Visual-thinking AI Report

Chuilian Kong

System Design

My AI system can be divided into two functioning components: the first component is called visual recognition, in this component, the system will statically analysis the input picture, recognizing the shapes and colors in the picture and then build a mental model of the picture. The second component is called mental imagery, in this component, the system will use the mental model to perform inductive reasoning based on given rules.

Visual Recognition

Visual recognition component takes the original picture as input, and after gathering all information it needs in the picture, it generates a series of mental representations as the output. More specifically, this component includes following 5 modules: color classification, fizzification, finding connected components, shape recognition and attach pivot & string points.

Color classification

The system will take a hand-draw picture as input, which means the color of pixels in the picture can be an arbitrary combination of a three element array [R, G, B]. However, not all of these colors make sense to a human observer, that is, according to the rules, we only have 5 colors can be translated into meaningful representations: black represents an immovable object, blue represents a movable object, yellow represents a pivot point, green represents a string, and white represents empty space. Therefore, the system is supposed to category the color of pixels into 5 basic colors: black, blue, yellow, green and white. To implement it, I used an intuitive approach – category colors to their closest basic colors. The measurement of closeness is the Euclid distance between the RGB vectors of two colors.

Fuzzification

Fuzzification is a process that the system blurs the input picture on purpose. Due to the difference between hand drawing styles, there are always some unexpected drawing flaws in each of input picture such as: small holes in the body of an object, a straight line turns out to be a little bit curved and a round that does not like a circle, etc. To alleviate the impact of these drawing flaws and build a more robust model, the system will blur the picture in a certain rule: the system first separates the picture into small size squares, and for each square, count the number of pixels in different basic colors, and then tag the square according to the most significant color in the square. Additionally, the after fuzzification, the input picture will break down to a smaller size representing format, in this format, my system can perform following processes more efficient. Therefore, in general, we have two major purposes to perform fuzzification: for better recognition and for efficiency.

Finding connected components

The system will maintain a list of objects and their states when constructing a mental model of the picture. In order to implement it, first we need to know which region of pixels can be seen as an object. The most intuitive idea should be, separate pixel regions according to their connectedness, i.e., my system needs to find the connected components in the picture. And of course, the connected components should be in one color respectively, since even if two pixels are connected, as long as they are not in the same color, they should not be considered parts of one object. The method I use to implement finding connected components is rather intuitive but has unexpected excellent performance on most of picture

inputs, may be partly due to the Fuzzification process in which we eliminate some of hand drawing flaws. Specifically, given a color, the system will traverse all entries in the tag matrix of the picture (which is the smaller size representation of picture generated by the process Fuzzification), if the system finds an entry with given color label is not marked, set the entry as a starting point of a new component, from this point, search for all entries next to it that has the same color label, mark all these entries and set them all as start point for searching, until a start point find all its neighboring are either marked or not in the same color.

Shape recognition

In order to construct a more accurate and human-like mental model. I expect my AI system is capable of recognizing different shapes and maintain a list of their shape features. To implement this, I used a function in Matlab image processing toolbox called regionprops. The function will take a logical image of one of the components, and then return a series of properties of that component. Those properties cannot directly category a region to a certain shape, but I can use those properties to help me achieve this purpose. I first define the knowledge space of my system, i.e., all the shapes the system are able to recognize. The first attempt is rather naïve – the system can only recognize filled circles and filled rectangles. After setting up the knowledge space, I get those four values about a region: the center of the region, the maximum axis length, the minimum axis length and the orientation of the region. If the maximum axis length is as more than two times as the minimum axis length, I assume this must be a rectangle (otherwise the hand drawing is supposed to be extremely bad), in this case, I first rotate the region by the angle of the orientation of the region, then use a bounding box to get its width and height. So for a rectangle, I have its center's position, width, height, and orientation as key features to define it. For the case that the maximum axis length is as less than two times as the minimum axis length, we use a filled circle and a filled rectangle to fit the region and choose one that best fit it. And for a circle, I have its center's position and radii (mean of the maximum axis length and the minimum axis length) to determine it. The recognition result is very good for most cases: I tried to recognize components in 8 pictures and it only missed 1 component. However, all that shape information except center is not used in further inductive reasoning. Partly because it relies on Matlab toolbox which is not allowed to use in this AI project and partly because the system is capable of doing inductive reasoning without that information. Though for further improvement, the system can use the shape information to generate a more precise mental imagery and even automatically modify the possible flaws in the original image (which is what I intent to implement but not finished due to time limit).

Attach pivot & string points.

