Project 3 Design Report

Knowledge Based Artificial Intelligence
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332

INTRODUCTION

Raven's Progressive Matrices or RPM are pictorial tests of abstract reasoning which consist of a matrix of visual objects to measure the test takers reasoning ability. In each test the test taker needs to identify the missing image in the pattern from a choice of images. One may equate the test to human intelligence. The key in most raven problems is to determine the transformation of the objects or a group of objects to determine what the last object should be. An example of a RPM is shown in Figure 1.

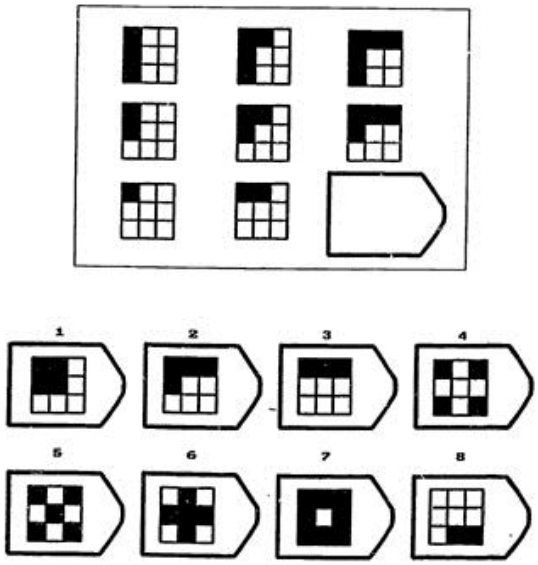

Figure 1. Example of an RPM problem

This design report sketches how an AI agent is implemented to solve RPM problems.

I. APPROACH TOWARDS THE DESIGN AND IMPLEMENTATION

The agent uses a multi-modal reasoning method where elements from an RPM problem are represented using a visio-spatial symbolic scheme, in which the elements can be manipulated symbolically but only with operations based in visual perception, e.g. translation, rotation, scaling, etc. The agent reasons over the images themselves without creating any verbal representations.

For the third project affine transformations like identity transform, horizontal, vertical, diagonal, rotation etc are used. In addition visual heuristics like dark pixel ratio, intersection pixel ratio, XOR/AND/OR differences between images and edge finding techniques are used. The approach builds confidence for each answer option after checking the given matrix relations in all possible directions for similarity using the above mentioned techniques. The answer having highest confidence is chosen. The approach falls under *Generate and Test strategy* where the generator is provided knowledge of possible transformations. The agent starts by examining frames for visual relationships and transformations that can be quickly detected by visual inspection. Next the agent uses *Incremental Concept Learning*. The agent uses image similarity ratio defined by Tversky[9] to match between images after applying various approaches. If this process detects the presence of one of the relationships within a matrix problem, the agent generates a prospective solution and looks for a matching answer. The more an answer option matches with the generated answer, higher will be its confidence.

II. DESIGN OF THE AGENT

The agent uses a visual approach to reason over the problems. It tries to reduce the input space to something both manageable and meaningful so that it can compute and correctly guess an answer from the given choices. The agent takes each possible answer choice and computes the likelihood it is correct. To do so, the agent takes a series of measurements capturing the relationship between each training pair in horizontal, diagonal and vertical directions. It then compares those measurements against each of the answer options, the combinations of any cell adjacent to the empty slot and each answer choice. Each comparison, if significant enough, casts a vote for the current answer as the likely answer with a weight directly proportional to the believed similarity of the cells.

First the agent checks for affine transformations. If using Affine transformations the agent finds a solution with high probability it adds to the confidence for that answer. Then the agent checks for dark pixel ratio between pairs in both horizontal and vertical directions. If not it checks the edges of the input images to detect similar shapes. If any training pair has similar shape the agent checks the answer choices for similarity with the given images for shapes. If found the answer's confidence level is increased further. Then the agent checks for addition and deletion of elements in the images using XOR/OR/AND operations. If a answer choice is found using this approach then answer option's confidence is increased. Finally the answer choice with he highest confidence is chosen. If no answer choice has any confidence value(all values are 0), the agent skips this problem as it could not determine any answer with sufficient confidence.

Figure 2 explains the process using the design approach using a flow chart.

