Augmented Reality Tic-Tac-Toe

Nikesh Muthukrishnan
Department of Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada
nikesh.muthukrishnan@mail.mcgill.ca

Ruiyin Wen
Department of Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada
ruiyin.wen@mail.mcgill.ca

***Abstract*—This project implements an augmented reality version of Tic-Tac-Toe. In this game, the user draws either 'X' or 'O' on a white surface. Using a webcam and digital image processing techniques, including spatial filtering and morphological processing, our algorithm automatically detects the background and filters it out. It further determines the borders of the surface, as well as the location of the user input. Our algorithm registers the position of the users' drawing, and displays the move in the appropriate places on the screen. In order to test the robustness of our algorithm, we test-drive it using artificially imposed noisy and distorted images. Our finalized algorithm registers user's drawing on white surface in near-optimal setup successfully, and performs robustly for artificially noisy images.**

I. Introduction

Augmented reality (AR) integrates the real environment and the virtual objects. In order to fully achieve the integration, an AR-enabled program needs to read the input from the real world, process it into machine-understandable data, and visualize the output. Several popular mobile games, including Pokémon Go[1] and Ingress[2], have partially enabled AR by superimposing data onto the images captured from the real world. On the other hand, researchers have implemented fully integrated AR games that both read the signal from the real world and respond with computer-generated objects.

In this project, we propose a solution for implementing Tic-Tac-Toe, a well-known mini game, in a full AR setting. We tested our algorithm in a well-lit room with the game drawn on a white board with black markers, and received successful results. Parameters are determined through the experiments conducted in the relatively optimal setup.

Although we conducted the experiment on an optimal surface, we implemented several statistical and morphological noises on the captured image to test-drive our parameters. By applying different filtering techniques, we are able to minimize the noise that impact our algorithm's ability to recognize shapes and register input locations. We are able to get satisfying results with reasonable amount of noise after image processing.

To aid in future replication studies, the source code of ARTTT and the experimental data is available online.[3]

[1]http://www.pokemongo.com/
[2]https://www.ingress.com/
[3]https://github.com/rayrwen/tictactoe

Paper organization. The remainder of the paper is organized as follows. Section II provides more detail about the background of augmented reality surveys related work. Section III describes the design of our experiment and details of the proposed algorithm. Section IV presents the results obtained. Finally, Section V draws conclusions.

II. Background and Related Work

Augmented reality (AR) is a technology where the real environment and the virtual objects are integrated in real time. AR is widely used in various fields, including medical visualization, maintenance and repair of complex equipment [1], as well as mobile applications geared towards entertainment [2]. Although AR technology has been developed since the last 20 years, it is no trivial task to process real-world images and fully understand human input accompanied with noise such as wiggles and shadows [3]. Although the scope of this project remains confined with the game of Tic-Tac-Toe, our project contains various techniques employed by previous researchers to process the human inputs.

In order to make the capture images accurate in mission-critical systems, researchers have dealt with various ways to correct and calibrate the images collected in AR applications. Janin *et al.* [4] developed technology to calibrate the head-mounted sensors with the display's virtual screen and users' eyes in a single coordinate system. Lepetit *et al.* [5] eliminates image drift and jitter by using a rough CAD model of the past real scene in real time. We focus on applying spatial, frequency and morphological filters for extracting information from noisy images, as well as visually and empirically verify the results from the denoised information through the agent of a virtual Tic-Tac-Toe game board.

Since the concept of AR has first been introduced as early as 1990s [6, 7], AR games have been developed in various forms. Tic-Tac-Toe has been implemented several times by different research groups [8, 9]. In this project, we replicate and improve the image processing procedures done by the previous researchers. Because of the time limit, we are not able to optimize the algorithm and refactor it into a more efficient environment, as mentioned by Maguire and Saltzman. However, our algorithm is robust enough to register users' input in either X or O, as well as operate on surfaces that have shadows and reflections of other objects.

III. Proposed Solution

Our proposed solution is to use a webcam (AUSDOM AW615 1080p), to read the board and detect the user's input and location. Based off the interpretation, the computer will return a counter move. Several assumption were made in regards to the algorithm and are the following:

- The paper is white and the Tic-Tac-Toe board is positioned approximately in the center of the image.
- The user is consistent with their piece choice (X or O)
- The user's drawing of the piece is well behaved (No strange X or unopen O)
- The user's input is written in black.

