A COMPUTER PLAYER FOR LINK GAME

Luxin Yang
Department of Computer Science, University of California, Berkeley
isla.yang@berkeley.edu

ABSTRACT
In this project, we follow the rule of playing link game in software program and make the computer to play with a given link game image. By implementing image rectification, block segmentation and feature matching, the program can finish any input link game.

1. BACKGROUND
A link game is that the player is offered an interface with m by n sub-blocks with anime or any featured pattern, here I took the example with pokemon. Giving such a graph, we first need to find a pair of sub-blocks with same pokemon, and then detect whether they can be connected. If yes, then click them and they will be deleted from the display.

There are three allowed connecting methods for two sub-blocks. They can be positioned on the same horizontal line or vertical line without any sub-block in between; Or turning points maximum to 2 can be used for connecting. The players' goal is to clear up all the pokemons to finish the game.

2. OVERVIEW
The goal of this project is to let a link game robot capture the surface of computer screen and control the mouse to play. Thus we need an eye to capture the input image, a robot brain to find matching and connectable pair, and a hand to select and click using mouse. The main software part here is the brain, and it can be done with three steps: block segmentation (although human brain do not need bother to pay extra attention on this step, computer can not ignore it), find matching pokemon, and detect connectability.

3. METHOD
Block segmentation is to segment the input captured image into m by n sub-block images, save them in m*n individual matrixes and treat them as individual pokemons. However, the image captured by robot eye can not always keep upright. For the most time it would be tilted or even blurred as is shown in Fig. 3, which require a pre-processing for the input image.

![Figure 1. Sample of the interface of a link game.](image)

![Figure 2. A link game robot.](image)

3.1. Image rectification
To reduce the error caused by image capturing distortion in segmentation. The input image is rectified using the Harris Interest Point Detector and Adaptive Non-Maximal Suppression introduced by Brown et al. [1]. After finding the four rectangular corners of the captured image, it is easy to do calculate the homography and warp it into a upright
3.2. Block segmentation using Fourier fitting

Here I assume that the background is simple and with little information, so I make use of the periodic boundary information around the pokemons and use Fourier fitting to approximate the size of the sub-block.

I first take the mean value of all horizontal grayscale pixel value and summing up all of them, the periodicity is quite obvious as in Fig. 3.4. By doing Fourier transformation, the main frequency (the peak) has high possibility to represent the repeat of block boundary. Thus we fit it with this single frequency Fourier wave (Fig. 3.5). Same to the vertical lines. The wave length is the block size and the initial phase tell us the position of the first block on the background.

3.3. Block matching

In order to define the similarity between each two sub-blocks, here I use Normalized cross-correlation (NCC) and use the peak value to evaluate the similarity. Then rank in descending order in an array and search from the pair with highest similarity of their connectability.
3.4. Detect connectability

When two sub-blocks are connectable, there are three allowed connecting methods. They can be positioned on the same horizontal line or vertical line without any sub-block in between; turning points maximum to 2 can also be used for connecting. In algorithm, I divide the total three types into two types. One is connecting with not turning point or one turning, the other is connecting with two turning points.

In software, I draw two horizontal and vertical lines from each point (remember that line must stop when hit an existing pokemon). When the two crosses have intersect, it can be detect as connecting with type 1. When straight line connection exist between the two crosses, the pair of pokemon is defined connectable in type 2.

3.5. Delete Block Pair

After all the above process, we could find matching pokemon blocks and define whether they can be linked. Deleting two blocks are not simply set the area to all zeroes on the output image and this is where most of the difficulties took place. When we find two matching pokemon while searching from the top in the array, yet they can not be connect, I keep them in its original ranking in the array. On the other hand, a connectable pair must also be cleaned from the array so as during the next search, we won't encounter mistake and speed up the whole program.

In order to visually show the result, I generated a video and display each frame for 0.6 second. Every time a pair of blocks is detected the same and connectable, a black frame is to help the viewer notice and then it will be deleted from the screen.

5. CONCLUSION

In this project, I implemented a block segmentation method using Fourier fitting to divide a certain input image into an array of sub matrices and value the similarity of them afterwards. Finally, the algorithm include detection of connectability to determine whether a certain pair can be deleted. Above all, the whole process substitute human brain for completing a link game.

6. REFERENCES