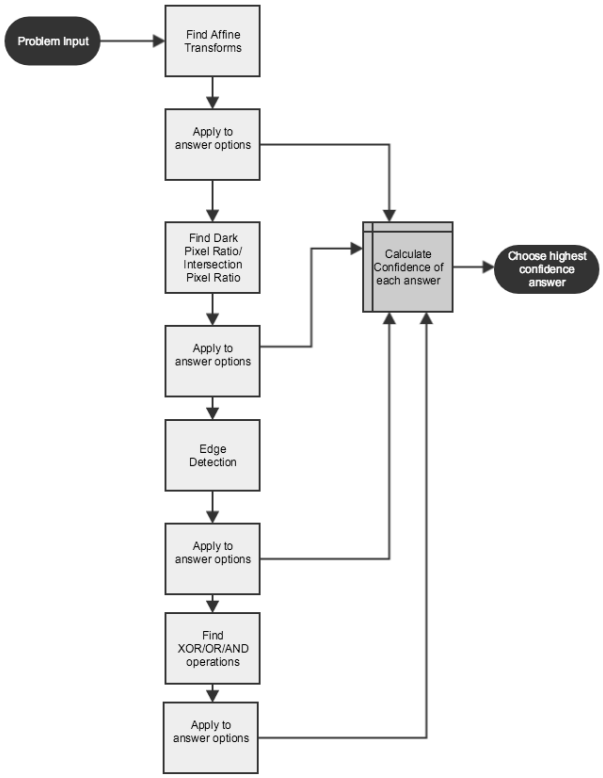

Figure 2. Flowchart for the Agent reasoning

III. Implementation of the Agent

The agent uses a hybrid approach to find the correct answer choice. The relationship measurements like dark pixel ratio, affine transformations, edge detection, intersection ratio, XOR, OR or AND operations are applied to the images to find the best possible answer match.

In any Ravens problem there exist simultaneous horizontal, vertical or diagonal relationships which must be maintained. In Figure 3, these relationships are illustrated using an example problem. As shown, relationships H_1 and H_2 constrain relationship $\mathcal{H}$, while relationships V_1 and V_2 constrain relationship $\mathcal{V}$. Also in Problem set D and E we observe diagonal relationships D_1 and D_2 which constrain relationship $\mathcal{D}$.

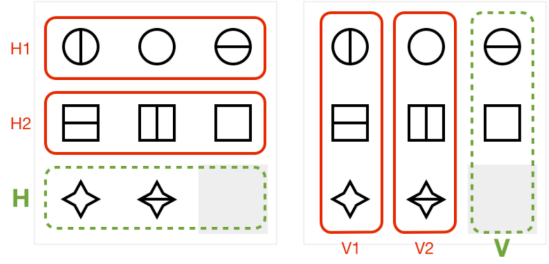

Figure 3. Relationships in a 3x3 matrix as mentioned in [8]

1) General Relation using Incremental Concept Learning: The general relation across Horizontal, Vertical and Diagonal directions is calculated from individual relations. We generalize over both H_1 and H_2 for coming up with $\mathcal{H}$. Similarly we generalize over both V_1 and V_2 to decide on $\mathcal{V}$. We generalize over D_1 and D_2 to come up with $\mathcal{D}$. The resultant general relation across these directions will be the intersection of the relations between adjacent matrix elements. For $\mathcal{H}$ is calculated as follows

$$\mathcal{H} = \{H_1(\theta) \cap H_2(\theta)\} \tag{1}$$

Similarly, $\mathcal{V}$ and $\mathcal{D}$ is calculated as

$$\mathcal{V} = \{V_1(\theta) \cap V_2(\theta)\} \tag{2}$$

$$\mathcal{D} = \{D_1(\theta) \cap D_2(\theta)\} \tag{3}$$

2) Calculating Confidence τ :: Confidence is calculated for each option element for answer. The option element having highest confidence is chosen as the answer. While calculating *Confidence* the similarity with the relation found in either $\mathcal{H}$ or $\mathcal{V}$ is checked with that of relation between A_i and either H or F. The generated image after applying the relation is checked for similarity using the Tversky's ratio[9].

