Obtaining Path Data From Maze Image Using Image Processing Techniques

Conference Paper · November 2016

CITATIONS
0

READS
989

2 authors:

Ozan Aki
Trakya University
13 PUBLICATIONS **20** CITATIONS
SEE PROFILE

Aydın Güllü
Trakya University
12 PUBLICATIONS **20** CITATIONS
SEE PROFILE

UNITECH 2016 GABROVO

INTERNATIONAL SCIENTIFIC CONFERENCE
18 – 19 November 2016, GABROVO

OBTAINING PATH DATA FROM MAZE IMAGE USING IMAGE PROCESSING TECHNIQUES

Ozan AKI
Trakya University

Aydın GÜLLÜ
Trakya University

Abstract

Aim of this study is obtain calculable maze path data from a maze image or taken maze photo. Image of maze can be a viewable natural maze or previously produced or pressed maze. After digitalization of maze image, image processing techniques are using to analyzing maze walls or lines. Bird's eye view correction can be applied if photo is not taken from bird's eye view. Adjustable threshold value is using to convert image to monochrome. Hough lines algorithm is used to detect maze walls or lines. Detected lines are clustering, concatenating and splitting from intersection points in order. Thus, uniform line segments can be obtained. Finally, these line segments on the maze walls or lines are converting to integer data and saving into two dimensional matrices.

Keywords: maze, maze image, path data, wall data, line data, hough lines, image processing.

INTRODUCTION

Mazes are not just funny puzzles for fun always. Researchers are using mazes for development of self-navigated mobile robots. Also other science fields using mazes for researching animal behaviors.

Almost all of these mazes are created by computer algorithms like depth-first search, prim's and recursive algorithms while some of ones created by hands [1, 2]. Also there may be natural mazes like caves, valleys and canyons.

For any reason, sometimes we need to obtain calculable path data from a previously produced maze. Image of maze can be a computer produced content or a photo of a physically maze.

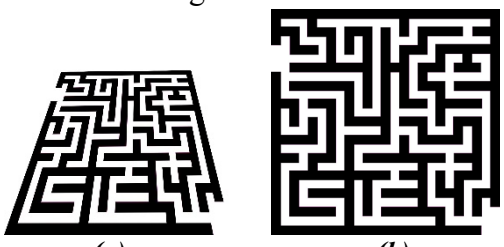
In this study, we have tried out reversing the maze production steps. We aimed to obtain all paths data from an image or photo of a maze.

For achieving this aim, we have used image processing techniques for analyzing maze image. Maze type can be walled or line.

First of all, bird's eye view correction can be applied to maze image. Because taken photos from out of bird's eye view makes images trapezoid. After applying some filters to image for reducing noise and smoothing image then maze image is converting to binary image using adjustable threshold value. Hough Lines algorithm is using for detecting horizontal and vertical maze walls or lines. Detected lines are clustering for same wall and line. Lines on the same alignment are concatenating along maze width or height. Then lines are splitting from intersecting points for determining cell borders. Thus, symmetrical line parts are obtained. Then lines just on black surface are removed. Finally, remained lines data are saved to the maze matrix and produced path data matrix shown to user in a text file for verification.

METHODS

Taken photos of things on horizontal plane from an angle out of bird's eye view makes images trapezoid. Bird's eye correction is using to transform image to correct camera angle effects on image.



(a) ***(b)***

Fig. 1. *Bird's eye view correction a) trapezoid image b) corrected bird's eye view*

Bird's eye view images have advantages for image processing. All lines in image like maze walls and lines are arranged by only horizontal or vertical direction. Thus, computational savings can be possible.

Color information in images are saving into three separate channel for three primary color named as R, G, B. If color information is not needed for image processing, mostly these images are converting to gray-scales. Thus, processing images are simplifying due to using only one channel for calculations. Equation 1 shows conversion of each pixel color data to gray level [3-5].

$$y = 0.2126R + 0.7152G + 0.0722B \quad (1)$$

Coefficients in equation 1 are represent intensity perception by human. Receptors in human eye has different sensitivity to the primary colors.

Like all electrical signals, images also have noise from environment and produced by image sensor circuit. For eliminating these unavoidable and unpredictable noises, smoothing filter can be used over image data.

