Name: Balaji Srinivasan

GTID: 903103975

Project 3 Reflections

The previous two projects reflections are at the end of this document for the evaluators to easily understand how I have developed my agent over the course.

Overall Solution Design:

Project3 aims at solving 3X3 RPM problems mainly via the visual representations alone. I was really puzzled with how to start with image processing using visual representations alone. With great failure in my project2 resulting on very poor algorithm or approach need to start from scratch for Project 3. Since project 1 code works only for 2X2 problems. Also this is my first attempt on Image Processing.

Initial went through the following papers on the methods on which we can solve using Visual representations

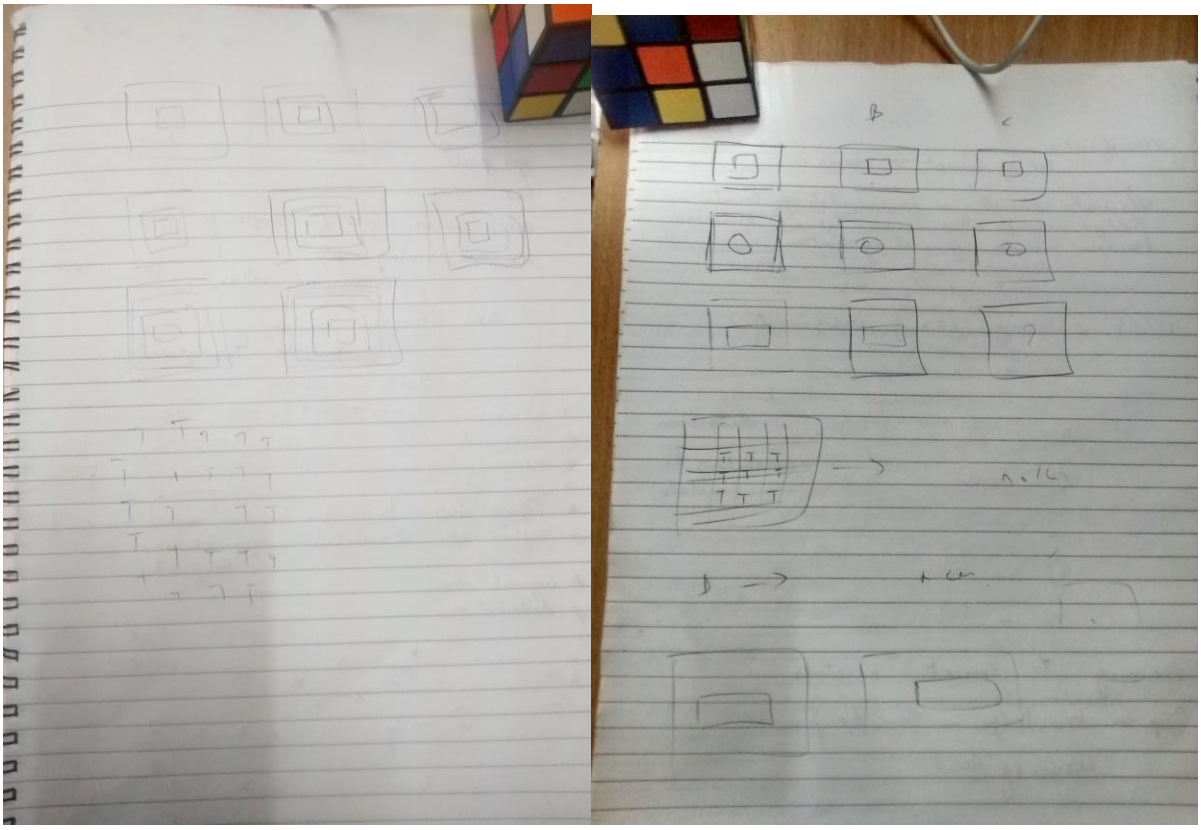
- Reasoning on the Raven's Advanced Progressive Test with Iconic Visual Representations- *Kunda,Mc Greggor and Goel*
- Addressing the Raven's Progressive Matrices Test of "General" Intelligence- *Kunda,Mc Greggor and Goel*
- Two Visual Strategies for solving the Raven's Progressive Matrices Intelligence Test- *Kunda,Mc Greggor and Goel*

After reading the papers provided the brief overview of how to approach the problems. In the meantime was also going through various Image processing techniques.

