CS7637 Project 3 Reflection

Chenxi Gao
cgao71@gatech.edu

INTRODUCTION:

Before I started project 3, I planned to use a similar approach in Project 2 and Project 1, assuming the problems can be solved mostly by simple transformations (e.g., unchanged, rotation, and reflection), addition/subtraction, counting, etc. However, after I closely examine multiple problems in Problem Set D and E, I quickly realized that the problems are much more complicated than the problems in Problem Set C because the majority of the questions do not have a clear one-shot solution. Some in the problem set take me a few minutes to understand and apply the best relationship to find a choice among all candidates. Even I solved two to three questions wrong. Additionally, Project 3 requires us to use a complete visual approach while its difficulty grows dramatically. For example, the semantic information in Problem D-04 and D-05 can be easily addressed by using a verbal approach such as a rectangle is inside the circle. However, it would be harder to identify such relationships in a visual approach because the agent cannot extinguish shapes, given that the images are represented in pixels.

The following are the requirements and expectations that our agent needs to accomplish in this project after I carefully observe most problems:

1. Our agent needs to solve the problems by a complete visual approach.
2. Our agent needs to find alternative ways to capture semantic information between images.
3. Sometimes, one single row or column is not sufficient to capture the relationships between images. Additional rows or columns and/or the diagonal entries may help together to determine the relationship.

Because Project 3 requires an entirely visual approach to solve the 3x3 RPMs, and I have already used a purely visual approach in both Project 1 and Project 2, I decided to further improve and build my agent based on my existing code. I think my agent is robust in finding simple relationships and comparing whether or not the two images are identical. Moreover, I decided to incrementally

improve by adding one or two features and/or tuning existing parameters to make my agent have better performance in solving the problems. In the journals and the conclusion, I will further explain the advantages, disadvantages, and limitations of using a purely visual approach comparing to a verbal approach and other potential methods of solving 3x3 RPMs.

JOURNAL 1:

This is my first submission at 9:53 p.m., 11/29. I tuned some parameters in Project 2 to possibly obtain a better performance in Project 3. Below is a summary of what is working and what is not working well. I will identify some strengths and weaknesses my current agent has. In this submission, I primarily focused on solving Problem Set D.

As I have mentioned before, my agent in Project 2 did well in identifying simple transformations, finding differences between images, and counting the number of same shapes in images. In this submission, my agent only did well in identifying same figure across rows/columns in Problem D-01, but it failed most of the other test cases despite the fact that my agent guesses a couple problems correctly.

I quickly realize that my agent from Project 2 was unable to capture certain semantic relationships in this project. For example, in Problem D-02, one of the evident relationships is the three figures (circle, rhombus, and triangle) appears only once in each row/column. Therefore, we need to choose the figure that does not appear in the third row/column but does appear in the other rows/columns. Another example is in Problem D-04, finding the differences between two images (XOR operation) cannot fully identify the relationship of the outer shape is changed, but the inner shape is unchanged. I remember in Project 2, my original implementation of XOR worked because some shapes exist in the first image disappears in the second image while the other components remain the same.

Therefore, I conclude that the Project 3 is much harder than the previous ones because:

1. The relationships have lighter weights as described in Lecture 3 so that it is generally harder to capture such relationship compare to simple transformations.

2. There are multiple semantic relationships intervene together in a single problem. Therefore, it is harder to dissect relationships because we need to consider many more cases than the problems in Project 1 and 2.

Below is a result of my first submission:

```
Basic passed: 9, failed: 15
Test passed: 6, failed: 18
Challenge passed: 5, failed: 19
Ravens passed: 7, failed: 17
Runtime: 10.070834398269653 seconds
```

Surprisingly, my agent answered 5 correctly in Basic, but it only passed 4 in Test problems in Problem Set D. I think my agent might be able to guess about two questions correctly based on my previous implementation. There are a lot more to implement to further improve my agent.

Comparing my agent's performance to what I would generally solve 3x3 RPMs of simple transformations, I think my conclusion is similar to that of Project 2:

1. It is similar to my conclusion in Project 2 that the way of finding and applying transformation is close to humans.
2. The method of comparing figures in 3x3 RPMs is similar to what I would do. For example, if I can find a relation in the first row, I would skip the second row and directly apply the same relation in the third row to get the answer. If it failed, I would try to find other relations.

JOURNAL 2:

This is my second submission at 5:05 a.m., 12/01. Below is a summary of how I implemented a method called *crop_and_compare* to dissect a portion of the image and compare them separately. In addition, I added more code in the simple transformation to compare the figures in the diagonal.