Gauss smoothing is one of blurring filter. Gauss filter has selectable 3x3, 5x5, 7x7 kernels. Applying one of these kernels smooth pixel transitions and reduce noise [3, 4]. Equation 2 show a typical 3x3 Gaussian kernel

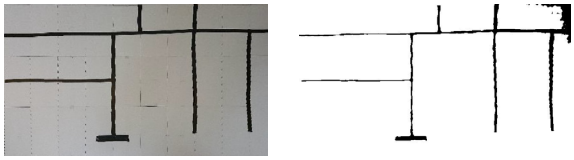
$$p = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (2)$$

Image pyramids also can be used for additional smoothing and reducing noises. There are two kind of image pyramids. Gaussian pyramids are using for down sampling images. Laplacian pyramids are using for up sample images. Thus, down sampling images for desired step and then up sampling same steps again will smooth image reduce noises [3, 5].

Thresholding is most used method in image processing. A threshold value is compared with each pixel gray scale value in image. Thus, resulting image has only two gray level: black or white. This is simplifying the process of image due to regarding color information stored in separate channels [3-5]. Threshold formula shown in equation 3.

$$q(x,y) = \begin{cases} p(x,y) \geq T, 1 \\ p(x,y) < T, 0 \end{cases} \quad (3)$$

Figure 2 shows an maze photograph image before and after converting to binary image using thresholding.



(a) (b)

Fig. 2. *Before (a) and after (b) Thresholding a maze photograph image*

Erosion and dilation are binary image algorithms. Erosion is using for eliminate burrs around borders. Just the opposite, dilation is using to fill small gaps around borders. Using both of these algorithms can reproduce smooth borders in image [3, 5]. Figure 3 and 4 shows erosion and dilation operations respectively.

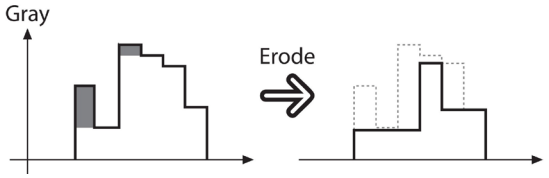


Fig. 3. *Erosion operation on pixel values*

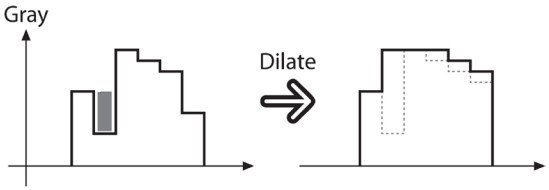


Fig. 4. *Dilation operations on pixel values*

Hough line transform is used for determining lines in binary image. This algorithm assumes that any point in binary image could be a part of any line [3]. Lines are defined as polar coordinates as in equation 4.

$$p = x \cdot \cos\theta + y \cdot \cos\theta \quad (4)$$

Hough transform algorithm testing each point in accumulator plane and calculate local maximum. Then calculating intersection of each tested point and results line parameter. In this study, only $\pi/2$ radian (90°) angle resolution is used for detect only horizontal and vertical lines [3, 5].

Hough line transform can produce many lines for thick lines in image. For elimination

line thickness variation in image, horizontal and vertical lines have been clustered.

Clustering lines have been done by summing all y components of points of all horizontal lines and summing all x components of points of all vertical lines. Equation 5 and 6 shows sums of points.

$$h(y) = \sum(P1_y + P2_y) \quad (5)$$

$$v(x) = \sum(P1_x + P2_x) \quad (6)$$

Calculating each local maximum of both *h(y)* and *v(x)* gives ordered walls or lines positions of maze Thus, each local maximums are using for separation point of line matrix.

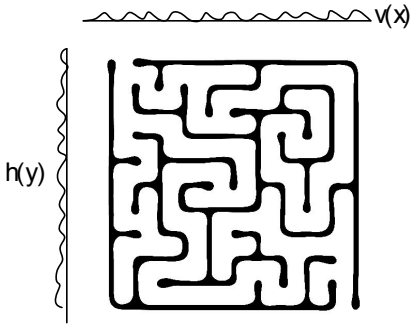


Fig. 5. *Distribution of horizontal and vertical lines*

Obtaining horizontal and vertical alignment lines used for separate maze to cells. Each cell defined as a rectangular form. Borders of these rectangular cells represent walls or lines related to type of maze.

For building maze data, each cell's border line are determined by examining underlying binary image. Border lines lying on the black ground are marked as no wall or no line. Border lines lying on non-black ground are marked as wall or line.

Figure 6 shows that a maze splitting rectangular cells for obtaining maze data.

