Kelly Cheng

k.cheng@gatech.edu

In Project 3, I experimented with many different approaches, achieving different degrees of success with each approach. I will briefly review these to explain how I concluded on my Project 4 approach. I began Project 3 by attempting to use OpenCV and image processing to relate the problems back to the verbal definitions given in Projects 1 and 2. I felt, at the time, that though there have been issues discussed as far as specificity of attributes like angles, that the logic my agent had for Projects 1 and 2 was much closer to the reasoning people use to figure out a solution. The verbal reasoning was a very methodical approach, considering all aspects of the objects, coming up with a proposed solution, and then finding the closest match to that proposed solution. Unfortunately, I immediately ran into one of the unique problems of visual reasoning - my lack of image processing experience! I was unable to implement OpenCV effectively. I errored out the majority of the problems, and was only able to achieve 3/40 of the basic 2x1 and 2x2 problems.

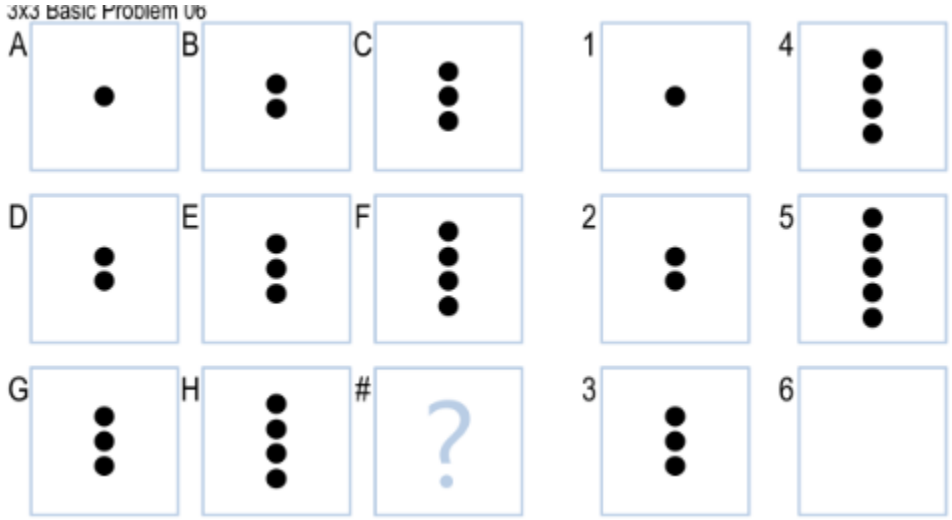
I moved on from that and attempted a pixel by pixel interpretation, attempting to use visual heuristics to find the solution. At the time, I was defeated, thinking that a pixel by pixel read-through was nowhere close to human cognition. Humans tend to look at pictures as a whole, not piece by piece, which was actually one of the reasons I believed the verbal reasoning wasn't well related to human cognition (too specific, and it needed to consider every attribute). The way it worked was by reading the RGB values of each pixel in A, compared it to the RGB values of the corresponding values in B, then found the same comparison for C and the solution figure. It chose its final solution by choosing the lowest percent difference between the A-B comparison and the C-solution comparison. Though ungainly, it was able to achieve 10/40 (1 of the 2x1 and 9 of the 2x2). Surprisingly, it did much better on the 2x2s than the 2x1s, possibly because of differences in the images for each group. Most likely, the 2x1s had many more non-exclusively black and white pixels (the surrounding pixels of an image are most likely a gray color), and the agent saw those as mismatches. To handle this, my agent used a very similar algorithm for the 2x1s, the only difference being that it used only the completely black pixels to determine the percent change. In the end, the combination of the two gave 16/40 correct solutions. The agent performed best on problems where images A and C were very similar, such as basic 2x2 problem 7. This is expected, as the changes made to similar images should be similar.

I spend a lot of time during Project 3 attempting to understand and implement the affine method. I found some success here. I was able to use the eight base unary affine transformations in Dr. Kunda's 2013 paper: identity, rotate90, rotate180, rotate270, identityflip, rotate90flip, rotate180flip, and rotate270flip to at least narrow down the answers. Those basic transformations only gave 3/40 correct, so I ultimately decided not to turn that version in, though I did resolve to go back to it in Project 4. In Project 4, I attempted to address the five base binary set transformations from Dr. Kunda's paper. Unfortunately, I ran into implementation issues and did not do much better.

After much thought, I realized that I was complicating the problem too much. Many times, humans just look at a problem as a whole and can fairly quickly pick out a solution based on the whole picture, not

analyzing each object attribute by attribute or pixel by pixel. I also realized that the 3x3 problems could be thought of in a different way than the 2x1 or the 2x2 images – they had many more relationships, which, while seemingly making the problems more difficult, actually made them easier. For every problem, I considered the rows, columns, and diagonals, then chose a solution that matched most if not all relationships. So unlike the 2x1 or 2x2 problems, the proposed solution's position could actually be weakened or solidified based on how well it fit into each of the relationships.

Take 3x3 Basic Problem 6, which the agent got correct:



Looking at the row, I see one dot being added going across the first row. The second row further comfirms this. I propose that the solution would add another dot to H. I can continue to strengthen my proposition by checking the columns. There is one dot added to each image in each row. So I confirm that 5 is the correct answer. This "check" couldn't be used in the 2x1 or 2x2 problems, since there was only one direction in the 2x1s, and the 2x2s largely didn't have the same transformation done in the row and column (we couldn't check diagonal, since there was only one image in the diagonal).

The agent similarly makes use of the multiple relationships like a human would. I chose 4 very broad rules that covered the majority of the problems: matching objects, adding objects, deleting objects, and random draw. The agent goes through each problem, checking if the problem fits into any of the categories, in that order. If the problem fits into one of the categories, the agent uses the black-pixel count method from Project 3 and a proposed solution of same, more, or less pixels to choose the correct solution.

Unfortunately, because the agent only looks at the images as a whole and doesn't do object and attribute interpretation, it doesn't react well to problems where the object is the same and the transformation can't be determined directly by a pixel count, such as 3x3 basic problem 19:

This problem would fit under the rule “matching objects” because the pixel count is the same for every image. Unfortunately, it’s also the same for every solution, so the agent can’t tell the correct solution. This points to one of the unique challenges of visual reasoning compared to verbal reasoning: while I felt verbal reasoning was too specific to be interpreted as similar to human cognition (many humans would interpret angle changes as “fuzzy numbers,” where the agent has to calculate the exact angle at which an object rotates), visual reasoning is almost too broad. The agent can’t “see” the difference between each image because it can’t read the placement of the Pac-Man mouth, just its overall size (through pixel count).

One thing that I would like to improve is the agent’s ability to “see” the objects. With more time, resources, and processing power, the agent would ideally find the number of objects being added or deleted (by looking at the added ratio) and choose a solution closest to the ratio instead of choosing any of the solutions based on added or deleted pixels. It could also be improved by implementing a security rating based on how many relationships (number of row relationships, number of column relationships, and number of diagonal relationships) match the proposed relationship. It could then choose the solution that matches to the relationships with the highest security rating.

In conclusion, though I started off Project 4 still under the impression that my Project 3 agent didn’t “think like a person,” I was able to actually change my mind by looking at the 3x3 problems. The idea that more relationships actually simplify the problem was a revelation to me – I saw that my agent was able to “see” the image as a whole, and it wasn’t really all pixel by pixel processing.