Amol Parikh
aparikh42@gatech.edu
November 30, 2014

Project 4

Reasoning the agent uses

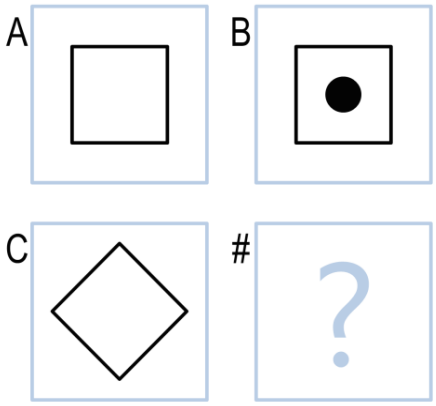
In this project the agent first has to read through all the images first to create knowledge base before it can solve the ravens matrices. To tackle this problem what i have used is opencv to identify the images and its properties. I use contour filling to identify the contours of the images. To do so however, the image is first converted to a greyscale image and then the contours are identified. Based on the number of edges that are identified, we determine what shapes are there in the images. So for examples if 3 edges are identified, we deduce that it is a triangle. Once this is done, we need to get the attributes of each object in each figure. I decided go to with 7 attributes per object. The attributes that i choose for each figure were Shape, Fill, Left-of, Above, Inside, Size and Angle.

Shape of the attribute was identified as explained above. Once the shape was identified, the centroid of each object was calculated using Moments. The centroid was in terms of X.Y co-ordinates. I then found out the pixel value of this centroid to find out if Fill was true or false. If the pixel value is "1" it meant it is false otherwise it is true. For the spatial arrangement of two or more objects, if the x coordinates of the centroid of the objects are the same and the y coordinates are different means that the objects are side-by-side and that one is to the left of the other. In the same way if the y coordinates of the centroid of the objects are same but the x coordinates are different means that one object is above the other. If both the X coordinate and the Y coordinate of two or more objects are the same but their height is different this means that the smaller object is inside the larger object. Thus we calculated Inside and Size together. The angle is calculated using moments and a function called fitellipse. This function calculates the angle of the major axis of the smallest ellipse that can be fit to figure. However this works only for figures with more than 4 edges and so for a triangle and a square we have to use the angle given by the moments.

Once each object has each of these attributes generated they are written to a cvs file for each problem. The Csv file has the figure name along with each attribute of each object along side. Once this csv we created we iterate through all the rows of the given problem to generate all the transition that have occurred. So all the transitions from A-> B , B->C etc are then recorded and written into another csv file. Once all the known transitions are written out, we get the attributes of each object of C(in case of 2x1) or H(in case of 3x3) and compare then to the new csv file. The first thing that we have to compare is the shape. Once the shape of C and left side of any of the transition matches we can say that this particular transition happen earlier and based on these knowledge we can create a new figure. Thus I have used Case Based Reasoning to get to the correct answer. Once we iterate through all the objects in C or H and find out the correct transitions for each of them to generate a figure, we check this figure with all the options provided to get to the correct answer.

How the agent comes to some of its correct answers?

The previous section discussed about how the agent works to select the correct option. It uses Case Based Reasoning to generate an answer which it then test against all options to give the correct answer. I will explain this using an example :



In the figure shown there the agent first has to identify the images. It does so using contour filling. It identifies the Square in A and the Square and the Circle in B.

It then calculates the centroid of all the objects in A and B.

The coordinates of the centroid of the Square and the Circle in B are the same and the Size of the circle is less than that of the square which says that the circle is inside the square. Also the pixel value of the centroid is 0 for the circle which says that the circle is filled.

The angle for the square in both A and B is 0/90.

All these attributes are written into a csv file.

Next another csv file is created which are all the transitions from A to B. So this csv file looks like this :

SHAPE	SQUARE	SQUARE
FILL	NO	NO
LEFT-OF	0	0
ABOVE	0	0
INSIDE	0	0
SIZE	134	134
ANGLE	0	0
SHAPE	NULL	CIRCLE
FILL	NULL	YES
LEFT-OF	0	0
ABOVE	0	0
INSIDE	0	0
SIZE	NULL	44
ANGLE	NULL	0

The csv above shows all the transitions of each object in A to B. Now we iterate through all objects in C and try to match the shape transition from A->B in C->D.

