Simple-RPM

Kabir Abdulmajeed[1], David Joyner[2], and Ashok Goel[3]

***Abstract*— This work presents a novel method for solving the Ravens Progressive Matrices (RPM) problem using a visual heuristics based computational AI framework. The Simple-RPM algorithm is a simplified and efficient method based on three stages: re-representation - a global contractive transformation, prediction stage, and a matching stage. In the re-representation phase, an integral score for each image is computed, closely followed by the prediction stage where the integral score corresponding to the missing RPM element is predicted. Finally, the predicted integral score is used to find the matching test image in the matching stage. The Simple-RPM performs above average on the Basic-B and Basic-C RPM problem sets. The adjoining code shows how this Agent can be implemented in roughly 10 lines of python code.**

I. INTRODUCTION

The Ravens Progressive Matrices (RPM) test is a commonly used test of general intelligence based on visual similarity and analogies [REF]. It is highly rated due to its correlation with other intelligence tests. Understanding and solving these tests are often open-ended and require background knowledge, common sense reasoning, and mental imagery [REF]. It also often serves a useful tool in psychometrics - the theory and technique of quantitative measurement of psychological variables such as intelligence and aptitude.

The RPM Problem contains a sequence of images in rows and columns that form a square matrix with a single missing element. Their formulation suggests pairwise analogies within rows and columns of the matrix. The problems are divided into sets with increasing difficulty. There are three published RPM versions: SPM - Standard Progressive Matrices, APM - Advanced Progressive Matrices, and CPM - Colored Progressive Matrices [REF]. The SPM is made up of 5 different problem sets with increasing difficulty, and each problem set contains 12 RPM problems each. It contains geometric analogy problems in a matrix of size (2×2) or (3×3) in more complex situations. Figure 1 represents an RPM problem with 8 input image elements (A-H) on the left, and 8 test images (1-8) on the right. Image 'A' in Figure 1 is referred to as an RPM input element, while image '5' is referred to as an RPM test element. Pixel values are either 0-black, or 255-white.

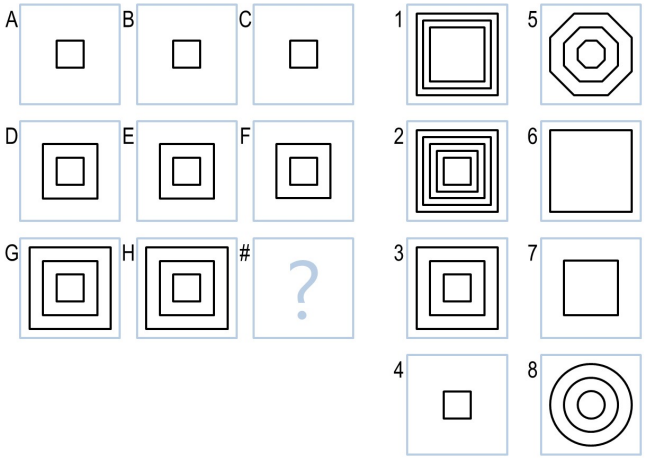

Fig. 1. A (3×3) RPM Problem Set

II. RPM AI AGENTS

Research on intelligence tests can be traced to Evans [REF] and has huge applications in Artificial intelligence. As pioneered by Alan Turing, many researchers in AI have built systems evaluated on intelligence tests. Majority of the computational approaches to solving the RPM rely on the translation of visual inputs to propositions [REF, REF, REF]. According to Hunt in [REF], two qualitative solutions to the Advanced Progressive Matrices (APM) problem are: (1) Gestalt - based on visual representations and perceptual operations, and (2) Analytic - based on feature representations and logical operations. It is interesting to figure if qualitatively different representations solve the RPM problem, as this abilities may extend to other domains of reasoning in cognitive tasks.