Draft Initial Design:

Started usually in arriving the basic pseudo code of how I am approaching the problems. After solving couple of problems as below got some mental picture on how to approach the problems via visual

representations.



Solving problems like this made me to easily realize how humans approaching the problem in a better way by just looking at the pictures. Even though that approach might be not exactly correct I was easily able to relate to the cognition that my agent should be able to have in order to solve the problems.

The idea or how I approached is basically plotting the entire array of pixels with True or false wherever application for each image. Then finding the difference or transition between the two.

Now I know the approach but how should I proceed with my coding. Because as said earlier I have never ever done image processing using Java.

Referring to an awesome post in Piazza on how to approach visual problems for first time gave me some insights. It also recommended learning from the exemplary projects. So referred couple of projects in exemplary projects in order to understand how Image processing is done via ImageIO and other core libraries.

In process of the learning got additional idea from the exemplary projects instead of just doing it against all the pixels one by one we can do it as overall weight. So without any reluctance started to try that.

The below is complete process in which my Agent for Project 3 evolved.

Agent Phase I

- Read all the images and converted into a simple collection with value of overall pixel count. Here Pixel count refers to the dark pixels alone.
- Once we have the complete the list of pixel count, next step is to find the transition or transformation occurring within the problems.
    - If the pixel count of A and B are same then there is no change
    - Made a basic rule that if pixel count of A is greater than B then the pictures are incrementing
    - If pixel count of A is less than B then pictures are decrementing.
- If there is no change in the transformation apply the basic rule against the G option and find the answer.

Problems or Issues in this Phase

Had the following problems

- Images pixel count varies across the options even for correct options
- There is no rule or algorithm to capture if incrementing or decrementing.

Agent PhaseII:

Ensured that the there is some accuracy level when comparing between the options and the answer that was found. This helped in fixing the issue of pixel count issues between images.

Later found over the piazza that people where converting the image to a single Binary Image such that these issues does not occur. But I never converted proceeded with problems in Basic Problems D.

- Most of the problems in D where of cyclic [I term it like that]. It means one of the options occur at particular interval of time or particular no of times.
- So improved the algorithm to accommodate that to identify the cyclic transformation type of problems.
- The algorithm I tweaked was

```
If transition of AB == transition of BC and transition of DE ==
transition of EF

      The transition type is no-change
```

```
If transition of AB == transition of DE and transition of BC ==
transition of EF

     The transition type is of column Wise

Else

     The transition type is of Cyclic
```

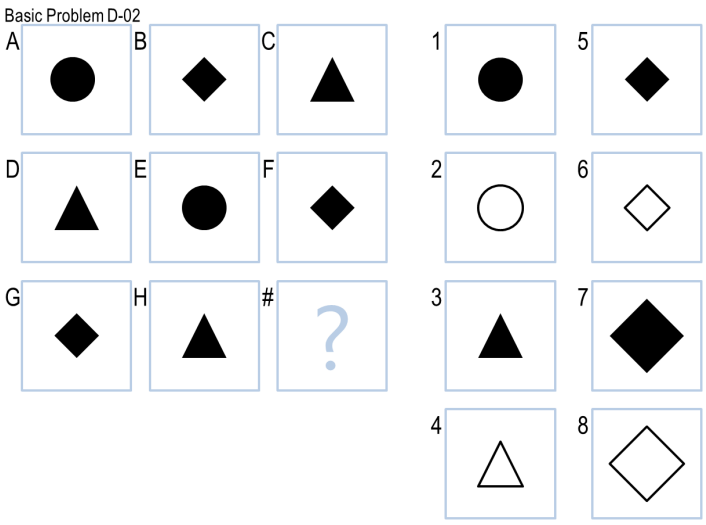
Now my agent was able to recognize the correct type of problems in D like human. But solving it became the puzzle.

So made an algorithm to solve the cyclic problems first.

