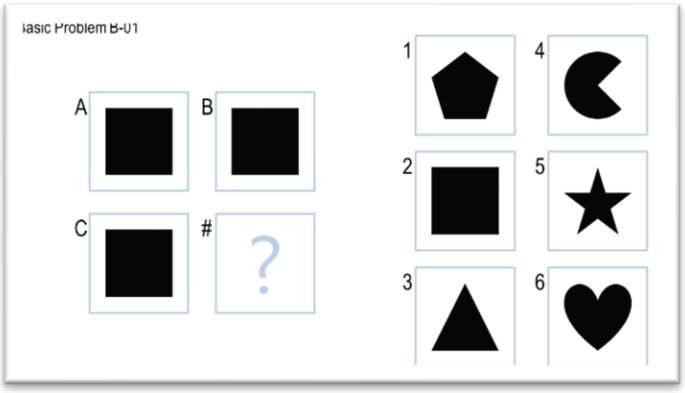

Figure 1— Example of 2D grid relationships where A, B, and C are the same

1. **Mirror image problems**: These problems involve situations where either A and B or A and C are mirrors of each other. A and B and C are mirror images in some cases. To determine these transformations, my agent checks for mirror relationships by flipping images horizontally and vertically. If a mirror relationship is detected, my agent searches for a

corresponding mirror relationship between B and the correct answer by mirroring B to predict the correct answer for D.

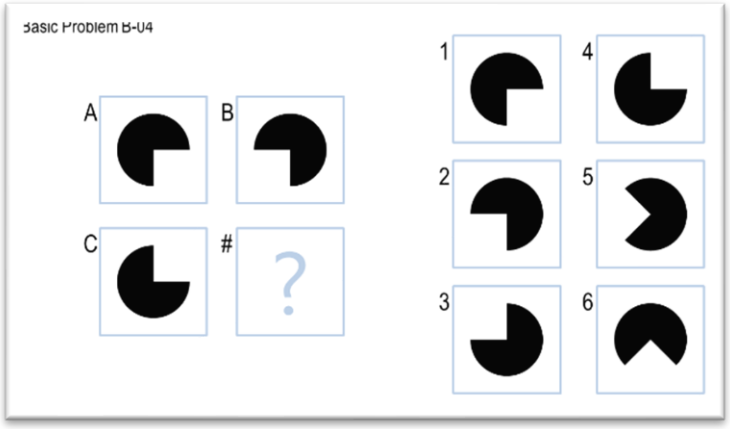

Figure 2 — Example of mirror image problems.

2. **Shading or shape removal problems**: In some problems, it is observed that a shape is removed, or the shading of a figure may change between positions. My agent detects these changes by comparing image contours using OpenCV's contour detection, analyzing the number of distinct shapes as well as detecting differences in shading. By converting the images to binary arrays and identifying the contours, my agent can count the number of shapes and assess whether an internal shape has been removed or if shading has changed. If one image has fewer shapes or different shading, my agent interprets this as a shape being removed, or modified, and selects the corresponding answer option based on these changes.

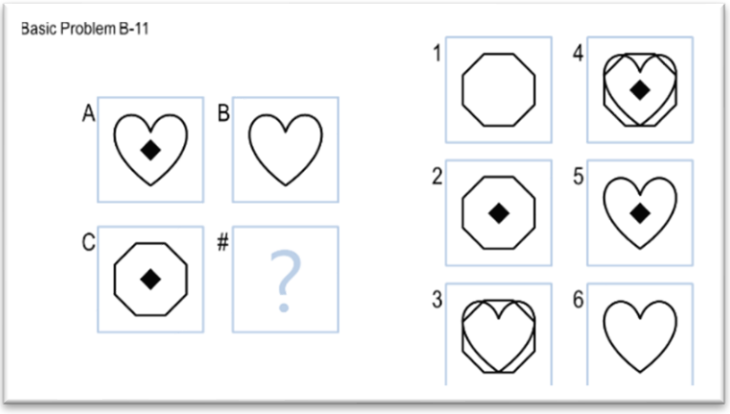

Figure 3 — Example of a shape removal problem.

2 AGENT PERFORMACE EVALUATION

My agent is currently performing well on the Basic Set B problems, correctly solving 9 out of 12 problems in this set. It successfully identifies patterns and transformations in problems B-01, B-02, B-03, B-04, B-07, B-08, B-10, B-11, and B-12. However, my agent is unable to identify the correct answer for problems B-05, B-06, and B-09. In these cases, my agent struggles to recognize the specific transformations or relationships between the figures, even though some of these problems seem to closely align with the patterns identified in the previous section. With an overall accuracy rate of 75%, my agent demonstrates solid performance but has room for improvement.