Verbal and Visual approaches are two common ways for solving the RPM problems. Similarly, the Fractal and Affine algorithms are two visual methods of solving RPM problems [REF]. They present a unique method of solving the RPM problem by direct image transformations without initial conversion to representative propositions. The Affine method [REF] relies on extrapolating visual image transformations (rotations, flips, mirroring, shading) that gives the best similarity within row or columns of the RPM matrix and applying the best transform to generate the answer. This answer is similarly matched with the test images for confirmation. On the other hand, the Fractal [REF] approach

*This work was not supported by any organization

[1]M. K. Abdulmajeed is with the Computer Science department, Georgia Institute of Technology, Atlanta, GA, USA. kbmajeed1 at gatech.edu

[2]D. Joyner is with the Computer Science department, Georgia Institute of Technology, Atlanta, GA, USA. d.joyner3 at gatech.edu

[3]A. Goel is with the Computer Science department, Georgia Institute of Technology, Atlanta, GA, USA. a.goel at gatech.edu

is based on decomposing the input images into grids (fractal encodings). Thereafter, each decomposition is manipulated by transformations to match elements within the RPM matrix. These transformations form compact representations that can be used to solve the RPM problem. A good account of the history of RPM solving methods can be found in [REF, REF, REF].

III. The Simple-RPM Algorithm

The Simple-RPM to the best of our knowledge is a novel approach to solving the RPM. It is based on a unique representational structure. The RPM problem is first transformed via a computational model into numerical values that depict the essence of the RPM image pixels. Unlike propositional computational approaches to solving the RPM, the Simple-RPM relies on the direct visual translation of the RPM problem into new forms from which analogies are deduced. Due to the pairwise analogies within the RPM problem sets, the algorithm also assumes that RPM elements within a row, column, or diagonal are related quantitatively. The advantage of this algorithm lies in its minimalist implementation and generalized structure. Unlike the Simple-RPM algorithm, the affine method requires initialization of desired transformers which can be large. Similar to [REF], the Simple-RPM algorithm proves that some of the RPM problems can be solved using purely visual representations.

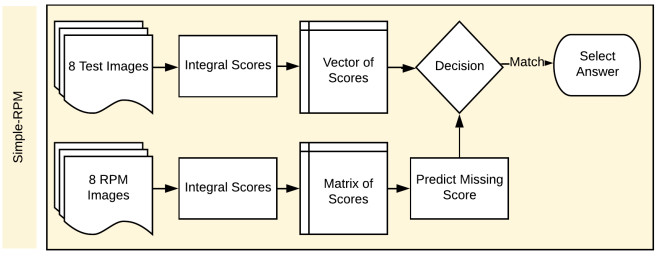

Fig. 2. Description of the Simple-RPM Algorithm

A. Global Re-representation

This is referred to as the re-representation stage. The objective is to obtain an integral score for all elements in the RPM problem. For a (3×3) RPM problem, this corresponds to input elements - A, B, C, D, E, F, G, H and test elements - $1, 2, 3, 4, 5, 6, 7, 8$. The missing element I from the RPM Problem Set needs to be found among the test elements. The integral score is computed for each image in the RPM problem set using equation (1) and (2) which corresponds to the integral image and its standard deviation respectively.

$$I_{\int(x,y)} = \sum_{x}\sum_{y} I(x,y) \tag{1}$$

The integral image, also known as a summed-area table corresponds to the cumulative sum of the image pixels. The popular Viola-Jones face-detection algorithm works efficiently by employing integral images [REF].

$$\chi = \sqrt{\frac{\sum(I_{\int(x,y)} - \bar{I}_{\int(x,y)})^2}{x \cdot y}} \tag{2}$$

This stage is a global operation that is opposite of the image fractal decomposition described in the Fractal approach [REF]. Instead of breaking down the image into segments, they are taken as a whole via the integral equation (1). Thereafter, the standard deviation (2) indirectly provides a measure of the dispersion of pixels in the images - a metric that captures patterns. This way, the entire input set $(2\times2)-1$ or $(3\times3)-1$ of the RPM problem set becomes a matrix of integral scores. The '-1' corresponds to the missing element being sought.

