Computer Vision in Robotic Zen Garden Systems: An Embedded Approach for Contour Detection and Pattern Generation

1st Pongsarun Tisuwan
Department of Electrical Engineering
Silpakorn University
Nakhon Pathom, Thailand
Tisuwan_P@su.ac.th

2nd Thanapat Pimkaew
Department of Electrical Engineering
Silpakorn University
Nakhon Pathom, Thailand
Pimkaew_T@su.ac.th

3rd Jongkol Ngensup
Department of Electrical Engineering
Silpakorn University
Nakhon Pathom, Thailand
Ngensup_J@su.ac.th

4th Jirayut Tomyim
Department of Electrical Engineering
Silpakorn University
Nakhon Pathom, Thailand
Tomyim_J@su.ac.th

5th Tanpisit Atipasaworn
Department of Electrical Engineering
Silpakorn University
Nakhon Pathom, Thailand
Atipasaworn_T@su.ac.th

6th Kittithuch Paponpen*
Department of Electrical Engineering
Silpakorn University
Nakhon Pathom, Thailand
Paponpen_K@su.ac.th

2025 13th International Electrical Engineering Congress (iEECON) | 979-8-3315-4395-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/iEECON64081.2025.10987755

***Abstract*—This paper presents an integrated system that combines computer vision and robotics to automate the creation of traditional Zen garden patterns. The system uses a single Raspberry Pi to detect contours in real time and generate aesthetically pleasing patterns around them, mimicking natural elements like water ripples and waves. By leveraging OpenCV for contour detection and custom pattern generation algorithms, the system produces dynamic, high-quality designs based on the placement and different sizes and shapes of objects within the garden. The experimental results of this system include high detection accuracy, with precision and recall rates exceeding 90% for object classification, and the ability to generate smooth, fluid patterns that closely align with traditional Zen garden aesthetics. Furthermore, the system demonstrates robust real-time performance and ensures seamless motor control. This paper showcases a cost-effective and efficient solution for blending robotics with creative automation in Zen garden design.**

***Index Terms*—Computer vision, Contour detection, Embedded systems, Motor control, Pattern generation, Raspberry Pi, Robotics, Zen garden.**

I. Introduction

Zen gardens, deeply rooted in Japanese tradition and Zen Buddhist philosophy, symbolize natural landscapes through the careful arrangement of stones, sand, and gravel. Raked sand patterns mimic water ripples, while rocks represent islands or mountains, creating spaces for meditation and mindfulness. Emphasizing simplicity (Kanso), asymmetry (Fukinsei), and naturalness (Shizen), these gardens blend spiritual practice with aesthetic beauty [1].

Automating Zen garden design with robotic systems enables precise and intricate patterns, surpassing the limitations of manual creation, especially over large areas. This technology allows for dynamic, customizable designs that adapt to object placement. Studies, such as [2], demonstrate the use of robotic arms and stepper motors for reproducible sand patterns, while research in [3] highlights the accuracy and flexibility robotic systems bring to evolving garden designs.

Incorporating computer vision into robotic Zen garden systems enables automated pattern generation by identifying objects like stones and trees in real-time and designing intricate, adaptable arrangements around them. This approach ensures precise, visually consistent patterns across large areas or frequent updates. Studies such as [4] and [5] explore object detection techniques for guiding motor functions and real-time adjustments in systems like 3D printing, while [6] highlights their use in automating smart gardening tasks, such as plant care. These examples emphasize the importance of object detection in improving precision and adaptability for automated gardening applications.

Pattern generation algorithms find diverse applications across fields, employing techniques like production rules for recursive fractal-like designs [7] and Voronoi diagrams to create balanced, natural-looking spatial divisions [8]. Parametric equations enable precise control for generating smooth curves and spirals [9], while Perlin noise is used to produce organic, wave-like patterns [10]. Additionally, Bezier and parametric curves provide flexibility for defining smooth, adaptable paths [11], making these algorithms versatile tools for dynamic and complex pattern creation.

Motor control systems with computer vision enable precise object detection and movement adjustment, advancing robotic automation for tasks like pattern generation. This paper explores real-time object detection integrated with robotic Zen garden design, where a Raspberry Pi system identifies objects like stones and creates intricate, meditative patterns

*Corresponding author

around them. This approach ensures accuracy, adaptability, and consistency, modernizing traditional Zen gardens through advanced technology while blending aesthetics with robotics.