Finally, the system will attach pivot and string information to the components. For almost every existed input pictures, the pivots and strings are always attached to blue objects. Therefore, I only consider the blue object for attaching a pivot point or string point. The method I applied here is simple but effective. I generate a bounding box for every blue object in the picture, and see if there is a yellow or green point within it. For a yellow point within a blue object, the system will find the connected component of the points and then get the center of the yellow component, consider it as the pivot point of the blue object. For a green point within a blue object, the system will mark the blue object as "stringed", and determine the attach point is on which side of the object (such as right, left, up, down), after checking all green point, tells the "stringed" blue object that who else are "stringed". There is an apparent restriction is the system can only model one string. Since most of input pictures have less than two string, this restriction is tolerable. In conclusion, the pivot and string information is stored in the correlating blue object, it knows whether it has a pivot on it, the position of the pivot, and whether it has a string attach to it, the position of the string attach point and which object the other side of string attaches to.

Mental model

After visual recognition procedure, the AI system transfer a hand drawing picture to a list of objects along with their specific properties as the mental model. When doing the inductive reasoning, the AI use the

information provided by the current mental model and with the help of some rules, generate an update state list, then update the states of objects, and do the inductive recursively until one of the object reach the terminal sate. More specifically, the system maintains two list of object, one is the list of blue objects, the other is the list of black objects. The properties along with all these object are: color, object identification, a logic matrix of all pixels of this object, shape information, and state information. The shape information includes the recognized shape of the object, the center position of the object, the width, height, orientation for the rectangle object, and the radii for the circle object. The shape information, as mention before, is gathered but not used (except for center and orientation) in current system, but hopefully can be used for further improvement of the system. The state information includes the pivot point position, the string point position, to which object the string attached, and if the object unstable along with the potential move angle. The state information is only for blue objects, since we assume only blue objects are able to move.

Mental imagery

Mental imagery component uses the built-in mental model and given rules to do inductive reasoning about the state of objects in the mental model, then generate the result as a series of pictures captured the step-by-step movement of the hand drawing physical system.

System framework

The framework of my AI system is simple and intuitive: first, the system will read a hand drawing picture as input and then construct a mental model of all these objects and represent them in lists of properties. Then check all movable objects if they are in an initially unstable state, mark all unstable object as "unstable" and initialize their potential moving angle. Then enter a loop, for each round, the system will check all movable objects that are unstable, and either move or rotate them for one step, then check if there is a collision after the action, if there is, add the corresponding updated state to update state list. After all unstable objects moved for one step, update states according to the list. Take a picture. Then check if there is an object that reaches its terminate state, if there is, terminate the system, else start a another round.

Reasoning rules

For each movable object,

If it has a pivot point and a string point

1. Set its string partner's next state to "unstable", give the partner a move angle.
2. Cancel the string attachment (avoid further complicated interaction)
3. Rotate it for 1 step
4. Detect collision and add update states to update list.

If it has a pivot point but no string point

1. Rotate it for 1 step
2. Detect collision and add update states to update list.

If it has no pivot point but a string point

1. Set its string partner's next state to "unstable", give the partner a move angle.
2. Cancel the string attachment (avoid further complicated interaction)
3. Move it for 1 step
4. Detect collision and add update states to update list.

If it has no pivot point and no string point

1. Move it for 1 step
2. Detect collision and add update states to update list.

When collision occurs,

for instance, A hits B. (which implies A is a movable object)

If B is immovable

1. check B’s orientation
2. A will still be unstable, but the moveAngle change to B's orientation.
3. Updatelist[A unstable moveAngle-> orientation of B]

If B is movable,

if B has a pivot point,

1. set B’s next state to “unstable”
2. give B a rotate direction
3. set A’s next state to “unstable”
4. give A a move or rotate direction (depends on whether A has a pivot)

if B does not have a pivot point,

1. set B’s next state to “unstable”
2. give B a move direction (same as A)
3. set A’s next state to “unstable”
4. give A a move or rotate direction (depends on whether A has a pivot)

Results

I’ve tested my system for picture input number 5, 7, 8, 15, 22, 19(my design) and all physical system are passed except my design. I will here analysis a little trickier problem 15, 22, 19, the running result of the rest of passed physical system can be found in my uploaded file.