Let the similarity between the Horizontal relation H and $A_i(\gamma)$, where γ is the relation between A_i and H be denoted by $S(\mathcal{H}_\theta, A_i(\mathcal{H}, \gamma))$. Similarity between Vertical relation $\mathcal{V}$ and $A_i(\gamma)$ i.e. $S(\mathcal{V}_\theta, A_i(\mathcal{V}, \gamma))$ is also found. Similarly in diagonal relation $\mathcal{D}$ and $A_i(\gamma)$ i.e. $S(\mathcal{D}_\theta, A_i(\mathcal{D}, \gamma))$ is also found. θ is either 1 or 2 in both cases which signifies relation in 1st row/column/diagonal or 2nd row/column/diagonal. The Confidence is then calculated as follows [8] :

$$\tau = \{S(\mathcal{H}_\theta, A_i(\mathcal{H}, \gamma)) \cup S(\mathcal{V}_\theta, A_i(\mathcal{V}, \gamma)) \cup S(\mathcal{D}_\theta, A_i(\mathcal{D}, \gamma))\} \tag{4}$$

A. *Affine transformation*

The affine transformations algorithm[4] seek to discover a single transformation T that holds across any of (A and B) or (A and C) of the RPM problem matrix. The algorithm has a set of basic transformations from which it can draw specific candidate transformations to test which among the two pairs has a stronger relationship. The algorithm proceeds as follows, in the case of a two-by-two RPM matrix problem:

1) For each transformation T_i:
   a) For A and B, check to see if T_i holds.
      i) If so, apply T_i to the B and compare with C. Compare using Tversky's ratio.
      ii) Check if same T_i holds for pairs (D,E) and (E,F). If not check vertical transformation, diagonal relationship.
   b) For A and D, check to see if T_i holds.
      i) If so, apply T_i to the D and compare with G. Compare using Tversky's ratio.
      ii) Check if same T_i holds for pairs (B,E) and (E,H).

iii) If there are no more transformations in memory, then we exit this algorithm.

2) For each answer choice A_i:
   a) Compute the Tversky's ratio between the predicted image and A_i. Add this to the confidence for this answer option.
3) Choose the answer option A_i having the maximum confidence.

However the affine transformations algorithm will not always produce a solution. The memory set of possible relations is restricted. This is done in order to prevent the algorithm from stalling indefinitely on a single problem.

The priority of the transformations is set with the help of the similarity weights. While deciding on the weights to assign to each kind of transformation, it is of primary importance to consider which transformations are more evident to the human mind. While weighing up the relevance of other transformations, it is pertinent to order them in ascending order of disruption; wherein those transformations which subtly alter the image (such as rotation or reflection) are given greater weights than transformations which drastically alter the image (such as deletion of an image, or a complete change in shape).

Example : We apply the above algorithm to the problem in Figure 4. First the affine transformation between (A,B) and that between (B,C) are found. For (A,B) the agent detects *Rotate 90* which is also similar to affine transform relation between (B,C). Similarly the relations between (D,E) and (E,F) is also *Rotate 90*. Then we apply *Rotate 90* on H and find out the Tversky's Ratio the resultant image and answer option. Answer choice 1 has the highest similarity and hence it will have the highest confidence. Hence the agent decides on option 1.

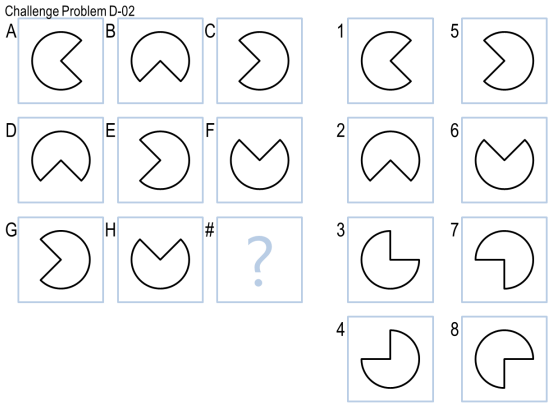

Figure 4. Affine Transformation example

B. *Dark Pixel Ratio comparison*

We calculate the dark pixel ratio between image pairs and if the pairs in H_1 have similar ratio to pairs in H_2 we use the dark pixel ratio between (G, H) and (H, A_i) to determine the confidence of answer A_i. Here is the pseudo code for the algorithm for horizontal direction:

1) Calculate dark pixel ratio between the image pairs (A,B) and (B,C). Let $darkABC$ be the ratio $(darkAB)/(darkBC)$.
2) Calculate dark pixel ratio between the image pairs (D,E) and (E,F). Let $darkDEF$ be the ratio $(darkDE)/(darkEF)$.
3) If $darkABC - darkDEF < \epsilon$, for each answer choice A_i:
   a) Calculate dark pixel ratio between the image pairs (H,A_i) and (G,H). Let $darkAns$ be the ratio $(darkGH)/(darkHAns)$.
   b) If $darkABC - darkAns < \epsilon$, add confidence equal to the reciprocal of $(darkABC - darkAns)^2$ to the answer option.
4) Choose the answer choice A_i that has the maximum confidence.

Example: We apply the above algorithm to problem in Figure 5.

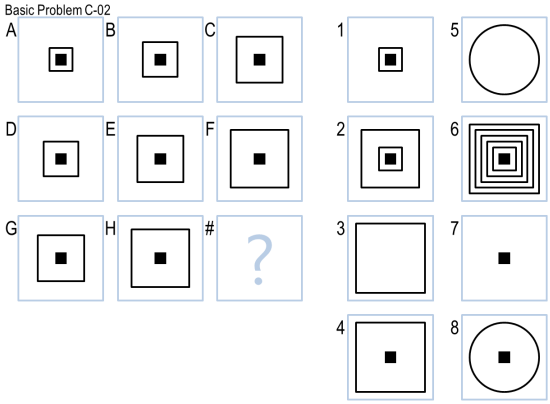

Figure 5. Dark Pixel Ratio example

$darkAB$	0.767857142857
$darkBC$	0.826388888889
$darkABC$	**0.929171668667**
$darkDE$	0.810586176728
$darkEF$	0.84855233853
$darkDEF$	**0.95525772533**
$darkGH$	0.840441176471

ϵ here is 0.05 . After calculating the $darkAns$ for each option A_i with $darkGH$ we find the following confidence for the options.

Option 1	Option 2	Option 3	Option 4
0.0	0.0	0.0	0.9684176822179952
Option 5	Option 6	Option 7	Option 8
0.0	0.0	0.0	0.0

The agent then chooses Option 4 as the answer.

C. *Edge detection and Intersection pixel Ratio comparison*

We apply the next technique which is based on Edge Detection of the figures and comparison. Using Python PIL

the edges of the cells (A and B) or (C and D) are found out individually.

```
img1 = figureOneImage.filter(ImageFilter.
    FIND_EDGES)
```

Then using Intersection pixel ratio the similarity of the two images are compared. Intersection Pixel Ratio is the difference in percentage of the number of dark-colored pixels present at the same coordinates with respect to the total number of dark-colored pixels in both matrix cells for a given set of contiguous pixels. Similarly the answer option having the highest similarity between the dark pixel ratio with the either B or C (whichever is least similar to the edge found in A). The algorithm proceeds as follows :

1) Edge detection is performed on all the images A, B and C. Let A', B' and C' be the images formed after applying Edge detection respectively.
2) Intersection Pixel Ratio between A' and B' and also between A' and C' are calculated. The pair having higher intersection ratio is chosen.
3) For each answer choice A_i:
   a) Apply edge detection to A_i to get image $A_{i'}$
   b) Calculate the intersection pixel ratio between $A_{i'}$ and the remaining of either B' or C'.
4) Choose the answer choice A_i that maximizes the intersection pixel ratio with the cell in the problem.

Example : We apply the above algorithm to the problem in Figure 6. First we apply the edge FIND-EDGES method to all three of A, B and C. Then we calculate the intersection pixel ratio of both pairs (A and B) and also (A and C). The pair having higher intersection ratio get a higher value for match, here (A' and B'). We then apply *FIND-EDGES* method to all the answer choices and then compare each answer option with B and C respectively and find out the intersection pixel ratios between them. Answer choice 5 after applying *FIND-EDGES* return the highest value when compared to C' for intersecting pixels with just **120 pixels different out of 33856 pixels** in all.