3 AGENT STRENGTH AND WEAKNESSES

My agent performs well on problems that involve direct pixel comparison, simple transformations, and cases where figures in positions A, B, and C are either identical or follow a clear transformation. In these problems, my agent effectively detects patterns, such as mirrored or identical figures, and correctly selects the answer based on pixel similarity and transformations like flips or rotations.

However, my agent struggles on problems like B-05, B-06, and B-09, where the relationships between the figures are more complex. Specific to B-09, my agent has difficulty detecting the more abstract transformations between the octagonal and square shapes, particularly with shading and shape transitions. B-05 presents a challenge where the transition patterns are less clear compared to the other similar problems, leading my agent to miss the correct answer.

4 AGENT EFFICIENCY

Based on the solution time analysis in figure 4 below, my agent performs efficiently overall but takes noticeably longer to solve **B-01** through **B-03**, despite these being some of the more straightforward problems conceptually. **B-02** is the slowest, taking approximately **0.08 seconds**, with **B-01** and **B-03** taking around **0.06** and **0.04 seconds**, respectively. Interestingly, **B-01** and **B-02** involve simple patterns: in both cases, A, B, and C are identical to the human eye, so the answer should be a direct match with them. For **B-03**, A and C are the same while A and B are mirrors, which logically suggests that the answer should be a mirror of B.

Despite these clear patterns, my agent spends more time processing these problems than expected. My best guess is that the reliance on NumPy for pixel-by-pixel array comparisons contributes to the increased runtime, as subtle differences between images could lead to excessive calculations. This overprocessing, especially in cases where direct logic could suffice, is a likely cause of the slower performance on these simpler problems.

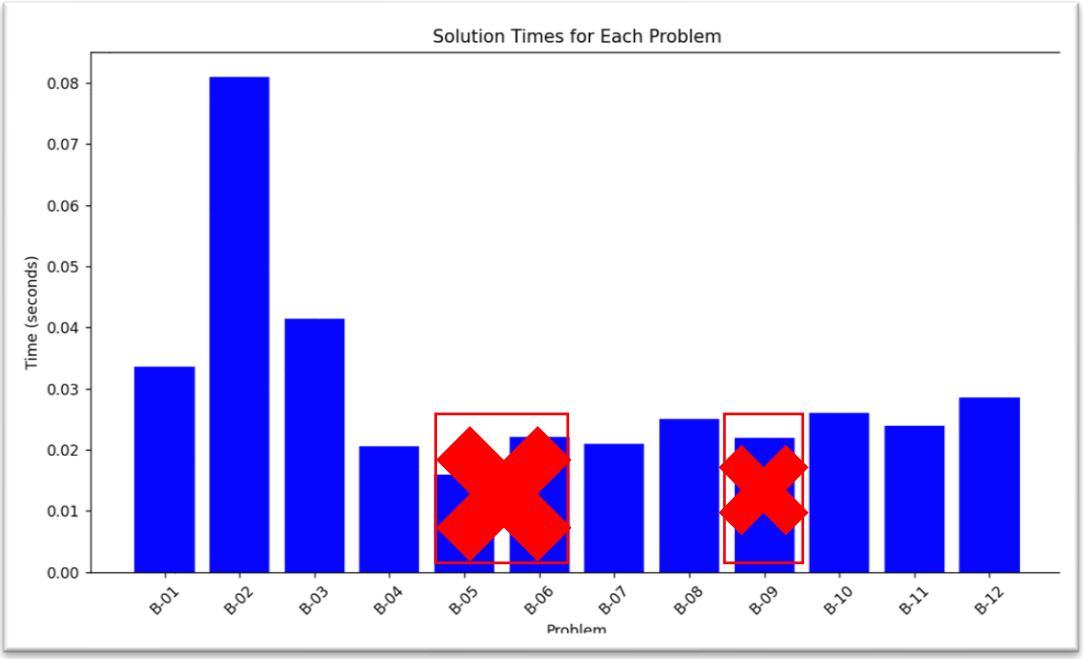

Figure 4 — Agent efficiency. Seconds per problem.

5 AGENT IMPROVEMENTS

To improve my agent's performance before the final project submission, I plan to focus on two primary enhancements: solving the problems that are currently incorrect (priority 1) and optimizing the efficiency of my agent's approach to finding correct answers.