Our algorithm can be divided into four major functions. Section III-A describes the *GetPaper* function that focuses the image on the paper and the Tic-Tac-Toe board. Section III-B describes the *GetBoardDimension* function that extracts the center of the board as well as the required thresholds to determine the location of the user's move. Section III-C explains how the *CaptureMove* algorithm determines the location and piece of the user. Finally Section III-D describes the *ReturnComputerMove* function that decides the computer's counter move. Several thresholds and values have been predetermined for optimal processing through trials. These values are listed in Table I. All morphological opertaions used a square structuring element.

TABLE I
HYPER PARAMETERS AND THEIR VALUES

Hyper parameters	values
Paper Threshold	0.65
Black Threshold	0.67
Re-scale Coefficient	1.5
Difference Threshold	0.1
Erosion Size	2
Dilation Size	6

A. GetPaper

Figure 1 illustrates a typical webcam captured image of a Tic-Tac-Toe board. The goal of this function is to remove the background and acquire a focused image of the Tic-Tac-Toe board. The image is first converted to grayscale and smoothed to filter out any background light pixels that may be wrongly accounted as the paper. The image is then binarized using the Paper Threshold indicated in Table I. This provided a mask for the paper. Next, the dimensions of the mask and it's centroid is captured using MATLAB's *regionprops* function. Assuming that the TicTacToe board is fairly centered on the paper, the dimensions of the masks are re-scaled by dividing by the re-scaled coefficients provided in Table I. By applying the mask to the original image and using the re-scaled dimensions, the function produces a zoomed in image of the TicTacToe board, illustrated in Figure 2.

B. GetBoardDimensions

Using the zoomed image in Figure 2 as an input, the goal of the GetBoardDimension is to determine the dimensions of the board as well as its center. The image is first binarized using the Black Threshold indicated in Table I. Although some of the board may not be determined as black, performing morphological dilation on the negative image ensures a better form of the board. Finally, by using this dilated image of the board, MATLAB's *regionprops* function is employed again to determine the centroid and the dimensions of the region indicated in Figure 3. Each direction's dimensions is divided by three (number of squares in a TicTacToe board) to determine an average x and y threshold, used to locate a new move.

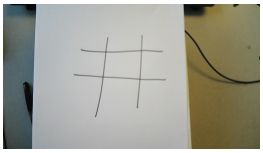

Fig. 1. Webcam Captured Image

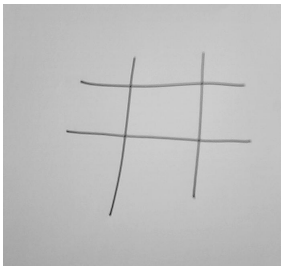

Fig. 2. Zoomed Image

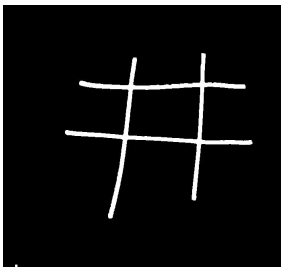

Fig. 3. Dilated Zoomed Image

C. CaptureMove

The CaptureMove function is called when a new move is acquired. The first step is to obtain the difference between the last saved state and the new state which highlights the new move. To better define the move and to remove any noise, the image is thresholded according to the Difference Threshold indicated in Table I. Furthermore, in case any singular noise pixels is captured as well, morphological opening is applied to elimiate these singularities. The output of these filters is an image of the new move. Using the *regionprops* function again, the centroid of the image can be found. Based off the center of the board and it's dimensions found by the GetBoardDimensions function, the position of the new piece can be determined relative to the center of the board. Finally,

by counting the number of connected components in the image's complement, the piece shape can be determined. An X will have a single connected component (the background), and the O shape will have two connected components (the background and the interior of the O). The results of the filtering is shown in Figure 4.

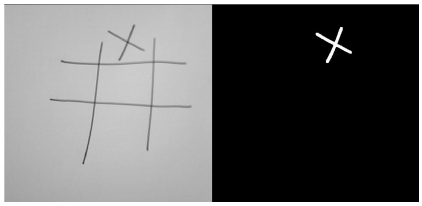

Fig. 4. Move Montage

D. ReturnComputerMove

The goal of ReturnComputerMove is to simply return a move so that user can continue. Once the results of the CaptureMove function is obtained, the captured move and shape piece is updated on a simulated TicTacToe board. The computer then selects at random empty location and deploys the opposite piece.