Algorithm for Cyclic Problems:

- Looped through the Pixel Count Collection for each of Question Options
- Made a count
- If particular image lets say square image occurred 3 three times and triangle image occurred 3 times and circle image occurred 2 times alone then the missing image is circle

Below is one sample problem of this type



Problems or Issues in this Phase

Since there were few pixel issues the counts of the images of similar pixels were not matching and hence few of the answers got failed. In order to fix this made the accuracy level algorithm for this part of the agent too. Still I have not used binary image conversion to make the picture of uniform.

Agent Phase III

In this phase I improved the algorithm in which the cyclic problems are solved. The main tweaks I made was to compare based on the accuracy. I ensured that percentage difference between the expected pixel count and answer option is very minimal and selected the answer.

Agent Phase IV

In this phase I concentrated on the columnwise problems of D.

My exisiting algorithm did not worked. Hence I need to use my mental approach which I used in order solve this kinda of problems.

So for each of the problem I created a two dimensional array of 1 or 0 s . Which I will be using for the identifying the transformations. In order to identify transformation I need to write a basic set of algorithms which ensured that it solved Problems in D.[All Rules are described in Final Algorithm]

Sample Two Dimension array 

Agent Phase V

In this phase I mainly ensured that the transformations are identified efficiently using the pixels of each of the images. Introduced a new algorithm to identify the pattern of the problem from one of the below using pixel by pixel comparison using binary operations over the pixels

- Horizontal Sum
- Horizontal Difference
- Vertical Sum
- Vertical Difference
- Horizontal Difference Reverse Direction
- Horizontal Intersection
- Vertical Intersection

Improved the accuracy algorithm to ensure that It identified the correct answer to the problem

Agent Final Phase

In this phase I integrated the verbal methods which I used for Basic Problems B and the visual methods currently implemented.

Final Algorithm for Visual Method

- Made a collection of all the options pixel count
- Based on the collection identified the transformation
    - Cyclic
    - Column Wise
    - No Change
- If no change applied the transformation on H and identified the answer based on the pixel count comparisons
- If cyclic made count of the objects based on pixel count comparision and ensured that all the objects occur 3 times
    - In few cases ensured that AE transformation percentage is compared with E-I' transformations for accuracy.

- If in case there are some discrepancies did the opposite diagonal comparisons to ensure that accurate answer is choose correctly.

- Identify the pattern of the problem. Based on that applied the corresponding image processing techniques over the pixel to arrive at probable answer. Which is again compared with the options based on the accuracy level and answer selected finally based on accuracy level.
    - Horizontal Sum→ Merge Images G and H
    - Vertical Sum→ Merge Image C and F
    - Vertical Difference→ Difference between C and F
    - Horizontal Difference→Difference between G and H
    - Horizontal Difference Reverse→Difference between H and G
    - Horizontal Intersection→ Common Portions of G and H
    - Vertical Intersection→ Common Portions of C and F

Agent Metrics

Problem Type	Method	Number Problems Solved
B-Basic	Verbal	9
C-Basic	Visual	1
D-Basic	Visual	9
E-Basic	Visual	9
B-Challenge	Verbal	0
C-Challenge	Visual	1
D-Challenge	Visual	0
E-Challenge	Visual	2

Agent Mistakes

- The agent is not able to solve most of C type problems due to current algorithm used on the cyclic problems. Also current algorithm for no change type of problems also needs improvement.
- Agent does not learn from the mistakes it has done based on the correctness of the solution or does not record the pattern of the problems it has identified for future reference. If it has been recorded it would be easy for solving future problems

Risks and Payoffs

- One of the major risk I took is not converting the image to binary image in order to ensure the pixel count are equal. Instead I improvised the algorithm to measure the accuracy of the answer with more no of transitions using vertical, horizontal and diagonal methods which has really paid me well by solving 9 problems in both D and E type.
- The rules I created for identifying the transformation using the pixels of each options. I feel the rule is very vague and just based on the image comparisions using pixels which I created using trial and error and improvised slowly for each of the problems.

Project 2 Reflections

I have added the project 1 reflections at the end so that the evaluators will be able to understand how my initial design of the agent was. Click here for Project one Reflections.

Overall Solution Design:

Project 2 aims at solving 3x3 RPM problems. I tried to solve the 3x3 problem with my initial design. The design remains the same for solving the 3x3 problems too. But internal algorithm changes. For initial design click here.