Experiment for picture 15

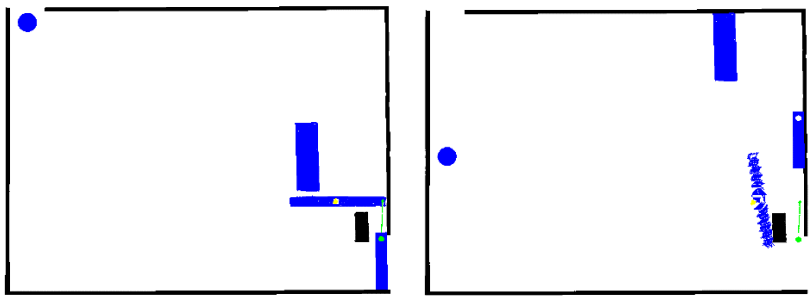

Figure 1. Begin and End picture of pic.15

In this experiment, the blue ball and the blue rectangle all fall down at the beginning, then the rectangle hits the blue bar and bounce up, the blue bar starts to rotate, causing the right blue bar being lifted by string.

Experiment for picture 22

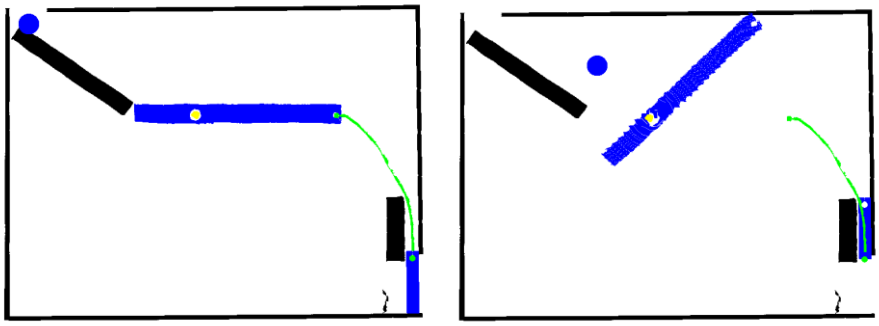

Figure 2. Begin and End picture of pic.22

In this experiment, the blue ball falls down along the black bar, then hit the blue bar and bounce up. The blue bar starts to rotate, causing the right hand blue bar being lifted by string.

Experiment for picture 19

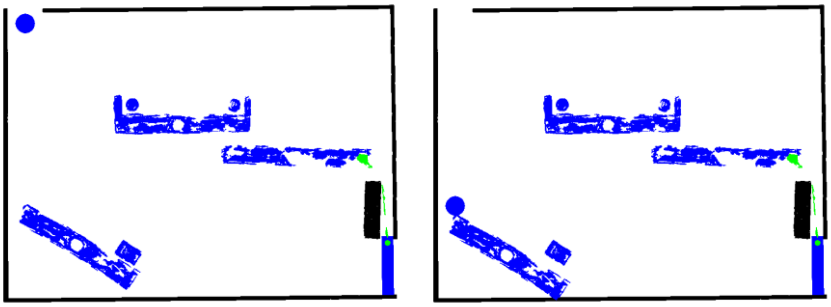

Figure 3. Begin and End picture of pic.19

In this experiment, the blue ball falls down, hit the blue bar, and they fall together. This experiment is not run as expected mainly due to the quality of recognized image is extremely poor. The system though found those blue, black, green connected components, but none of the yellow component could be found. There is no yellow point in the color classified image. To fix it, we may need to consider a more dedicate color classification method.

Conclusion

My AI system is capable of recognizing different components of a hand drawing physical system and with the help of reasonable rules, my system will generate a series of pictures indicate the future events of the system.

Further improvement

Multiple string support

As mentioned above, the current system supports at most one string in a physical system. It's mainly due to the difficulty to recognize which two objects the string is connected. For most input pictures, strings are not connected to its attach point because the blue color is darker and stronger than the green, so when we draw green string through a blue object, part of the string will disappear. In this case, the system cannot use the connected component to determine which two objects are connected by a string. This problem could be solved by more dedicated color classification and as long as the system is able to recognize a complete string, the system could easily add multiple string connection information to object's state vector, and the inductive reasoning rules will remain the same, and the problem will be fixed.

Complex string interaction

For simplicity, I deducted the string interaction between two objects connected by a string. What I assume is the string is very weak so that any force on it will break it, i.e., the inner force of string only take effect once, at the moment that one of the object start moving. This simplicity won't affect the results in most cases in the picture's set but if the system could emulate the real-world string behavior, the running process will be more natural.

Automatic modification

As mentioned above, I originally gather the shape information in order to use regular shapes to correct the flaws in original hand drawing picture. Such as the circles are not round or the rectangles are not straight. In the regularized mental model, every movable objects and most of immovable objects has their unique shape and position information, instead of storing the whole logical matrix of the object, the system only needs to store the shape and position information to determine an object. And also, the inductive reasoning process would be faster and more precise.