Imagery-based AI for Solving Basic Topological Problems

Kunal Nabar (kunal.p.nabar@vanderbilt.edu)
Department of Computer Science
Nashville, TN 37240 USA

Abstract

Topological reasoning is a common skill for human beings in the western world. How human beings are able to process topological/geometric images and come to conclusions isn't fully known. Do we have inherent notions of geometrical objects and set cognitive processes when it comes to comparing them? This paper presents an AI system that uses a visual-imagery based approach to solve the problem of identifying discrepancies in topological systems.

Keywords: topology, geometry, AI system, difference image identification

Topological Problems

This paper aims to solve the problem of identifying unique topological images via an AI system.

The inspiration for the problem presented in this paper comes from the Dehaene (2006). A study was conducted on the Mundurukú, an Amazonian indigene group. The participants in the group were children and young adults. This group has had very limited contact with the outside world. Furthermore, people from this population do not receive any formal arithmetic or geometric education. They serve as a reasonable group to determine a human's innate skill to find unique geometric figures.

The test was conducted by giving the participant a set of six visually similar images that illustrated a particular geometric concept. For example, the initial training set outlining color and orientation is shown in Figure 1:

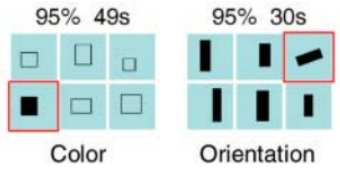

Figure 1: Initial training set used to outline problem

As shown above, the two "unique" images, outlined in red, are very easy for a human being to spot. The Mundurukú performed with high accuracy on both training sets.

However, the participants were also tested on four topological concepts: inside, closure, connectedness, and holes.

Problem Types

The four problems (closure, connectedness, and holes, inside) tested the skill of the Mundurukú children at understanding these topological concepts. Each of these topological problems is solvable by the AI system.

Closure the concept of having a fully closed curve. Identifying the unique image requires recognizing the curve that is not closed.

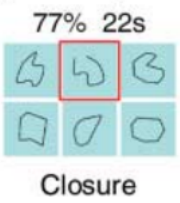

Figure 2: A representation of the closure problem

Connectedness the concept of having a fully connected object that is made up of a uniform continuous color. Identifying the unique image requires finding the image with an object that has multiple components.

Figure 3: A representation of the connectedness problem

Holes the concept of having an object that contains gaps or holes within the outermost boundary of the object. Identifying the unique image requires finding the image with any point in the outer boundary of the object that is of a different color.

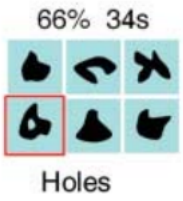

Figure 4: A representation of the holes problem

Inside the concept of having an object contained within a curve. Identifying the unique image requires recognizing the image that contains an object that falls outside of the curve.

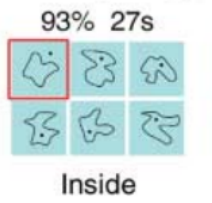

Figure 5: A representation of the inside problem

Each of these problems are simple for the human observer. Mundurukú participants solved these problems at the following rates:

Table 1: Mundurukú participants test results

Problem Type	Correct %	Avg. time
Inside	93%	23s
Closure	77%	22s
Connectedness	68%	23s
Holes	66%	34s

These findings are far above a rate of chance (16.6%). As found in the paper, this implies that humans likely have topological recognition capabilities that are not a result of cultural or social norms, they are innate. And replicating an AI system to work on the problem via an imagery-based approach could be very worthwhile.

Note: most of the figures and findings written in this section are from Dehaene et al. But serve as a very good basis for the understanding of the approach to the AI system.

How Humans Approach the Problem

When speaking with people who attempted to solve this problem, they tended to use abstract words such as "inside" "outside" and "contained." These words are understood by humans in context, but they are incredibly complex from a computational standpoint. The system obtains notions of closure and containment by finding the unions of different image regions. It is possible that human's use a similar method to solve this problem. But tend to just "see it" rather than employ the methods consciously.