Pseudo Code:

Huge changes happen in the overall logic or algorithm

1. First generate the Frames for A, B, C , D, E, F , G , H
2. Find the transformation from A→B ,B→C
3. Once we have the initial transformations we need to find the differences in these transformations lets name it as horizontalDifference.
4. Now we have a single row transformations and their differences
5. Our next step is to proceed to last row. Find the transformation from G→H.
6. Once we have the transformation we need to find the transformation from H→Options(1…8)
7. For each of the options we need to find the transformation and find the difference between the transformation and the G→H. Let's name it as horizontalDifference<optionno>.
8. As soon as we find the horizontalDifference<optionno> for each option we need to rate the differences. By comparing it against the horizontalDifference and rating it against scale of 10. If we get a rating of greater than 9 return the option as answer and loop for next problem.
9. In case if all the options ratings are not greater than 9 we need to store the ratings for each options and just the return the option which is having highest value of rating among them.

How Agent Works:

The above logic or design has not been implemented completely hence the below working model is based on the pseudo logic and the expected behavior of the agent.

Few changes has been made from Project 1 till now for the agent to work efficiently.

1. The RavensProblem is passed to the agent.
2. Check for the Problem Type
    a. If problem type is 2x2 then follow logic as here
    b. If problem type is 3x3 then check if it `hasVerbal ()` representation perform as per belowFirst the agent checks whether the Ravens Problem `hasVerbal ()` representation.
        i. If the Problem has the verbal representation then it extracts the Figures of options "A" and "B". This is achieved using `getFigures().get("<option value>")`
        ii. Then it find the transformation that happens between A→B as below
            1. For particular attribute it transformation returns "nochange" if both A and B has the same value for the attribute
            2. Else it returns the values as comma separated values which can be used during applying the transformation.
        iii. Then it find the transformation that happens between B→C similarly
        iv. Then it finds the horizontalDiff of these two transformation. Finding Differences works as below
            1. If both transformations of same set of shapes then it just add value of noShapes as "nochange"
            2. If both transformations has increase or decrease in the no of transformations value of noShapes is set with corresponding values.
            3. If the attributes values are changed between two transformation then assess the change either value change and store as comma separated values.
            4. If there is no change to the attributes then nochange is put against the attribute value.

v. Find the transformation from G→H.

1. Now we will be looping through the option values 1,2,3,…8. Find the transformation from H→Option.Find the difference of transformation as in step iv. Also evaluate their rating against the initial horizontalDiff on scale of 10 and store it if the value is less than 9.Else return the option as answer . This achieved by looping thorough all the `RavenFigures` in the problem and identifying the ones which have only numbers in it.

    a. During the comparison we rate values based on the attribute value equality.

    b. At the end overall weight for the option on scale of 10 is calculated.

vi. The agent identifies the highest rated transformations difference option as solution if none of the option rating is greater than 9.

Risks & Payoff in the Design:

The main risks of the agent are as below

- Considered only horizontal transformations. Also the horizontal transformation are considered only for the first and last row. The risk here is if in case there is pattern change over the second row in how the transformation occur agent may fail to solve the problem.
- Vertical transformations are not considered which is huge risk in getting the desired answer for the problems which has particular pattern.
- Modified the initial agent 2x2 solution for horizontal and vertical comparisons.

Scope of improvements

- More generalized shape transformation algorithm – Currently shape transformation are just based on the values passed as attributes. This can be improved by more generalized method in which lessons learnt in the past can be used from previous problems that are solved.
- Improving the algorithm to combine horizontal and vertical transformations into a single transformation to arrive at the answer.

- Image comparison to identify the Object Structure. And then applying the existing algorithm for identifying the solution

How new agents solve 3x3 similar to human

The new agent design is very much similar to how a human approach multi matrix problems. The human does not look out for all the rows or all the columns to solve the problem. It just takes a subset of them and generalize them and solve the problem. The agent behaves similarly. But the learning system how the human brain does for these kind matrix problems might not be there for this agent. But it can be improved to learn on the transformations patterns and generalizations and just apply them for many problems based on the accuracy of the agent.

Other Information

This agent logic for 3x3 is currently in progress and I will be able to complete it within days from now. I was not able to complete the development for this week due to travel priorities. Not a reason but letting the evaluators know the background. Once the development is completed may be I will be adding the results to this document and will try uploading to tsquare after the deadline to keep track of my progress.