First and foremost, solving the incorrect problems is the top priority. For each problem in this problem set, I need to analyze where my agent is making errors in logic or image comparison. For example, in problems like B-05, B-06, and B-09, where my agent struggles to identify the relationship between shapes, I will refine my agent's image-processing logic to better handle these

transformations. This will involve adding the necessary code that will addresses these issues. I suspect this will also include improving contour detection for internal shape changes and/or enhancing the mirror-image detection logic. I will also introduce more sophisticated shape and shading analysis using OpenCV's image-processing tools to recognize these visual changes more accurately. With these specific changes, I hope to enable my agent to identify the transformation in B-05, B-06, and B-09.

Lastly, I will improve the overall efficiency of my agent, particularly in the simpler problems (B-01 to B-03), where it currently takes longer than expected. To do this, I will add an initial logic layer that quickly identifies problems with simple patterns before applying more time-consuming transformations. This selective approach will reduce unnecessary pixel comparisons and transformations, streamlining my agent's decision-making process. I will also need to either optimize the use of NumPy for pixel-wise comparisons by ensuring that my agent only performs these computations when simpler checks have failed or find an alternative solution to further reduce computational overhead.

By addressing the incorrect problems as a priority and reducing unnecessary computations for simpler problems, my agent's accuracy and efficiency should improve significantly before the final submission.

6 AGENT DESIGN FOR 3X3 PROBLEMS

To generalize my agent's design to cover 3x3 problems in addition to the 2x2 problems, I will have to expand my agent's logic to handle more complex relationships and patterns that arise in 3x3 matrices. Here's the plan for doing so:

1. **Extending the 2D Grid Logic to 3x3:** The 2x2 problems already rely on detecting relationships between rows and columns using a 2D grid of images. For 3x3 problems, I will extend this logic to a 3x3 grid, where my agent checks for consistency in rows, columns, and diagonals. This will involve comparing images in each row (A → B → C), each column (A → D → G), and diagonals (A → E → I) to identify patterns. I will also use pixel comparison and transformation techniques, similar to those in 2x2

problems, to detect mirror images, rotations, or shape changes across the grid.

2. **Pattern Recognition Across Rows and Columns:** In a 3x3 matrix, patterns will span multiple figures across rows or columns. To accommodate this, my agent will check for transformations that occur consistently across a row or column. For example, if a shape rotates or scales as it moves from left to right across the matrix, my agent will detect and apply these transformations to find the missing figure in the third row or column. This will be done using the same transformations (flip, rotate, mirror) that are used in the 2x2 cases, but applied across larger sequences of figures.
3. **Contour and Shape Analysis for 3x3 Problems**: Some 3x3 problems involve complex shape transformations (e.g., removal or addition of shapes). For these, I will extend the current contour detection methods used for 2x2 problems to 3x3 matrices. My agent will analyze the contours of shapes in a more sophisticated way, such as detecting if shapes are progressively removed, resized, or shifted across rows and/or columns. This enhancement will allow my agent to handle problems where the changes in figures are more subtle or gradual in the 3x3 problems.
4. **Fallback Approaches:** For particularly challenging 3x3 problems, where standard transformations or row/column logic do not apply, I will implement fallback approaches, such as comparing the entire grid at once using pixel-based analysis. Additionally, more advanced shape and contour analysis using OpenCV can and should help detect subtle differences in shading, shape alterations, or symmetry that might not be easily captured by simple transformations. This enhancement will allow my agent to still propose a best guess even when no obvious pattern emerges.

By scaling up the current logic for detecting visual relationships in 2x2 matrices and expanding it to handle the additional complexity of 3x3 matrices, my agent will be able to handle a broader range of problem types and transformations. This generalized approach will make my agent capable of solving more difficult Raven's Progressive Matrices problems in future sets.

7 PEER FEEDBACK

I would appreciate feedback on improving shape recognition and contour detection, as this is one of the biggest challenges I'm currently facing. I use

OpenCV for contour detection to identify shape changes, such as additions, removals, or scaling of internal shapes. However, this method has not always been reliable, especially when dealing with more complex shapes or subtle changes between figures. I'm interested in hearing if any classmates have experimented with more advanced shape recognition techniques or different libraries, specifically Pillow, to improve contour analysis accuracy. Beyond that, while efficiency is important, it's secondary for me if my agent solves the problems correctly. Nevertheless, if classmates have suggestions on how to optimize my agent's efficiency without sacrificing accuracy, I'd be open to exploring that feedback as well.