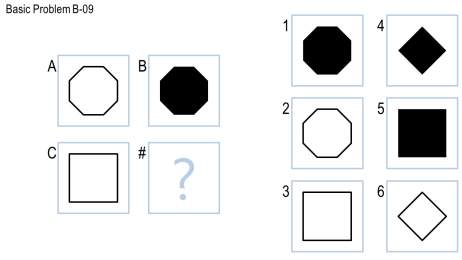

Figure 6. Edge Finding example

D. XOR, OR, AND operation and dark pixel ratio comparison

The fourth measurement technique used is the XOR, OR, AND operation between images. When images in the cells have some parts added or deleted between them, applying these operations on image pairs gives the part that is subtracted or added, highlighted. The algorithm provided below is for XOR operation. Similar are the algorithms for OR and AND operations.

1) XOR the image pairs (A,B) and (B,C). Let XOR(A,B) and XOR(B,C) be the images formed respectively. Calculate dark pixel ratio for the pair XOR(A,B) and XOR(B,C).
2) XOR the image pairs (D,E) and (E,F). Let XOR(D,E) and XOR(E,F) be the images formed respectively. Calculate dark pixel ratio for the pair XOR(D,E) and XOR(E,F).
3) For each answer choice A_i:
   a) Calculate XOR(H,A_i) and XOR(G,H)
   b) Calculate dark pixel ratio for XOR(H,A_i) and XOR(G,H). Add confidence equal to the reciprocal of *Difference of dark pixel ratio between* $XOR(H, A_i)$ *and* $XOR(G, H)$ to the answer option.
4) Choose the answer choice A_i that has the maximum confidence.

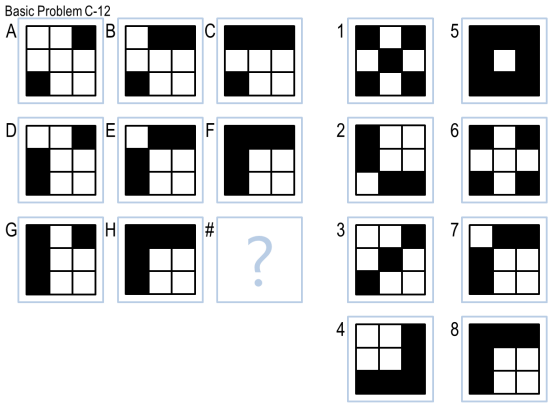

Figure 7. XOR operation example

Example : In Figure 7, $H_1(\theta)$ can be found by applying action XOR on either two of the matrix elements pair in the first row. Applying XOR to both gives the similar dark pixel ratio. Again applying AND operation gives same number of black pixels in each pair. We observe that the number of squares in the first row of each individual matrix element is increasing. Hence our initial concept of the relation will be a addition of new dark squares to each matrix element and also constant number of squares apart from that i.e two (one in upper right corner and one in lower left corner). Similar relation can be found by XOR with both pairs in H_2 also. From Equation 1 we have for our example the following :

$$\begin{aligned} \mathcal{H} &= \{H_1(\theta) \cap H_2(\theta)\} \\ &= \{(XOR, AND_1) \cap (XOR, AND_2)\} \\ &= \{XOR\} \end{aligned} \tag{5}$$

In Equation 5, we get $\mathcal{H}$ as XOR since only the XOR operation in both H_1 and H_2 give same result. Hence we

generalize the relation and found that the common relation in our problem is XOR. Applying XOR to third row will possibly give us a answer.

$\mathcal{H}$ is XOR and also $\mathcal{V}$ is XOR as can be observed from the figure. Applying XOR between all option images and horizontal matrix image element F in figure 7 will give us $A_i(\mathcal{H}, \gamma)$. We observe that XOR operation with option 8 will give similar dark pixel ratio with F as we observe in pairs (A,B) or (B,C) etc. Similarly in vertical direction too $A_i(\mathcal{V}, \gamma)$ will give a similar dark pixel ratio with any vertical pairs. The ratio will be most similar in case of option 8 and hence its confidence ratio will be highest. With this information the agent will choose answer option 8 for this problem.

IV. Accuracy, Efficiency & Generality

A. Accuracy

The agent performed well on the Basic Problem set B, C, D and E solving correctly all the 12 problems in each set. The agent also solved 12 Challenge problems correctly. It is to be seen how well the agent performs over the test problems.

B. Efficiency

The agent seemed to perform quite efficiently along with the accuracy. The execution times for the basic problems are summarized in the below table.