II. System Structure

A. Hardware Components

Fig. 1. presents the architecture of the proposed system. The hardware components in the proposed system include various power supplies, such as the 5V-12V/3A-1.5A power bank for the Raspberry Pi, the 5V 3A HIECUBE HE05P15LRN for an isolated circuit, and a 12V 3A power source for the stepper motors. The Raspberry Pi Camera Module 2.1 connects via the CSI port. The Raspberry Pi 4 Model B features a 1.5 GHz processor, 4 GB RAM, and 11 GPIO pins are used for motor control. An optocoupler isolation circuit (TLP521) isolates six signal channels to protect the processor from return currents. The A4988 stepper motor driver manages motors with up to 2A current and supports 1/16 micro-stepping for precision. Finally, the NEMA 17 stepper motor, with a 1.8° step angle with 45-65 Ncm rated torque, provides precise movement control for the system.

B. Software Components

The software components of the system are structured to ensure efficient performance across various functionalities. The computer vision and object detection module handles the real-time detection and recognition of objects, providing input for pattern generation. The pattern generation module creates the design or pattern based on the detected objects and system requirements. The motor control translates the generated patterns into motor commands, driving the hardware to draw in the physical environment. The camera interface module handles image capture, while the Raspberry Pi serves as the processor, bridging the vision system and the processing unit.

III. Contour Detection

Fig. 2 presents a flowchart outlining the integration of computer vision techniques for contour detection in the proposed robotic Zen garden system. The process starts by initializing the PiCamera [12] and configuring image capture parameters, as detailed in TABLE I. Continuous frame capture follows, with each frame cropped and converted from grayscale to binary format. Gaussian blur is applied to minimize noise, and binary thresholding is used to highlight key objects [13]. Morphological operations are then performed to refine the binary image, ensuring accurate contour extraction [14]. These contours are utilized as inputs for the pattern generation algorithm, creating a feedback loop that enables the generation of intricate and detailed patterns.

IV. Pattern Generation

The process of pattern generation begins by acquiring contour inputs and verifying their presence. For single objects or merged objects, the system calculates the centroid of the largest contour and scales it. The scaled contour is then

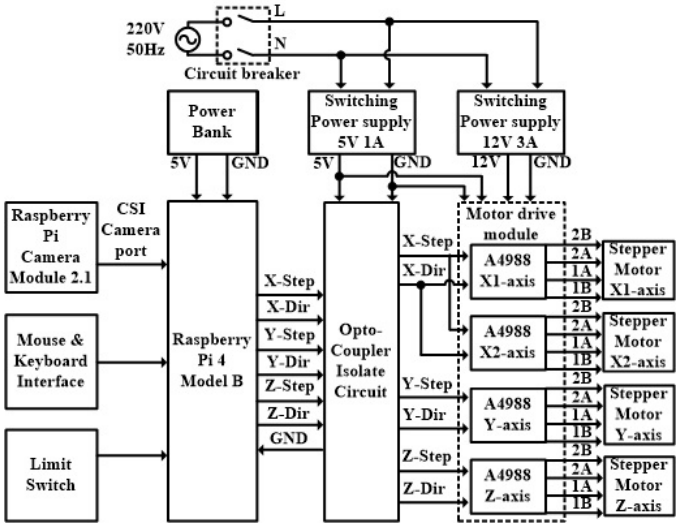

Fig. 1. The proposed system structure.

confined to a specified frame, and the data is downsampled for further processing. Following this, the system identifies the endpoints of the contour using the initial point, smooths it with cubic spline interpolation, and eliminates any points outside the frame. The refined contour pattern is then returned, completing the procedure. In cases where two separate objects are detected, the process follows the same steps as the single-object scenario until the contour smoothing stage. After smoothing, the primary contour is identified and isolated, halting when it overlaps with another object. The remaining contours are subsequently processed, with points outside the frame being removed. The finalized refined contour pattern is then returned, ensuring precise handling and accurate processing for multiple objects. The process flow is depicted in Fig. 3.