System Overview

Inspiration and Background

This system was designed with inspiration loosely taken from Kunda (2013). A system was created in order to solve Raven's Progressive Matrices intelligence test problems using visual representations. These visual representations were rotated and manipulated, and then compared using a visual similarity metric in order to determine a best case result. A comparable approach was taken here to solve the problem and obtain a viable solution.

Approach to Solution

Each of the "objects" in the image take up certain amount of space in terms of pixels in the image. The rest of the image is taken up by white space. The union of all of the pixels containing the objects and all of the pixels containing the white space will yield all of the pixels in the image. The general approach to solving the problem is to pick a starting pixel of both black and white color, referred to as a *seed*. Begin growing from the seed for each image until two different image masks appear, black and white respectively. The union of these two pixel sets will yield a unique number for the solution image compared to the rest of the image set.

Closure Growing the white region of the closure will either result in getting caught inside the closure or outside the closure. The black region will grow along the curve of the enclosure. The area of the entire image will never be reached, unless there is not a full closure, making one continuous white region instead of two.

The first image in the figure is the black mask, the second is the white mask, the final image is the union of the two.

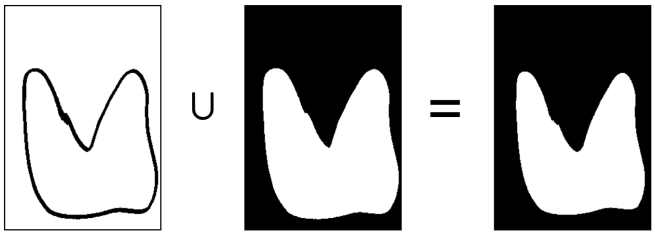

Figure 6(a): An incorrect closure answer union

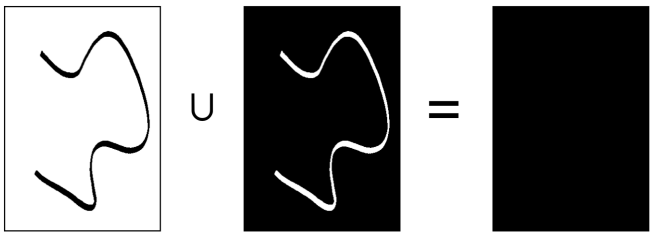

Figure 6(b): A correct closure answer union

The union in 6(b) yields the entire pixel set, this will be the only image to do so and is therefore the correct answer.

Connectedness Similarly, you can grow both the white and black regions for the connected components. In this case, all image sets will yield a full union except the desired answer.

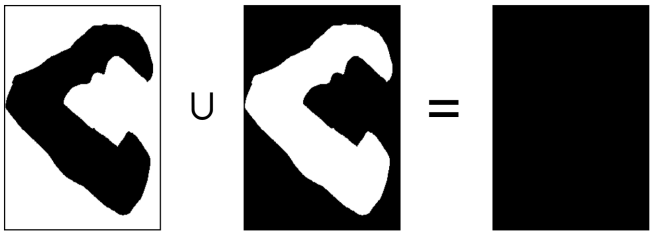

Figure 7(a): An incorrect connectedness answer union

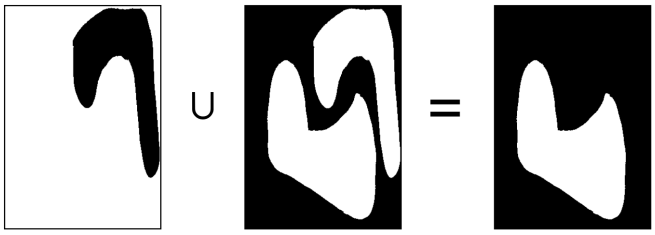

Figure 7(b): A correct connectedness answer union

The union in 7(b) fails to yield the entire pixel set. This will be the only image to do so and is therefore the correct answer.

Holes A method analogous to the connectedness solution is used to solve the holes problem. A depiction is shown in Figure 8(a,b).