I noticed that some of the problems in this problem involve either a change of shape in the center or a change of shape around the center. For example, in Basic Problem D-04, the shape in the center stays the same in the same row, and the outer frames are the same in the same column. Therefore, I implemented the following method:

1. Crop the image in the center of each figure and store it in a *crop_list* across a row/column (use *image.crop*)
2. Make another copy of the figure, pad the center using floodfill and store it in an *outer_list*
3. if the outer frame or center of the images in the lists are the same:
    a. Find all the figures in the answer set so the outer/center match and store it in a list
    b. If outer frames are the same, compare the center to the *crop_list*. The solution is the one that appears in *crop_list* but does not appear in the list we just created.
    c. If the centers are the same compare the outer frames to the *outer_list*. The solution is the one that appears in *outer_list* but does not appear in the list we just created.

In addition to the above algorithm I implemented, I notice the figures in the diagonal entry are the same in some problems. Therefore, I compare figure A and E and find the solution to be identical to A if A and E are the same.

Below is the result of my submission:

```
Basic passed: 10, failed: 14
Test passed: 8, failed: 16
Challenge passed: 8, failed: 16
Ravens passed: 6, failed: 18
Runtime: 9.047215223312378 seconds
```

At this moment, my agent is able to solve 8 out of 12 Basic Problems correctly. I realize that there are limitations to this method:

1. One shape has to be located in the center, and the other one must be around the center. Otherwise, this method will not work.
2. It is uncapable to capture additional relationships if both outter frame and the center change at the same time.

Comparing my agent's performance to humans at this point, I think it acts pretty similar to the way I solve this problem. If I cannot instantly get an answer, I would first check what shapes in the images are the same, and what are different. Then I would try to reason out the differences between these shapes and apply the same relationship to find the answer.

JOURNAL 3:

This is my third submission at 5:59 a.m., 12/01. Below is a summary of how I fixed a problem in the above *crop_and_compare* method and greatly improved my agent's performance to solve this kind of problems.

The above algorithm, I floodfill the center of the image to white when I tried to get the outer frame. However, this would leave a black frame on the edges of the shape in the center that would impare my agent's ability to compare with similar images. I did some research online, and I replaced the floodfill method by a *Image.paste* method. I would create a completely white image and paste it to the center.

In addition to the above problem, I notice this caused my agent to answer some of the problems wrong. In particular, this method does not work if the outer frame is empty. For example, in Basic Problem D-08, the outer frame is always empty because only one shape exists in each image. Therefore, I added another constraint so that the outer frame must not be completely white.

Below is the summary of the result:

```
Basic passed: 13, failed: 11
Test passed: 7, failed: 17
Challenge passed: 6, failed: 18
Ravens passed: 6, failed: 18
Runtime: 8.422865390777588 seconds
```

At this moment, my agent can answer 9 out of 12 problems correctly in Basic Problem Set D.

Again, my agent's performance comparing to humans at this point would be similar to what I wrote in journal 3.

JOURNAL 4:

This is my fourth submission at 6:58 p.m., 12/01. Below is a summary of how I implemented a XOR3 method to find the differences across a single row/column.

I visited the visual reasoning thread, trying to find a more sophistciated way to compare a row/column of images at once. Aidan said we may try to use a XOR3 method, where "XOR is defined by a dark pixel existing in the output of 3XOR

if and only if that pixel exists in exactly one of A, B, or C". Then, we can apply the same 3XOR in the third row/column, trying every single figures in the answer set, to find the best match comparing to the result of the previous two rows/columns. After I closely look at some problems in the dataset, I believe this approach makes sense, and it will work for many problems such as Basic Problem 02 and 03 because the composite of the rows would be the same. Below is how I implemented the XOR approach:

1. XOR3 is defined as finding the differences between image1 and image2, the differences between image2 and image3. The result is the difference of the two differences generated above.
2. I record the XOR3 in the first two rows/columns. If the result mathes, I fit each answer in the third row/column and try to find the best match to the composite figure.

Below is the summary of the result:

```
Basic passed: 13, failed: 11
Test passed: 7, failed: 17
Challenge passed: 3, failed: 21
Ravens passed: 10, failed: 14
Runtime: 6.2764892578125 seconds
```

I notice that the performance in Basic and Test Set stays the same, but the overall runtime is improved by 2 seconds. Later I noticed a mistake in the XOR operation, and I will discuss it in the following journals.

In this submission, I think my agent performs differently from what humans would normally do. Because we distinguish objects in RPMs as shape, but our agent does not have a concept of shape in a completely visual approach. Computers view images as pixels, and our agent finds relationships between pixels. Humans would not perform comparisons between images using pixel operations.