V. Performance Evaluation Matrices

To measure the correctness of contour extraction, several metrics and approaches can be used. The key is to compare the extracted contours with the ground truth or desired outcome to assess their accuracy and reliability.

TABLE I
Initial Values for Contour Detection and Pattern Generation

Contour Detection	
Camera resolution	1280 x 720 pixel
Camera frame rate	25 FPS
Zoom settings	(0.175, 0.175, 0.65, 0.65)
Width of cropped image	464 pixel
Height of cropped image	652 pixel
Gaussian blur	(5, 5)
Binary threshold values	60, 255
Morphological kernel size	(60, 60)
Pattern Generation	
Epsilon for down samples	2
Smoothing factor for Cubic spline	400

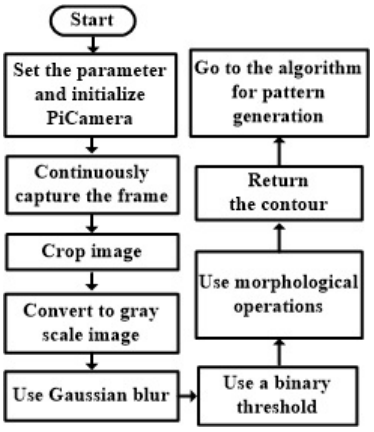

Fig. 2. The flowchart for object detection.

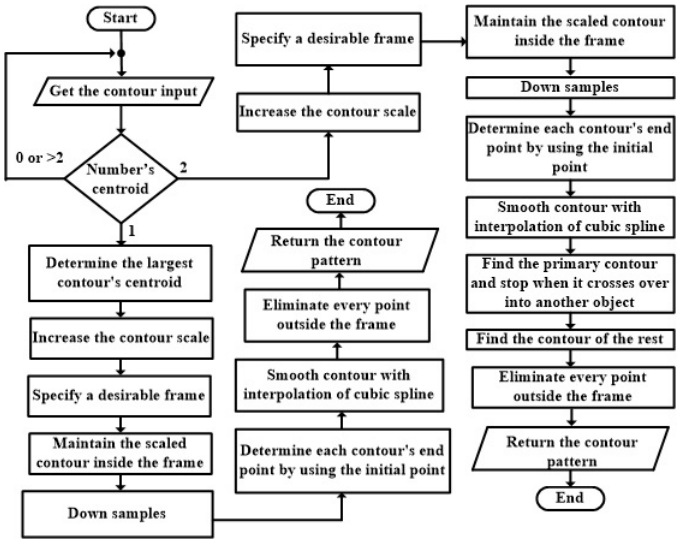

Fig. 3. The flowchart of pattern generation.

A. Intersection over Union (IoU)

IoU is a widely used metric in object detection tasks, and it can also apply to contour extraction. It measures the overlap between the extracted contour (predicted) and the ground truth contour (actual) as shown in equation 1.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{1}$$

An IoU closer to 1 represents a higher accuracy of contour extraction. Values above 0.5 are generally considered acceptable in many applications.

B. Precision and Recall

After evaluating IoU, the precision and recall can be calculated as following equations,

$$Precision = \frac{True\ positive}{True\ positive + False\ positive} \tag{2}$$

$$Recall = \frac{True\ positive}{True\ positive\ + False\ negative} \tag{3}$$

Where *True Positives* are the detected objects that match ground truth objects (IoU $\geq$ threshold). *False Positives* are the detected objects that don't correspond to any ground truth objects (IoU $<$ threshold or no match) and *False Negatives* are the ground truth objects that were missed by the detection algorithm. Precision is used to measure how much of the generated pattern is correct while recall is used to measure how much of the expected pattern was successfully generated.

C. F1 Score

The F1 score is a statistical measure used to evaluate the performance. It combines precision and recall into a single metric to provide a balanced measure of accuracy with the following equation.

$$F1 = 2 \cdot (\frac{Precision \cdot Recall}{Precision + Recall}) \tag{4}$$

D. Position accuracy

Given the absence of an encoder or position sensor, the proposed system operates in an open-loop configuration, which complicates the process of verifying position accuracy due to the lack of direct motor feedback. This challenge becomes particularly evident when working with non-geometric shapes, where the pattern does not adhere to simple, predictable forms such as circles or rectangles. Nonetheless, image processing techniques can be utilized to obtain position accuracy data.