The first object in C is square which matches the first transition. So we copy all the attributes of the 2 column to generate D. So the square in D has the following attributes such as No fill, Not Left-of, Not Above, Not inside. In case of size i have calculated the difference between the size in the objects of A and B. So I add the difference to the object in C. So here I add 0, hence the size of object in D is the same as that in C. Similarly for angle the difference is stored and just need to add the value to the angle in C. Hence the angle of square in D is also 45.

Next there isn't a figure in C which is the same as Null. So a circle would be generate in D with the same attributes as those of the circle in B.

All this is stored in an array and is compared to each of the options. The option closest to the answer generated is then returned as the answer.

Why the agent makes mistakes?

One of the major mistakes that the agent makes is that it is not able to identify between pac-man and a circle. So hence all the problems having both a pac-man and a circle are incorrect. Also is there is an angle difference between different pac-man in the problem, it cannot identify that as well because a circle cannot have any angle. Another mistake that is makes is that even if there is a tiny difference in the size of the generated object and the correct option, the agent does not ignore it just not giving nay option as the correct answer. To counter this problem, i have introduced a metric which penalizes the agent by 0.5 point for a difference of 1 in the size of the generated object and the correct option. However m the major problem that the agent faces is during the image processing phase where it is not able to get the contours of multiple images one inside the other. Even if it does the logic behind identifying an object inside, left-of or above another object is incorrect as the centroid of the a circle or a square inside one another are correct but when it comes to a triangle it doesn't work. A triangle inside circle/square based on my logic comes out to be above the square /circle.

What can be done to improve the agent?

The major part of the improvement can be done in the image processing. In the correct agent, it is a very primitive form of identifying images. Getting the spatial arrangement and the orientation of an object is very difficult and i have resorted to getting the centroid which does not work in each case Instead using a method in which we can get the slope of the base of each object and get the angle from that would be much more helpful. Also in case of spatial arrangement, clustering of some sort to get which side each object is on would be better than the method that i have applied here.

How long does it take to run?

The agent does a lot of image processing initially before it can generate csv files and get the final output. Hence, the time take to read and process the files is a bit longer that reading through text files an generating csv files. Also the agent for every images has to access the image from the main memory and also write csv files in the main memory is a little slower than previous agents. However it does not take unreasonably long times to get the answer.

Relationship between the agent and human cognition.

The agent uses case based reasoning to solve Ravens Matrices. It check whether there has been a previous case where the correct shape and transitioned to something else. If so, then if mimics that transition and generates a new new step by step. It copies all the transitions that were there in the previous similar transition thus generating a new image. This is very similar to how humans also solve the same problem. Humans would look at matrices and see how each shape is transitioning to the next, thus creating a knowledge base of each transition in a way the agent does. It then looks at the final object and checks whether a similar transition has already occurred or not. If it has then humans use that transition to generate the answer exactly how the agent does. However, humans can find patterns in the transitions and if the shape is not in the knowledge base, they can get the answer from the pattern that they identified. This cannot be done by the agent and thus falls short when the shape it encounters is not part of the knowledge base.

Difficulties and differences in addressing these problems visually rather than propositionally

One of the major difficult was to generate a knowledge base of each object in each figure along with all its attributes. When this information was already provided to us as in the previous project, the major effort just went into analyzing and generating the correct answer. However in this case major effort went into get the knowledge base correct after which the analysis and the generation part could be done. Another major hurdle was that there were minor differences in the images in problems such as the size or the angle. These could not be perceived by the human eye but were exposed by the image processing algorithm Due to these minor differences in the figures, the generated answer would be a little different from the actual answer and needed some sort of weight to counter the differences so that we could end up getting the correct option.

This was much harder than previous project solely because of the visual nature of the problems. Once that was tackled , the next part was more or less the same.