For Basic Set B :

Problem	*Time for execution(in secs)*
Set-B Problem 1	0.746165037155
Set-B Problem 2	2.22168493271
Set-B Problem 3	0.216490983963
Set-B Problem 4	0.80955696106
Set-B Problem 5	1.86861205101
Set-B Problem 6	2.02880120277
Set-B Problem 7	2.36169791222
Set-B Problem 8	2.51885199547
Set-B Problem 9	3.41495609283
Set-B Problem 10	3.99423599243
Set-B Problem 11	3.53287005424
Set-B Problem 12	3.34531593323

Total Time : 27.069453001 seconds

For Basic Set C :

Problem	*Time for execution(in secs)*
Set-C Problem 1	0.80054807663
Set-C Problem 2	3.16310095787
Set-C Problem 3	4.02250599861
Set-C Problem 4	3.81651210785
Set-C Problem 5	4.38305783272
Set-C Problem 6	4.06873607635
Set-C Problem 7	3.05666685104
Set-C Problem 8	3.75078296661
Set-C Problem 9	2.98950004578
Set-C Problem 10	3.20583510399
Set-C Problem 11	0.991039991379
Set-C Problem 12	4.71316790581

Total Time : 38.9793188572 seconds

For Basic Set D :

Problem	*Time for execution(in secs)*
Set-C Problem 1	0.759038925171
Set-C Problem 2	3.00184297562
Set-C Problem 3	3.36692810059
Set-C Problem 4	7.98400187492
Set-C Problem 5	5.32661890984
Set-C Problem 6	5.68460202217
Set-C Problem 7	5.02895617485
Set-C Problem 8	4.94763588905
Set-C Problem 9	5.4588830471
Set-C Problem 10	4.99197602272
Set-C Problem 11	3.21445798874
Set-C Problem 12	4.35244393349

Total Time : 54.1206860542 seconds

For Basic Set E :

Problem	*Time for execution(in secs)*
Set-C Problem 1	5.62871193886
Set-C Problem 2	5.65607309341
Set-C Problem 3	6.04926395416
Set-C Problem 4	4.32282495499
Set-C Problem 5	4.99113202095
Set-C Problem 6	4.86555194855
Set-C Problem 7	4.95998001099
Set-C Problem 8	4.52058887482
Set-C Problem 9	4.78757190704
Set-C Problem 10	4.67517590523
Set-C Problem 11	4.74486994743
Set-C Problem 12	3.99122095108

Total Time : 59.1964411736 seconds

C. Generality

The agent uses four approaches to solve all the problems. The first approach, i.e the affine transformation solve 8 of the Set B Basic problems and 2 of Set C Basic problems, 4 of set D and E. Dark Pixel ratio calculations solves 8 out of the Set C Basic problems. The Edge Detection approach helps to solve 1 of the Set B Basic problems and 3 of the challenge problems correctly. The XOR/AND/OR method solves 12 of the Set D and E Basic problems. The approaches used are quite general and helps to solve a wide range of problems with the same techniques.

V. Shortcomings of the Agent Design

In the affine transformation algorithm above, there are two biases in searching for the correct transformation.

- There is a bias in examining rows first, then columns and finally diagonals.
- There is a bias in the order in which transformations are checked.

These biases would not matter if there were exactly one transformation for a given matrix problem. However, the

presence of ambiguity in some problems means that, if two or more transformations hold, or if a transformation holds for both row and column, the algorithm will use the first one that crosses its path, which may or may not be the correct one.

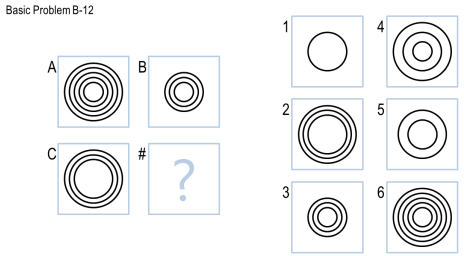

Figure 8.

Figure 8 gives an example of a problem with this kind of ambiguity; this problem is analogous to one in the actual SPM test. One transformation that could hold in this problem would be, across the top row, deletion of two outer rings. Using this transformation on the bottom row, the predicted answer would be identical to answer 1.

However, one could also have a transformation across the first column that can be interpreted as the inner three rings scaled and covered the outer rings. Applying this transformation to the right column would predict an image identical to answer 2. So which one is the correct answer?