Project 1 Reflections

Overall Solution Design

My agent is designed over the principles learnings by recording cases and case based reasoning approach. The agent works for all the problems which have verbal representations. For problems which does not have verbal representations the agent just skip the problem. Here the cases that are recorded already are on the representations of the verbal. Also based on the previous values the case based reasoning help in identifying the values of attributes.

Pseudo Code:

1. First generate the Frames for A, B, C
2. Find the transformation from A→B
3. Apply the same transformation from C which will result in C'
4. Now we have probable solution for the problem.

5. We have the recorded cases as problem solution options 6. We will find the nearest closest answer comparing the options against C'

6. Based on the option finding we have to record or store these cases such that for future problems can be used.

How Agent Works:

3. The RavensProblem is passed to the agent.

4. First the agent checks whether the Ravens Problem `hasVerbal ()` representation.

    a. If the Problem has the verbal representation then it extracts the Figures of options "A" and "B". This is achieved using `getFigures().get("<option value>")`

    b. Then it find the transformation that happens between A→B as below

        i. For particular attribute it transformation returns "nochange" if both A and B has the same value for the attribute

        ii. Else it returns the values as comma separated values which can be used during applying the transformation.

        iii. Using the transformation found the transformation is applied over C to identify the probable solution C'. [Generate the probable solution]

            1. If the attribute value in transformation is "nochange" then same value as that of the C is being transferred to C'

            2. Based on the attribute name match with the transformation and C the corresponding transformation is received.

                a. Further Based on particular attributes certain logics exists

                b. If "angle" then transformation angle value is subtracted from C angle value to get the exact angle

                c. If "alignment" then identify the alignment based on preset rules for the values of "bottom","right","left","top"

                d. If "fill" based on the transformation values the exact fill is identified using the logic of preset values.

            3. After applying the transformation we have the probable solution C'.

4. Now we will be looping through the option values 1,2,3,… compare with C' to find the most nearest solution. This achieved by looping thorough all the `RavenFigures` in the problem and identifying the ones which have only numbers in it.

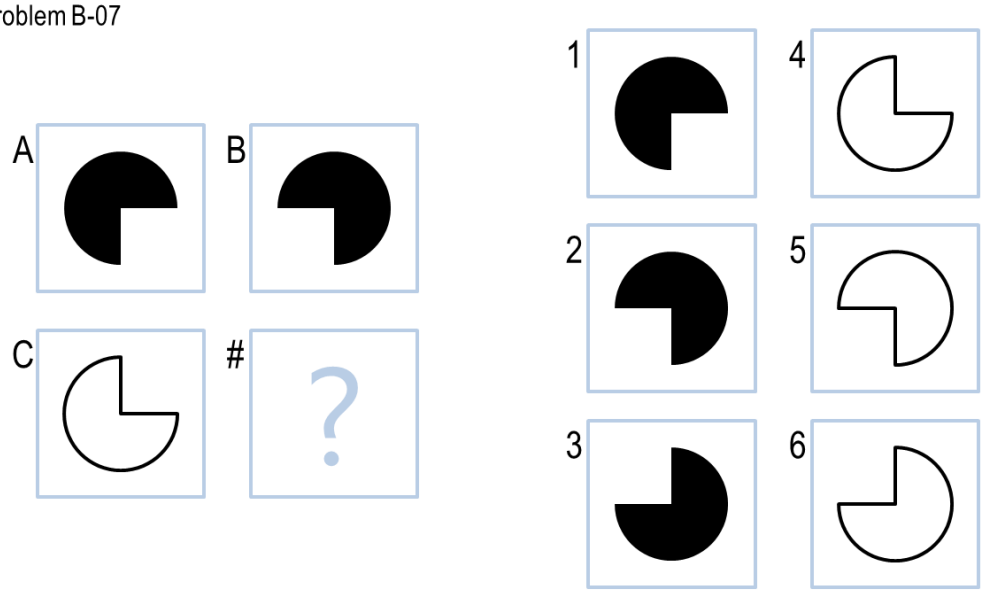
    a. During the comparison we rate values based on the attribute value equality.

    b. At the end overall weight for the option on scale of 10 is calculated.