IV. Experimental Results

The algorithm performed excellent in optimal conditions and the results were highly repeatable (demonstrated in class[4]). In order to better test the limitations of the algorithm several noises and transformation were added to the captured images, to simulate environmental problems. These simulated environments are attempted to be filtered and their performances are recorded. Section IV-A add a motion blur to the images to simulate an unstable camera. Section IV-B adds a vertical sheer to the image to simulate a non-flat Tic-Tac-Toe game. Section IV-C and IV-D adds various noise that could be present when using a poor camera or a transmitted image.

A. Motion Blur

Motion Blur was incorporated into the image using MATLAB's fspecial function. Motion blurs were tested at various pixel lengths at $45°$. Motion of 5, 15, 25 and 35 pixels performed as expected and all shapes were properly determined and positioned accurately. At 45 pixels, the center of the board is no longer captured accurately, as illustrated in Figure 5 and the board's centroid and dimensions are skewed. To properly find the required measurements, the SURF (speeded up robust features) algorithm can be used. The four strongest SURF points tend to be the four corners of the inner square of the Tic-Tac-Toe board, as depicted in Figure 6. Using the co-ordinates of the four points, the center of the Tic-Tac-Toe board can be calculated through their average, and the x and y thresholds can be determined through the difference of the points x and y positions. This approach allows for a maximum amount of 60 pixel motion blur to be present while not affecting the game play. This is due to the SURF algorithm choosing noisy pixels as the corners instead of the appropriate pixels. Using a median filter along with SURF, is expected to solve the problem.

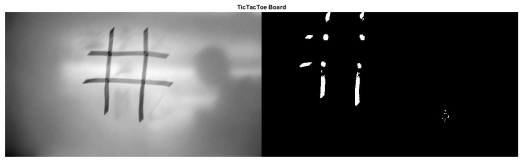

Fig. 5. Motion 45 pixels board

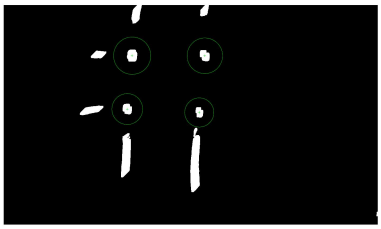

Fig. 6. Motion 45 pixels board SURF

B. Vertical Sheer

Vertical Sheer is a form of spatial transformation, which was applied to the images using the imtransform function. At a $5°$ sheer, the algorithm cannot properly recognize the center of the board, skewing the board dimensions once again. Figure 7 illustrates the clumps of noisy pixels in the corner of the image that are affecting the dimensions. Using a 9x9 median filter, most of the noise could be eliminated which allows the GetBoardDimensions function from Section III-B to accurately position the board, allowing the future pieces to be properly positioned once again. This approach is functional until a maximum of $45°$ sheer is applied. Figure 8 illustrates the noisy corner pixels on the board at $45°$ sheer. Performing local histogram enhancements by measuring the averages and variances in each neighbourhood, is expected to solve this problem.

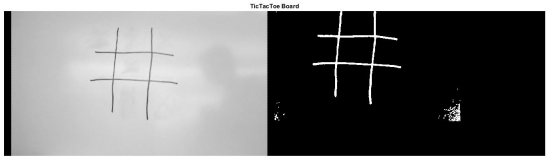

Fig. 7. Vertical Sheer board at 5 degrees

C. Salt-and-pepper Noise

Salt-and-pepper noise, as known as impulse noise, is a noise caused by sharp and sudden disturbances in image signal. Because of the sharp spikes in the image signal, their appearance in a gray-scale image is black or white. We first imposed noise