The visual pattern is created using contour detection and a pattern generation algorithm, while the actual pattern is captured via a camera once the robotic drawing is completed. These two patterns are overlaid into separate layers using image processing techniques to identify positional errors. Furthermore, key points in the overlaid patterns are analyzed. For shapes requiring specific key points, the positional accuracy is assessed by calculating the deviation between corresponding key points in the visual pattern and the actual pattern, as expressed in the following equation.

$$PositionError = \sqrt{(x_{act}^2 - x_{vis}^2) + (y_{act}^2 - y_{vis}^2)} \tag{5}$$

Where (x_{act}, y_{act}) represents the key point of the actual pattern produced by the robotic drawing, while (x_{vis}, y_{vis}) corresponds to the key point of the visual pattern generated through the object detection and pattern generation algorithm. This process is repeated for multiple points within the pattern to assess the overall positional accuracy.

VI. Experimental Results

A. Overall system configuration

Fig. 4 shows a robotic drawing system setup controlled by a Raspberry Pi 4 Model B, which includes several components organized on a workbench. The system features a Raspberry Pi camera module V2.1 for visual processing, a drawing stick mounted on a frame, and a sand tray as the drawing surface. Multiple stepper motors (X1, X2, Y, and Z) control the movement of the drawing stick across the tray, with

limit switches (LS1-LS5) ensuring precise positioning. An additional circuitry setup, highlighted in yellow, includes a motor drive module (A4988), an optocoupler isolation circuit, power supplies (5V1A and 12V3A) and a power bank. Control and monitoring are managed through a connected monitor, keyboard, and mouse interface.

The implementation utilizes Python version 3.9.2, along with libraries and packages including *cv2*, *picamera.array* (PiRGBArray), *picamera* (PiCamera), *time*, *numpy*, *pigpio*, *threading*, *scipy.interpolate* (CubicSpline), *os*, and *matplotlib*. The software dependencies are based on Raspberry Pi OS Raspbian, with the camera interface configured at a resolution of 1280x720 and a frame rate of 25 FPS.

A set of 3D-printed objects categorized into simple and complex shapes, with each object's dimensions labeled is display in Fig. 5. The simple shapes, shown on the left, include basic geometrical forms like a circle, rhombus, equilateral triangle, hexagon, square, and crescent, all with a consistent height of 15 mm and a diameter or width of 40 mm. The complex shapes, shown on the right, include more intricate forms such as tree models, stepped and layered structures, and varied naturalistic shapes, with heights ranging from 10 mm to 75 mm and various base dimensions.

B. Procedure for experimentation

The process begins by setting the starting position and moving the pen to the home point. Next, the system determines the maximum steps along the x - and y- axes by moving the pen independently along each axis until it reaches the respective limit switch. These maximum steps are then used to calculate the transformation ratio for both axes, pixel per step ratio. After that an object is placed on the sand tray, and its contour is detected. The detected contour is used to generate a pattern, which is captured for reference. Pressing the start button on the keyboard initiates the drawing process, during which the system follows the generated pattern. Upon completing the drawing, the final pattern is captured, and the pen is moved to the end position, marking the end of the process.

C. Evaluation of object detection

Object detection results for eight distinct cases, each involving different object configurations and conditions are shown in Fig. 6. Cases range from separated and overlapped simple objects to complex objects under varying conditions, including sand coverage. This comparison highlights the detection algorithm's performance under various complexities described in detail in Fig. 7.

Fig. 7 presents the precision, recall, and F1 score performance metrics across varying IoU thresholds for these eight experimental cases. Case 1 consistently achieves the highest precision, recall, and F1 scores across all thresholds, maintaining near-perfect values even as the IoU threshold increases, indicating strong performance and reliability. In contrast, Case 7 and Case 8 exhibit very low scores across all metrics, suggesting poor detection accuracy and overlap alignment.