JOURNAL 5:

This is my fifth submission at 3:22 a.m., 12/02. Below is a summary of how I further improved the *crop_and_compare* method, a modification to XOR3, and some attempts to count the number of different shapes in Basic Problem D-12.

I noticed that some problems in the diagonal can be separated to two figures where one of them is the same across the diagonal. This can also be solved by the *crop_and_compare* method even though all shapes in rows/colums are changing, such as Basic Problem D-07. Similar to the approach I described before, I separated out the outer frame and compare if they are similar. Then find the figures in the answer set, which have the same outer frame. The answer is the one that have different center.

Additionally, I made changes to XOR3 to be finding the differences of image1 and image2, and the result is the difference of the difference betwwen 1 and 2 and 3. It is similar to an XOR gate with 3 inputs. After I tested it, I noticed that this is not entirely correct. Because in the truth table of XOR3, the last row when all of the 3 inputs are 1, the result is 1. However, Aidan stated that "one black pixel exists in exactly one of A, B, or C". Therefore, my agent failed to solve problems like Basic Problems D-06 because all rows have the same outer frame, and the current XOR3 approach cannot separate the outer frame out.

The following is the summary of this submission:

```
Basic passed: 12, failed: 12
Test passed: 7, failed: 17
Challenge passed: 11, failed: 13
Ravens passed: 8, failed: 16
Runtime: 7.018309831619263 seconds
```

In this submission, the accuracy dropped in Basic Set. I found out that it is hard to tune many of the parameters inside my code. Because computers can only compare two images by pixels, and sometimes identical images to us may have a slight shift, so the resulting two images are not perfectly aligned. The computer may view them as different images. However, if I increase certain threshold, my agent may answer some other problems wrong because the wrong answer may "sneak in" because of the increased threshold.

Comparing my current agent to human, I think unlike humans, my agent is solving around the problem, but not directly solving the problem. Because many times our agent cannot have a same approach like human to solve some problems, we, as humans, must try to find alternative ways that are suitable to our agent.

JOURNAL 6:

This is my sixth submission at 4:26 a.m. 12/02. Below is a summary of how I fixed the XOR3 problem and trying to tune some parameters in my code to have a better performance.

To address the previously mentioned problem in XOR3, I first convert the three images to numpy array, where all values are between 0 and 255. I inverted the white and black pixels by subtracting each image arrays from 255 and add them all together. Lastly, my agent returns the array of the sum to have the value less than 255 equals 0. This approach ensures that only one black pixel exists in a certain location in the same row.

Later, I tried to improve my existing counting method by ratio, but I found it to be very difficult. Because the shapes across each row/column are different, I cannot use the normal ratio method because it only works when the shapes are the same. Another method is trying to separate out each shape and count them individually, but this is very difficult because the shapes are not at a specific location in all images. Furthermore, it is hard to identify some shapes visually. For example, a circle would appear to have many edges and corners.

Below is a summary of this submission:

```
Basic passed: 14, failed: 10
Test passed: 9, failed: 15
Challenge passed: 9, failed: 15
Ravens passed: 7, failed: 17
Runtime: 4.254072427749634 seconds
```

At this time, the accuracy increases, but the accuracy of Test Set D is capped at 6. I think there might be an overfitting problem either my parameter fits too well to the Basic Problem Set, or the correct method of solving problems in Test set differs to how my agent approaches to the same problem.

Comparing to my agent to humans, I realize that trying to make the machine to distinguish shapes or intersection of shapes at different location is much harder than humans. This is due to the fact that humans recognize an image at the same time, but machines can only access pixels. They do not have an entire view of the image.

JOURNAL 7:

This is my seventh submission at 8:33 a.m. 12/02. Below is a summary of how start tackling problems in Problems Set E and implement the *addition* and *difference* method.

Surprisingly, the method I implemented in solving Problem Set D is able to solve 5 out of the 12 problems in Set E. Problems such as E-06 and E-07 can be solved by using the XOR approach. However, I realized that most problems can be solved using an addition and difference approach.

For example, in Basic Problem Set E-02 and E-03, we can think of the third row/column is obtained by merging the first two images in the same row/column. Because the black pixel's value is 0 and the white pixel's value is 255, but we are trying to adding black pixels to the white pixels, we have to first invert all the pixels to its opposite value before the addition operation, and invert it back when comparing to the answers in the answer sets.

Then, I implemented the *difference* method (or XOR method) to solve Problem E-05 because we can think of third row/column is obtained by the differences between the first and second image in the same row/column. The process is similar to the *addition* method above. Finally, we need to invert our result to find a match because difference is expressed in white pixels.