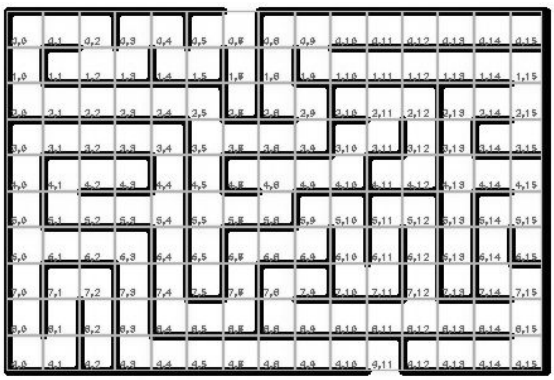


Fig. 6. *Splitting maze to cells*

Each cell borders data is representing by a bit mapped integer number. Lower four bit of the number represent walls or lines around cell. After each cell's value then maze data is assembled into two-dimension matrix. Cell value calculated as equation 7. Acronyms *t*, *l*, *r*, *b* letters are corresponding to *top*, *left*, *right*, *bottom* borders of cell.

$$v = t \cdot 1 + l \cdot 2 + r \cdot 4 + b \cdot 8 \quad (7)$$

An example of conversion result values for maze shown in figure 6 are shown in table 1.

Table. 1. *Representation of maze cells data*

3	9	5	3	5	3	4	3	1	9	9	9	9	9	13
6	11	8	12	10	12	6	6	10	9	9	9	9	9	5
10	9	9	9	9	5	14	10	5	11	1	5	3	9	4
3	9	9	9	5	2	9	9	12	3	12	6	6	11	12
6	3	9	13	6	6	11	1	9	12	11	12	2	9	5
6	10	9	9	4	2	9	12	3	5	3	5	6	3	4
2	9	9	5	6	6	3	9	12	6	6	6	6	6	14
6	3	5	6	6	14	6	3	13	10	12	6	14	10	5
6	6	6	6	10	9	12	10	9	9	9	8	9	9	4
10	12	14	10	9	9	9	9	9	9	5	11	9	9	12

Digitized maze data can be used for variously purposes like obtain maze metrics, reproducing or modifying maze, solving the maze etc.

RESULTS AND CONCLUSION

An application has been developed to achieve aim of this study. For ease of interaction by user, Microsoft Visual Studio, C# Windows forms application has been preferred. Emgu wrapper for OpenCV libraries has been used for image processing [3, 6].

Figure 7 shows the developed application interface. Maze image on the left side is original maze image. On the right side, processed maze image has been shown.

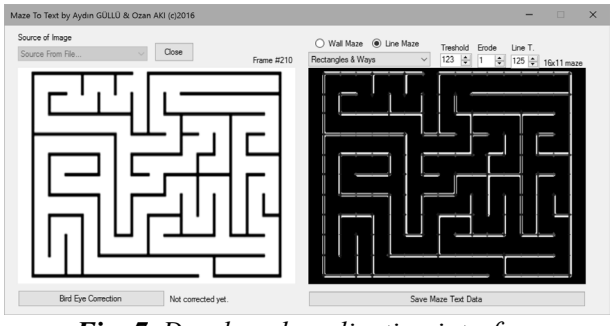


Fig. 7. *Developed application interface*

User can adjust the image threshold value, erosion and dilation iteration numbers. By combining these parameters, best fitting values can be adjusted if image contain suitable maze lines.

Variously maze images has been tested and converted to the calculable data successfully. But if any maze image contains lines out of horizontal or vertical orientation, application will be failed to detect these lines.

Note that, success of application is tightly depended on user parametric tuning experience. In this study, we effort to maximize accuracy as possible.

User adjustable parameters like threshold makes this application to user experience depended. But there may be an option for fully automatic and adaptive work without disturbing user. This option may consider for future works.

Application is designed for only orthogonal wall or line mazes. Due to this restriction, maze data will not have obtained properly if maze have lines out of orthogonal. For the future works, algorithm can be designed direction independent for more complex mazes. Even more arc walls or lines can be detected for circular mazes.

REFERENCE

[1] Kozlova, A., J.A. Brown, and E. Reading. ***Examination of representational expression in maze generation algorithms***. in 2015 IEEE Conference on Computational Intelligence and Games (CIG). 2015. IEEE.

[2] Xu, J. and C.S. Kaplan, ***Vortex maze construction***. Journal of Mathematics and the Arts, 2007. 1(1): p. 7-20.

[3] Bradski, G. and A. Kaehler, ***Learning OpenCV: Computer vision with the OpenCV library***. 2008: " O'Reilly Media, Inc.".

[4] Akı, O., ***Elektriksel Veri Toplama Ve Kontrol Sistemlerinde Makine Görüşü Ve Uygulamaları***, in Marmara Üniversitesi Fen Bİlimleri Enstütüsü. 2004, Marmara Üniversitesi: İstanbul.

[5] Intel, ***Open Source Computer Vision Library Reference Manual***. 2000, Intel.

[6] Schildt, H., ***C# 4.0: The complete reference***. 2010: Tata McGraw-Hill Education.