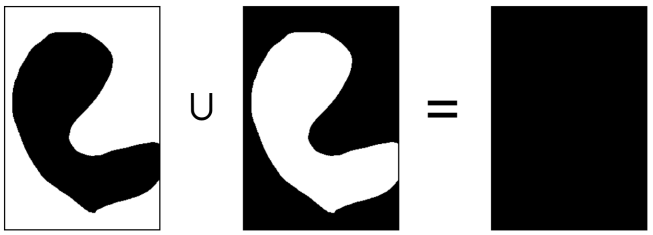

Figure 8(a): an incorrect holes answer union

Figure 8(b): a correct holes answer union.

Once again, the union in 8(b) fails to yield the entire pixel set. This will be the only image to do so and is therefore the correct answer.

Inside The most complex solution is the inside closure. The initial algorithm fails to yield the entire pixel set, and therefore, no correct answer. However, in the case where no black/white union in the image set yields the entire pixel set. Every contour in the image is filled, changing the inside problem into a connectedness problem, and the image set is processed again.

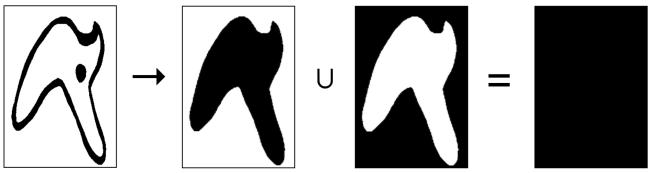

Figure 9(a): the incorrect inside answer union

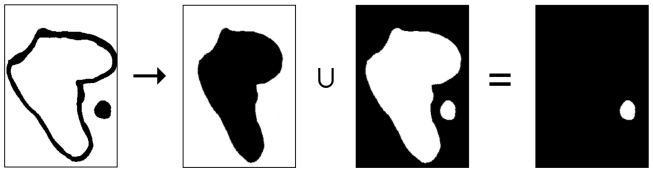

Figure 9(b): the correct inside answer union

Like the connectedness problem, the union that fails to yield the entire pixel set is the correct result.
For each of the image sets, either only one image will yield the full pixel set (closure) or all but one will yield the entire pixel set (connectedness, holes, inside). It is

trivial to find the different image after these metrics have been calculated.
Each of the figures in this section are outputs of the "mental image" that is stored within the AI system, not the raw image itself.

System Implementation

General Information

The system was implemented in Python, the AI only takes in a set of images and attempts to apply topological techniques to find the correct answer. The images were processed using the OpenCV library, some OpenCV functions were also used to shrink and fill image masks in order to obtain necessary information that the AI system uses to make decisions.

Inputs and Outputs

The only parameter to the *main* Python file is the path to the directory containing all of the desired images. The input to the system is all of the images read in through OpenCV and stored in a Numpy array. However, no OpenCV operations were performed at this time in the image pipeline. Although the images shown above have blue backgrounds, images with black figures and white backgrounds were used in order to maximize contrast. Images were both hand drawn and computer drawn. The system will output the name of the different image on the console and then amount of time it takes to execute. The system can also take in options to save the output image or display the image after the system completes.

Image Processing

The images are passed as 2-D Numpy arrays to the system. The system then begins the image processing. First, the entire set is cleaned. Cleaning requires shrinking each image to a preset ideal height and width. This "ideal" size is the size used for the "mental image" that is stored in the AI system as it makes decisions (Height=250,Width=300). These numbers were reached after attempts to find the ideal numbers that will not distort the image as to achieve an incorrect result but are small enough that algorithms that execute later in the pipeline achieve acceptable times. Otsu's method[1] is applied in order to determine the best binary threshold for the image. This is used instead of a simple closest-value approximation in order to deal with noisy or non-computer-generated images.

[1] https://en.wikipedia.org/wiki/Otsu's_method

Seeding, Region Growth, and Finding the Unique Element

Seed Selection For each mental image, one black and one white pixel are randomly selected from anywhere in the image. The other potential approach to selecting the seed would be to pick the first instance of a white pixel in the raw image and the first instance of a black pixel in the raw image. This approach is purposely not chosen in order to avoid starting from the upper left hand corner of the image since it is more computationally inefficient to perform region growth from that a random spot on the image than that corner.

Region Growth Each seed was then passed into a function that performed a "flood fill" using OpenCV. This function would attempt to fill pixels adjacent to itself, starting at the seed, that had the same color. A binary mask of the filled pixels is created for both the black and white seeds. The union of these two masks will yield the pixel set that is used for the AI's decision making.

