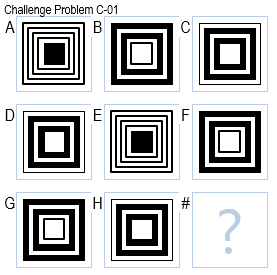
CS 7637: Project 2 Journal

Zachary Todd  
ztodd3@gatech.edu

# 1 Introduction

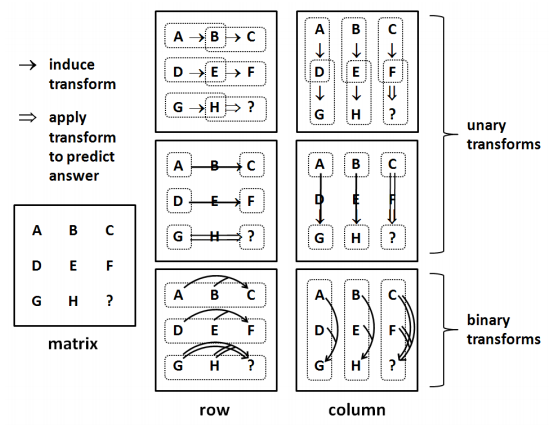
This paper will outline the second iteration of developing a KBAI agent capable of solving Raven’s Progressive Matrices (RPMs). This iteration will focus only on solving 3x3 matrices as the agent is already quite successful at solving 2x2 matrices.



***Figure 1—***For clarity, I will refer to different Images in a 3x3 Raven’s Problem as A, B, C, D, E, F, G, H, and # as shown above for the rest of this paper.

## 1.1 Initial approach

I plan on identifying unary and binary transforms in rows and columns (Kunda). These transforms can then be applied to image H or F as seen in Figure 2 to derive a guess. This guess will then be compared to the possible answers.



***Figure 2—***By detecting patterns in the given problem, the agent can make a guess at the correct answer.

# 2 First strike (06/19) #1

In this submission, I utilized a blob detection algorithm called Connected-Component Labeling. In essence, this algorithm assigns labels to “blob” of black pixels that are separated by white pixels. In doing this, the agent is able to count the number of blobs in an image.

## 2.1 Performance

The agent did as expected for this first submission. While it was only able to answer a few questions correctly, it did not answer any questions incorrectly. This indicates that the algorithm for removing bad answers is not making mistakes.

## 2.2 Cognitive connection

The agent is only using one technique to solve these problems right now, but it is a technique most humans would use. There are many of these problems where certain answers are most definitely wrong. A human would be able to recognize these bogus answers, eliminate them from consideration, and make their decision-making process easier. This will also help tremendously with computation time.

# 3 Diagonal improvements (06/22) #2

In this iteration, the agent now checks diagonal blocks to see if they are the same. If they are, it will immediately check the available answers to see if one matches. This solves Challenge Problem C-01 very well. This is indeed a special case and is not an efficient way to build the agent. However, by implementing a solution to this case, I am laying the framework for how the agent will solve the RPM when rows or columns are the same. The algorithm will follow the same process implemented here. I expect the next iteration will show large improvements in the agent’s performance.

Additionally, I added functionality to the agent’s ability to determine if two images are the same. Before, the agent would check pixel by pixel, and determine a percentage of different pixels. Now, the agent does this, but it also uses two other metrics to determine similarity. The first is the total number of black pixels in each image normalized over the mean of the length and width of the image. The next metric is the center of mass of all pixels in the image. The agent takes the root mean squared of the different pixel count, the total number of pixels, and the center of mass to find the similarity.

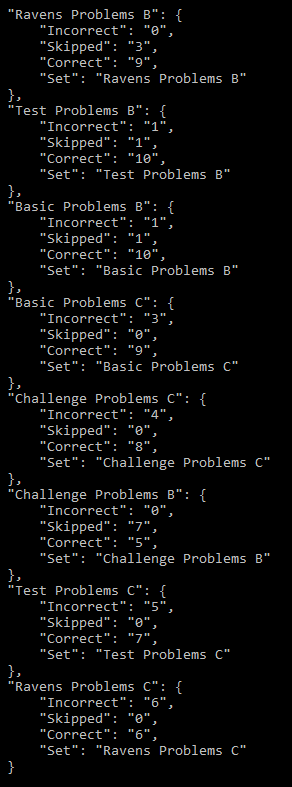
## 3.1 Performance

## 3.2 Cognitive connection

# 4 All about the pixels (06/24) #3

Added a check for amount of pixels in an image. Also changed the criteria for selecting an answer. The agent now selects an answer at the end that it is most confident in. This is exactly how a human would act. There is no penalty for guessing the wrong answer.

## 4.1 Performance



## 4.2 Cognitive connection

Kunda, Maithilee, et al. “A Computational Model for Solving Problems from the Raven’s Progressive Matrices Intelligence Test Using Iconic Visual Representations.” *Cognitive Systems Research*, vol. 22-23, 2013, pp. 47–66., doi:10.1016/j.cogsys.2012.08.001.