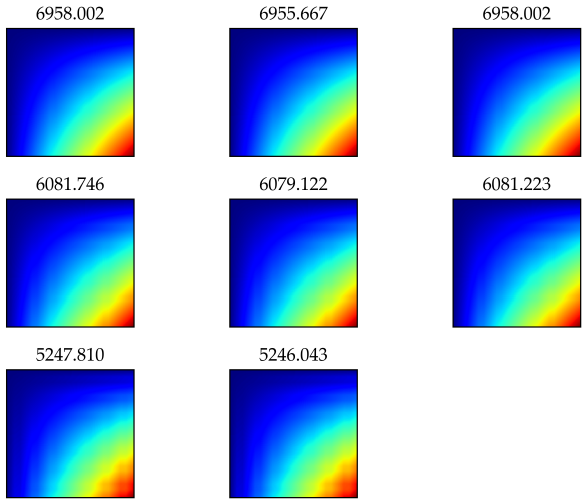

Fig. 3. Integral images and scores for the Input RPM elements

B. Prediction Stage

This stage is where analogical reasoning occurs. Given a matrix of $(2\times2)-1$ or $(3\times3)-1$ known elements (Figure 1), how can we predict the missing "1" element? If the data was large, say an $(n\times n)-1$ case, then an intuitive way to find the missing element is by linear or nonlinear modeling. Since the matrix is small however, a gradient is used to calculate $\hat{I}$ which is an estimation of the integral score for the missing matrix element I. There are different directions to calculate the gradient. In the RPM problem, analogies can be found by exploring three directions: row, column, and diagonal. Along the last row, the elements include: $G, H, ?I$, along the last column, the elements include: $C, F, ?I$, and along the diagonal, the elements include: $A, E, ?I$ - where $?I$ is the missing element. These are the three most plausible analogical relationships - similar to how humans would see the RPM problem.

.
.
.

.

$$\hat{I} = \Delta \cdot H,\ where\ \Delta = H/G \tag{3a}$$

$$\hat{I} = \Delta \cdot F,\ where\ \Delta = F/C \tag{3b}$$

$$\hat{I} = \Delta \cdot E,\ where\ \Delta = E/A \tag{3c}$$

Finally, the gradient Δ is calculated using either (or combinations) of equation (3a), (3b), (3c) corresponding to the "along-row", "along-column", and "along-diagonal" analogies respectively. These equations rely on the assumption that a linear quantitative relationship exists within the RPM Problem Set. The estimate of I drives the next stage - matching.

C. Matching Stage

The matching stage is straightforward. Here, we are looking for the test image that closely aligns to the missing element. Since all the test images have been converted to integral scores, a simple distance metric is used to find the test image that has the closest distance to our prediction. This stage involves creating a vector Ψ which contains integral scores for all the test images (size 6 for 2x2 RPM Problems, and size 8 for 3x3 RPM Problems). The test image with minimum distance to $\hat{I}$ is given by equation (4).

$$I = I_{\Psi} = argmin_{\Psi_X} \left| \hat{I} - \Psi \right| \tag{4}$$

IV. Modifications to Simple-RPM

A number of modifications to the Simple-RPM can be implemented especially at the prediction level. These include: combining multiple directions to predict the missing element, using a correction factor to minimize prediction error, utilizing scoring metrics such as mean or maximum of the integral image, and even combining these metrics in prediction.

A. Multiple Directions

It is possible to have the best analogies along the row, in some cases along the column, and rarely along the diagonal. While the vanilla implementation of Simple-RPM takes either of these, it is possible to combine the row and column for prediction. The diagonal has been observed to give poor predictions. Thus, two predictions $\hat{I}_1$ and $\hat{I}_2$ are defined, and equation (4) is evaluated for each prediction, and the best match is selected. Note however that, analogies in some directions might dominate others. A tolerance threshold can also be used to bound false-positives.

B. Correction Factor

The correction factor is based on utilizing more information present in the RPM Problem. Since predictions are governed by a combination of 2 elements only, other elements become redundant, inactive, and underutilized in solving the RPM problem. Instead, a correction factor is implemented to minimize the prediction error by drawing clues from these redundant elements. Since the integral scores of the trio A, B, C are known, we can infer the prediction error e along this direction as the difference between C and $\hat{C}$, and the adjusted prediction $\hat{I}$ becomes.

$$\hat{I} = \hat{I_o} + \alpha \cdot (\hat{C} - C) \tag{5}$$

Where α is a correction weighting between $\{0 - 1\}$, and $\hat{I_o}$ is the apriori estimate of I. This correction factor is direction-sensitive, and can be combined as well.