[4]https://www.youtube.com/watch?v=0Tg5WOAFIhM

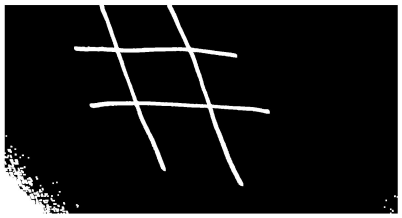

Fig. 8. Vertical Sheer board at 45 degrees

in different density, i.e. portion of pixels that are corrupted, on the images. As a common practice, salt-and-pepper noise should be dealt with median filter or other low-pass filters. We then applied both a median filter and a Gaussian filter on the images before testing them with our algorithm. After repetitive experiments, we are able to correctly identify shapes and locations of user input until 15% of corrupted pixels. At a density of 20%, the algorithm are not able to register the locations of user inputs correctly. Figure 9 illustrates the sparse noise that are detected after pre-processing the image when the density level is 20%. Because of the sparse noise, the algorithm fails to measure the dimensions of the board, hence registers the inputs incorrectly.

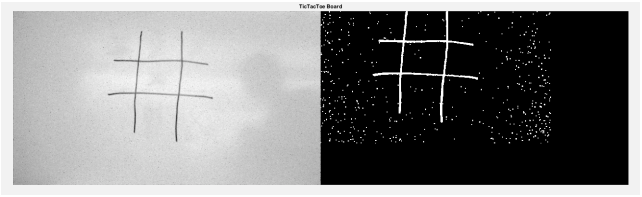

Fig. 9. Salt and pepper noise with 20% occurance probability

D. Gaussian Noise

Gaussian noise is a type of statistical noise with probability density function equals normal distribution (Gaussian distribution). Since the noise is random, they are independent with each other and hard to eliminate using filters. We impose noise with variances range from 0.01 to 0.03 times of the variance of the original signal. We then apply common practices to reduce Gaussian noise, including averaging the neighbouring pixels and applying Gaussian filter, both eliminating pixels with higher frequency. We are able to correctly identify shapes and locations of user input until the variance is 0.015 times the original variance. When the variance of noise keeps increasing, there are more noisy pixels at the border of the image, hence corrupting the dimension registration of the playing board. Figure 10 shows that when the variance equals 0.03 times the original variance, the noise severely corrupts the image and fails the recognition.

V. Conclusion

In this project, we implement an algorithm for producing an augmented reality version of tic-tac-toe. Using several image processing techniques, we are able to recognize the

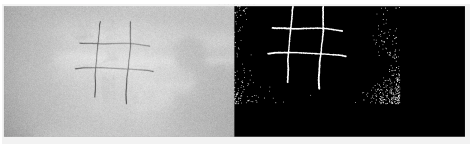

Fig. 10. Gaussian noise with 0.03 times the original signal variance

borders and the locations of the user input. After reading in the preprocessed data, our algorithm successfully displays the computer's decision on the counter move, finishing the output requirement for an augmented reality game. We further conduct experiments with images that are impacted by different types of noise, and receive encouraging results with images that have reasonable amount of noise. Although the results are satisfying, we are open to incorporate the reality scene with computer objects in the future. For example, using a projector to project the computer move onto the paper/board used as input, or using sensors from mobile devices to accomplish the similar goal.

References

[1] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE computer graphics and applications*, vol. 21, no. 6, pp. 34–47, 2001.

[2] M. Serino, K. Cordrey, L. McLaughlin, and R. L. Milanaik, "Pokémon go and augmented virtual reality games: a cautionary commentary for parents and pediatricians," *Current opinion in pediatrics*, vol. 28, no. 5, pp. 673–677, 2016.

[3] F. Zhou, H. B.-L. Duh, and M. Billinghurst, "Trends in augmented reality tracking, interaction and display: A review of ten years of ismar," in *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2008, pp. 193–202.

[4] A. L. Janin, D. W. Mizell, and T. P. Caudell, "Calibration of head-mounted displays for augmented reality applications," in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*. IEEE, 1993, pp. 246–255.

[5] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua, "Fully automated and stable registration for augmented reality applications," in *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2003, p. 93.

[6] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators and virtual environments*, vol. 6, no. 4, pp. 355–385, 1997.

[7] C. Burdea Grigore and P. Coiffet, *Virtual reality technology*. London: Wiley-Interscience, 1994.

[8] C. Kimer, E. R. Zorzal, and T. G. Kirner, "Case studies on the development of games using augmented reality," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, vol. 2. IEEE, 2006, pp. 1636–1641.

[9] J. Maguire and D. Saltzman, "Augmented reality tic-tac-toe."