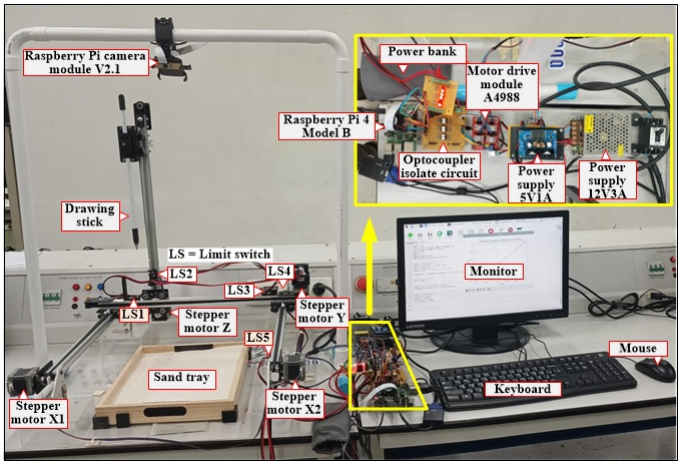

Fig. 4. Robotic Zen garden system setup with Raspberry Pi control and stepper motor mechanism.

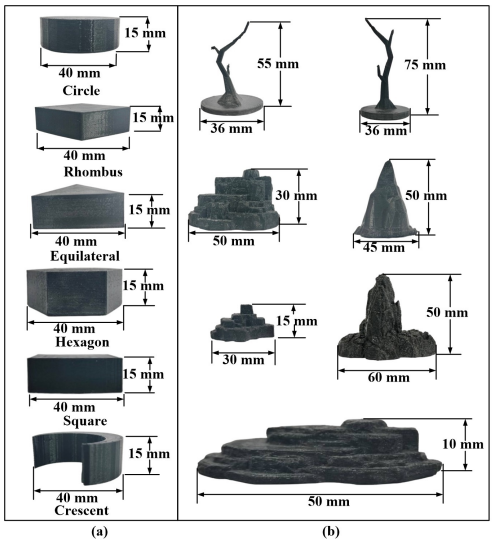

Fig. 5. Dimensions and types of 3D-printed (a) simple and (b) complex shape objects.

Cases 3 and 4 show moderate but declining performance as the IoU threshold increases, indicating some sensitivity to stricter overlap requirements. This visualization highlights that certain cases (like Case 1) are robust across thresholds, while others significantly degrade as overlap criteria become more stringent. These results could be appropriate for initial insights into pattern generation performance, especially in terms of alignment accuracy and overlap (measured by IoU).

D. Evaluation of pattern generation

Fig. 8 and 9 illustrate the visual workflow for creating patterns with single and separated objects. These figures demonstrate that the object detection and pattern generation algorithms, as outlined in Fig. 2 and 3, successfully produce the input pattern, which guides the stepping motor to precisely replicate the design on sand.

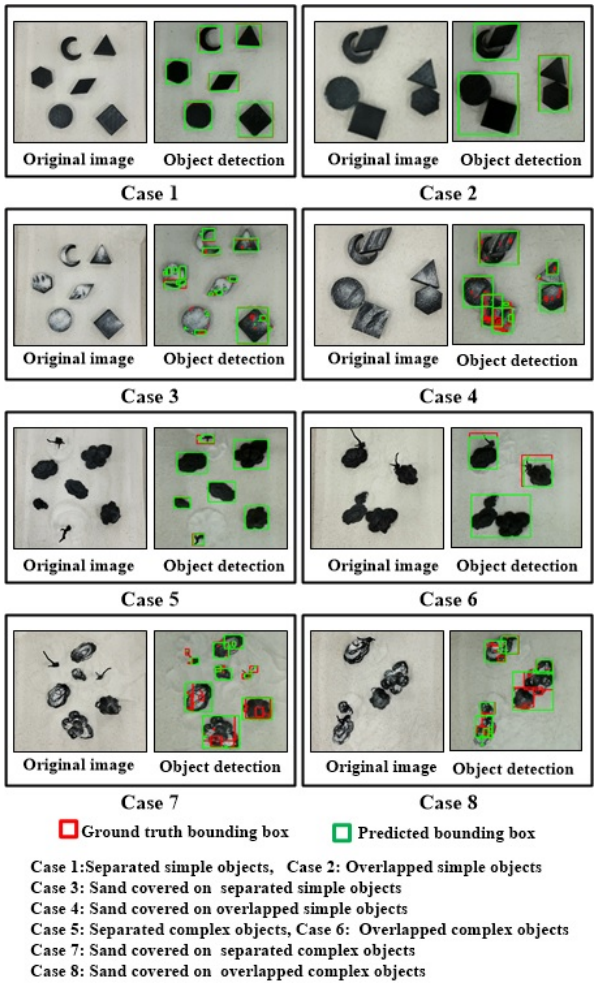

Fig. 6. Evaluation of object detection performance across varied object configurations and conditions