The following is the result:

```
Basic passed: 18, failed: 6
Test passed: 9, failed: 15
Challenge passed: 9, failed: 15
Ravens passed: 11, failed: 13
Runtime: 4.9302685260772705 seconds
```

I notice it is much easier to improve Basic Problem Set because we can see the problems, but it is much harder to pass the hidden tests. I will discuss about this later in the conclusion.

Comparing my agent's *addition* and *difference* method to humans, I think my agent performs similar to what I would do. Additionally, computers are capable to perform such operations much faster and more precisely than human.

JOURNAL 8:

This is my last submission at 9:42 p.m. on 12/02. Below is how I improved the *addition* method and my thought about how to implement the rest two problems my agent answered incorrectly in Problem Set E.

I noticed my agent answer incorrectly in E-11 because in this problem the white pixels of the first two figures are considered added together. Therefore, I added more code to add the first two pictures without inverting.

Below is the final result:

```
Basic passed: 19, failed: 5
Test passed: 11, failed: 13
Challenge passed: 9, failed: 15
Ravens passed: 12, failed: 12
Runtime: 4.951265096664429 seconds
```

I looked at two problems my agent answered incorrectly, E-04 and E-12. Even I do not know the correct way of solving E-04, and the images are not aligned. Therefore, I chose not to do this question. I believe E-12 requires us to separate the shapes. However, I did not implement this part.

Again, the comparison between my agent and human is similar to that of the last journal.

CONCLUSION:

I submitted my last submission by tuning some parameters in my code because I got 6/12 in Test.

I think my agent performs well in Basic Problem Set because it is able to answer the majority of them correctly, but it performed poorly in the test set. I understand that the test sets are composed by questions that are very similar to the basic set, but still, the shapes and the relationship in test set may differ to what I observed in the basic set. Another potential reason is the problem of overfitting. Because my code uses many parameters when comparing whether two images are the same. I tried to tune the parametors to fit the Basic Problem Set, but I do not know if the same parametors fit in the Test Problem Set.

I would characterize the overall process of designing my agent to be targeting one type of problem at a time, and I designed improved my agent gradually. First, I focused on *crop_and_compare* for questions that have a shape in the center and another shape outside. Next, I implemented 3XORs to capture the inner relationship between 3 images all together. Finally, I worked on addition and subtraction between images.

Comparing a visual approach to a verbal approach of solving RPMs, I think sometimes a complete visual approach is difficult to capture any simple verbal relationships, especially when the shapes are intervened together. I believe the combination of both visual and verbal approach would make our agent to achieve better accuracy and solve RPMs more like humans because we use both to solve RPMs.

In terms of complexity, I think a visual approach would be faster for computers because computers perform arithmetic very fast, given that the size of input images is relatively small. However, if our agent uses a complete verbal approach, I believe the complexity would grow exponentially if the number of objects increases because we need to consider all possible relationships between shapes.

Comparing my agent's approach to my approach to solve the RPMs, my conclusion is similar to what I concluded in the journals and from Project 1 and 2. The similarity is my agent mimics human's knowledge and reasoning by using a library to transform and find similarities between images. Some simple transformations, addition, and subtraction are kind of similar to humans. The differences are:

1. Instead of comparing the pictures by recognizing shapes and patterns, my agent compares images by finding the relationship between pixels in a greater precision.
2. Humans tend to solve RPMs by recognizing shapes, but my agent solves them by comparing pixels and performing pixels operations.
3. In a completely visual approach, my agent cannot distinguish shapes and find the relations between shapes like what humans would do.
4. My agent is not able to learn or create knowledge like a human.
5. My agent has to check each condition sequentially. Sometimes, such sequence may give incorrect result. However, human usually solves the problem by trying out the easiest or most evident transformation first. I

believe it is possible to rank such relationships in code but deriving an exact order would be hard because some may in conflict to the others.

There are some limitations that impact my agent's overall performance:

1. In a completely visual approach, my agent cannot distinguish shapes and find the relations between shapes like what humans would do. This ultimately affects my agent's ability when it tries to count the number of different shapes.
2. My agent cannot separate out shapes if the location of shapes is unknown.
3. My agent would fail to compare the differences if the pictures are not perfectly aligned.

I think there are also other methods of solving RPMs as well. For example, we can use Convolutional Neural Network (CNN) to train our Basic Problem Set. The performance would be nearly perfect in training data, but I am unsure about the testing data. We definitely need more examples of RPMs to make it achieve high accuracy.

If I had more time or computational resources, I would spend time resolving the limitations I stated above. If we are allowed to use some existing libraries like OpenCV, distinguish and separate out shapes would be much easier.

In the end, I want to point out that I enjoyed doing the projects. I want to say thank you to you all who are managing this class.