C. Other Scoring Metrics

While the standard deviation based scoring metric portrays pixel patterns, other metrics can be used such as the mean and maximum of the integral image. Using the maximum provides average performance as it returns a single pixel value that embodies every other pixel in the integral image. The mean on the other hand induces a lot of false-causes since it is a measure central tendency not dispersion. At the same time, multiple metrics can be combined but they must be normalized and debiased.

D. Implementation - Algorithm

The prediction stage highlights that only two of the input elements is needed for each analogical direction. Thus, only two integral scores adjacent to the missing element are needed to implement the Simple-RPM making its implementation very simple - in fact, the basic Simple-RPM requires just 5 lines of code. Factors to be considered in implementation include: removing image noise as this can bias the integral score (integration induces drift as in Figure 3), ensuring the images are normalized - if RPM elements A, B, and C contain the same images, then they should have the same pixel intensities, and spatial distribution. It is also beneficial to normalize pixel values to the $\{0\text{-}1\}$ range. Careful selection of analogical directions is important.

Simple-RPM Algorithm (Pseudocode)

- Load Input Elements
- Predict missing Integral score
- Match for missing Integral score

V. Results

The approach used is in a much more low-level compared to how humans would attempt to solve the RPM problem. The images are transformed to numerical values in the re-representation phase. Thus, a 2D image of 184x184 pixels is encoded in a single numerical value which is difficult to visualize for humans but very easy for computers to work with. Achieving this re-representation is powerful and in antithesis to using image fractals. Once the numerical value is obtained, in similar fashion to human cognition, analogies can be obtained by comparing the values, or finding a parameter α when applied to A yields E and thus when applied to E yields I through diagonal analogy.

A. Results on RPM

The resulting accuracy on RPM Problem-C is Basic = 10/12, Challenge = 6/12, Test = 9/12, Ravens = 8/12. For the RPM Problem-B on the other hand; Basic = 9/12, Challenge = 6/12, Test = 9/12, Ravens = 8/12. This shows how well the algorithm generalizes over the different sets. It took around 4.4s to run locally. The agent implementation can be categorized as an $O(1)$ operation. The performance of the Simple-RPM was very good across different sets. This submission highlights minimalist code required for the Simple-RPM algorithm. A good comparison of results from the RPM problem sets can be found in [REF].

TABLE I
RESULTS OF THE SIMPLE-RPM ON RPM SETS A, B, AND C

Simple-RPM	Basic(/12)	Test(/12)	Challenge(/12)	Ravens(/12)
Problems-B	9	9	6	8
Problems-C	10	9	6	8
–	–	–	–	–

VI. CONCLUSIONS

The Simple-RPM AI Agent has some similarity to how a human would attempt to solve the problem. First, it takes a big picture of the image using the integral images as a human would and represents it with a scoring metric just like a human would attempt to capture the image in memory. Thereafter, the agent uses analogies by looking across the row and/or columns to find the relationship between the RPM elements similar to how a person attempts to solve the problem. However, unlike a human who would try to find transformations and matches, the agent generates the expected answer and checks the test images for a match.

APPENDIX

.

Implementation of the Simple-RPM Agent (Agent.py) can be found in www.github.io/SimpleRPM

ACKNOWLEDGMENT

Acknowledgement and appreciation goes to Alan Turing.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

REFERENCES

[1] @articlekunda2013computational, title=A computational model for solving problems from the Raven's Progressive Matrices intelligence test using iconic visual representations, author=Kunda, Maithilee and McGreggor, Keith and Goel, Ashok K, journal=Cognitive Systems Research, volume=22, pages=47–66, year=2013, publisher=Elsevier

[2] G. O. Young, ÒSynthetic structure of industrial plastics (Book style with paper title and editor),Ó in Plastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 1564.

[3] W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123135.