There are some other inefficiencies within the agent design which leads to decrease in performance and solving time. The most major bottleneck is calculating and comparing pixel by pixel every image. This process repeats the calculations required to find the best match for an transform by accessing same pixels for B and C images multiple times. Also this bottleneck is dependent on the number of objects within a figure. Therefore, as the number of objects in a figure rises, so does the run time of the agent. For the 3x3 matrices this causes huge run times for each problem with maximum going above 4 seconds.

VI. Possible Improvements to the Agent Design

There are several ways in which the such shortcomings could be resolved [4]. Reiterating some of them in the current context below :

- First, the algorithm could just stick with its first prediction, and sometimes its biases would result in the correct answer being chosen, and sometimes not.
- Another option is to explicitly assign biases to the transformations in memory, where, for example, a transformation comprised of a single operation (e.g. reflection) would be examined before transformations comprised of multiple operations (e.g. rotation plus scaling). This approach puts additional requirements on the memory store and assumes that such rankings of transformations are somehow justified.
- A third approach is to add an additional step to the algorithm that examines the final matrix (with the predicted answer choice in place) for qualities of symmetry or other global measures. This approach raises questions as to what precisely the RPM tests are measuring, if such problem ambiguities are resolved by a fixed, global scale of optimal answers. It could be that aspects of symmetry play a key role, in which case it would be interesting to compare the demands of computing symmetry on the RPM to the requirements for recognizing symmetry in other cognitive tasks.
- Other improvements are incorporating meta reasoning into the process the agent takes to solve a problem. Meta reasoning could be used to determine which method of solving a problem is most likely to produce the most accurate result. If best approach is known before solution of the problem begins, then there isnt has much of a benefit to solving the problem using multiple approaches. Therefore, this would limit the number of times that the agent has to solve the problem to one and greatly decrease processing time per problem.

VII. AI agent vs Human Cognition

The design of this agent proves that sometimes using multiple approaches to solving a problem can prove to be beneficial as one approach will perform better in certain situations where it others might be produce as accurate of results. This approach is similar to what humans do when solving complex logic problems. It is common that we will apply different strategies that we have learned to a problem and then use the collection of information we have gathered from the application of each strategy to derive our conclusion. We often refer to this approach as exploring the different options we have to solve a problem. More typically, humans will only consider each approach which they are aware of to solve a problem, determine the best approach and solve the problem that way. The agent mimics the human way of solving problems using a visual approach. The human mind tries to find the solution by applying different transformations to the image and checking which transformation leads to the solution. Although the number of transformations in disposal to the human mind is by many times higher than that of the agent, the approach is similar. Again when the agent uses OR/AND/XOR operations, the agent mimics the human approach of detecting similarity, addition and deletion of objects in the images and then comparing them.

References

[1] Winston, Patrick Henry, *Artificial Intelligence*, 3rd ed. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1992.

[2] P. Norvig and S. Russel, *Artificial Intelligence, A Modern Approach*, 3rd ed. Upper Saddle River, New Jersey: Prentice Hall, 2010.

[3] S. Belongie et.al., *Shape Matching and Object Recognition using Shape Context*. IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol 24, No 24, April 2002.

[4] M. Kunda, K. McGreggor, A. K Goel, *A computational model for solving problems from the Raven's Progressive Matrices intelligence test using iconic visual representations.*. Cognitive Systems Research, 22-23, pp. 47-66.

[5] M. Kunda, K. McGreggor, A. K Goel, *A fractal approach towards visual analogy*. In Proceedings of the International Conference on Computational Creativity, Lisbon, Portugal, January 9 2011.

[6] K. McGreggor and A. K Goel, *Fractal analogies for general intelligence*. In Artificial General Intelligence. Springer. 177188, 2012

[7] D. Joyner et. al., *Using Human Computation to Acquire Novel Methods for Addressing Visual Analogy Problems on Intelligence Tests*. Proceedings of the Sixth International Conference on Computational Creativity, June 2015.

[8] K. McGreggor and A. K Goel, *Confident Reasoning on Ravens Progressive Matrices Tests*. Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, June 2014.

[9] Tversky, A., *Features of similarity*. Psychological Review. 84(4):327, 1977