5. The agent identifies the solution only if the weightage is greater than 9 for us to have the more accurate answer.

Please find the below example of the agent works for one of the Basic Problem

Basic Problem B-07



A	B	C	#

`shape:pac-man`
`fill:yes`
`angle:45`
`size: very large`
	`shape:pac-man`
	`fill:yes`
	`angle:135`
	`size: very large`
		`shape:pac-man`
		`fill: no`
		`angle:315`
		`size: very large`
			?

1. Identify Transformation between A→B

   Shape: nochange

   Fill: nochange

   Angle:90

   Size: no change

2. Apply C→C'

   Shape: pac-man

   Fill:no

   Angle:225

   Size: very –large

1	2	3	4	5	6
shape:pac-man fill:yes angle:45 size:very large	shape:pac-man fill:yes angle:135 size:very large	shape:pac-man fill:yes angle:225 size:very large	shape:pac-man fill:no angle:315 size:very large	shape:pac-man fill:no angle:135 size:very large	shape:pac-man fill:no angle:225 size:very large

3. Weightage rating for each of the options

1	2	3	4	5	6
5.0	5.0	5.0	7.5	7.5	10

Risks & Payoff in the Design:

The following are the risks I have taken in designing the solution

- For identifying the alignments we have considered only four options to identify. If incase if some other alignment attribute value to then the logic may fail. But for the basic problems these options work out well.

- Only horizontal comparison of A and B are considered based on which the transformation is arrived. This solves all the basic problems.

Agent Mistakes

- Currently the agent will be able to solve only 2X2 RPM problems. It's due to the hard coded selection of options for comparisons. This can be avoided by using the similar logic in which we have detected the answer option.

- My agent currently arrives at the transformation only based on horizontal comparison. But the current agent algorithm can be improved by have one more step to check for vertical comparison of A and C and arrive at one more transformation. Then we will be having two transformations each with can be applied on C to arrive at C' and C1'.This can be compared against the options for nearest solutions based on the weightage. Based the correctness of the solution the agent can learn on which type of comparison should be applied to arrive at answer in the future.

- Current agent does not able to solve problems which are of type addition or removal. The mistake agent has is it is not able to identify the exact transformation of these type of transition. The mistake here is currently the agent identify the transformation based on the number of attributes which will fail obviously for these kinds of problems. Change in the approach in which we identify the transformation should make it possible.

- Only solve problems which have verbal representations. This is because the overall algorithm is completely dependent on the verbal representations. This can be improved by introducing the algorithm for the image comparison.

Scope of improvements

- More generalized shape transformation algorithm – Currently shape transformation are just based on the values passed as attributes. This can be improved by more generalized method in which lessons learnt in the past can be used from previous problems that are solved.

- Improving the algorithm to perform the same for 3X3 problems

- Improving the algorithm of identifying the transformation to a more generalized way than based on values

- Image comparison to identify the Object Structure. And then applying the existing algorithm for identifying the solution

Other information related to Agent

Current version of the agent prefers verbal representation of the data. Because we would like to evaluate our agent's performance on the verbal data first and based on the same approach improvements can be done the agent to read the visual images and identify the object data and apply the existing approach to find the solution. Out of the 12 Basic Problems Agent is capable of solving 9 problems with accuracy of 100%. The performance of the agent is not too slow but considering the real time input lot of improvisations on the usage of data structure will obviously increase the performance to handle huge input data.

My Agent vs Human Cognition

In real time human would be solving the problem by identifying the image and building a pattern or data for that image. When comparing two images these data that are in human memory are retrieved to identify the transformation and apply the same on the question. My agent cannot be compared to human brain due its ability to solve the problems of varied type. My agent is capable of only solving 2X2 RPM problems. Human's brain capacity to build data based on the image is not available with the current version of the image. But it can be implemented.

Also human brain has a great learning capability from its previous way in which the solutions are handled. But agent learning mechanisms currently is at very low of 5% which needs to be improved to at least 40%+ in order to handle problems of various types.

In designing the solution I was able to learn a systematic approach of solving 2X2 RPM problems. But the systematic approach cannot be applied to all kind of 2X2 problems. The approach has to be improved to match in way at least closer approach of human brain solving.