Finding the Unique Element For each mental image, one black and one white pixel are randomly selected from anywhere in the image. The number of pixels found in the union are divided by the total number of pixels in the image for each one. Either all but one of the resulting values is equal to 1.0, or only one is equal to 1.0. The image corresponding to the odd element out is selected as the final answer. If none of the images yield a resultant value equal to 1, the system will fill contours of the original images and execute again (in the case of the "inside" problem).

Additional Information about External Libraries

OpenCV Used for image I/O, performing basic transformations (resizing proportionally, binary threshold calculation, contour filling) and the flood fill operation after the seed value was obtained
Numpy Used to store the image and pixel sets over the life cycle of the program and some basic indexing operations.
Matplotlib Used to output the image at the end of the execution if the "show" option is selected
Random, sys, os, timer Built-in Python Libraries that are used for miscellaneous simple operations.

Results

General Performance

The system seemed to perform very well in terms of correct answers on each data set that was created,

achieving the right answer every time. The time taken to achieve the answer varied for each data set. It seemed that the system performed equally well on both hand drawn and computer drawn images. The system takes ~2x the amount of time to run on the "inside" topological problem. This is because the system must run twice on these problems. After failing to find an answer the first time, the contours must be filled for each element, and then entire operation must be executed again.

Hand-drawn vs. Computer-drawn

There was no performance boost for using computer-drawn images instead of hand-drawn images. However, the system performs the same operations on the image regardless of the type of input. It is likely that the computer-drawn image would produce the correct answer without any image processing more often than the hand-drawn. However, the state of the object prior to processing is not checked. This is intentional, parsing the image and performing the same operations on it regardless of small discrepancies on a pixel by pixel level seems to keep fidelity to the processes used by human beings to process mental images through the eye. The performance on different input mediums is shown below in Table 2(a) and 2(b) for hand-drawn and computer drawn data, respectively.

Problem Type	Hand	Computer
Closure	0.383s	0.275s
Connectedness	0.337s	0.366s
Holes	0.369s	0.357s
Inside	0.623s	0.707s

Table 2: Table outlining the AI system execution time for each type of problem

The data obtained from this trial was run on the same data set, but one was hand drawn and scanned and the other was drawn in an application similar to MS Paint.

Errors and Future Possibilities

Known Errors

As of now, there are no specific data sets that can cause the system to fail in the domain of the problem as described. However, the specific solution and approach that is taken to the problem does prevent the AI from expanding to more tasks in this domain. If the system were to process an image that solved the a topological problem that necessitated knowing properties of the object (such as three squares and a triangle, with the triangle being the odd element out), this process would fail.

Future Possibilities

It would be ideal to expand the abilities of the AI system. Creating a system that can also solve the other problems defined in Dehaene et al. would be far more useful. The task chosen for this system was not trivial, but solving every core geometry trial given to the Mundurukú participants would be more challenging.
It would also be ideal to configure the system to take in one continuous image that has 6 different areas that correspond to the image as shown in Figure 10.

Figure 10: one image containing all of the trials

When tested on humans, they were able to solve the connectedness trial in Figure 10 on one regular piece of paper while viewing all of the images. Image contouring is possible to find the component images and cutting each image by contours by the computer. However, in some cases (such as Figure 10). The correct answer will be split into two different contours. When a single object contains multiple components, the AI system does not have a way to tell which object set the components belong to. Human beings generally have some cognitive methodology to solving this problem. Learning more about this method and incorporating it into the AI system could improve the process moving forward.

References

Dehaene, S. (2006). Core Knowledge of Geometry in an Amazonian Indigene Group. *Science Vol. 311* (pg. 381-384). Washington, DC, USA: American Association for the Advancement of Science.

Kunda, M. (2013). A computational model for solving problems from the Raven's Progressive Matrices intelligence test using iconic visual representations. *Cognitive Systems Research Vol 22-23* (pg. 47-66). Amsterdam, Netherlands: Elsevier Science Publishers.

Chaganti, Shikha. (2016) Personal communication.