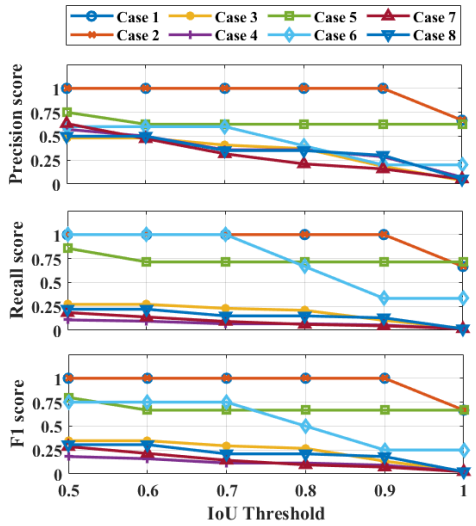

Fig. 7. Impact of object complexity, overlap, and sand coverage on precision, recall, and F1 scores across IoU thresholds

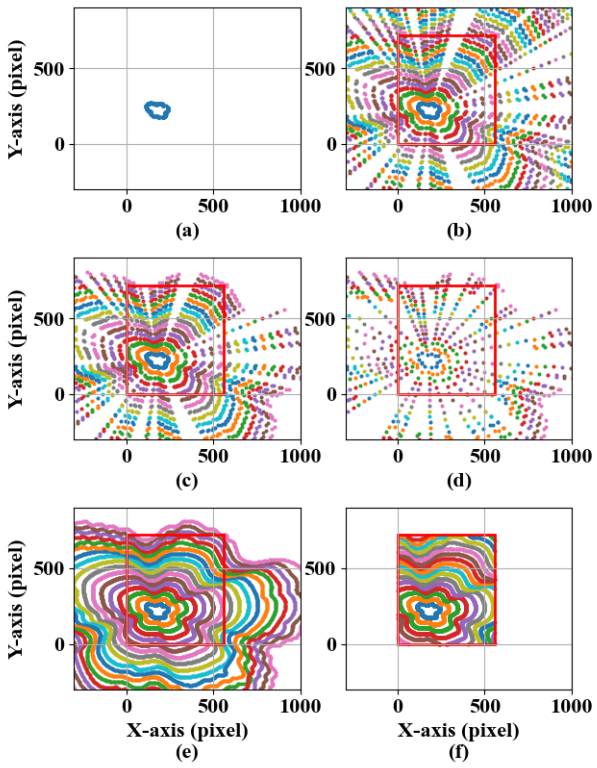

Fig. 8. The process for creating patterns with a single object involves: (a) obtaining the contour input, (b) scaling the contour and specifying a desirable frame, (c) maintaining the scaled contour within the frame, (d) downsampling, (e) smoothing the contour, and (f) removing points outside the frame.

E. Evaluation of position accuracy

Position error verification is initially conducted using a robotic Zen garden system on A4 paper, as it provides a clear and measurable reference for numerical evaluation. After obtaining these results, the paper is replaced with sand to replicate the process in the actual proposed system.

Figure 10 illustrates the position error observed when the actual patterns are plotted on A4 paper. The error between the actual and generated patterns is minimal, averaging approximately 8 pixels in distance, indicating a slight shift. This highlights the importance of precise calibration of the camera and pixel home position to minimize errors.

Following verification through drawing on paper, the sand tray is used to evaluate the proposed system's performance. Fig. 11 demonstrates the system's capability to accurately replicate the generated patterns across different scenarios, including a single object, overlapping objects, and separated objects, all with minimal position errors.

VII. Conclusion

This paper introduces a comprehensive system that integrates computer vision and robotics to automate the creation of traditional Zen garden patterns. Powered by a single Raspberry Pi, the system identifies object contours in real time

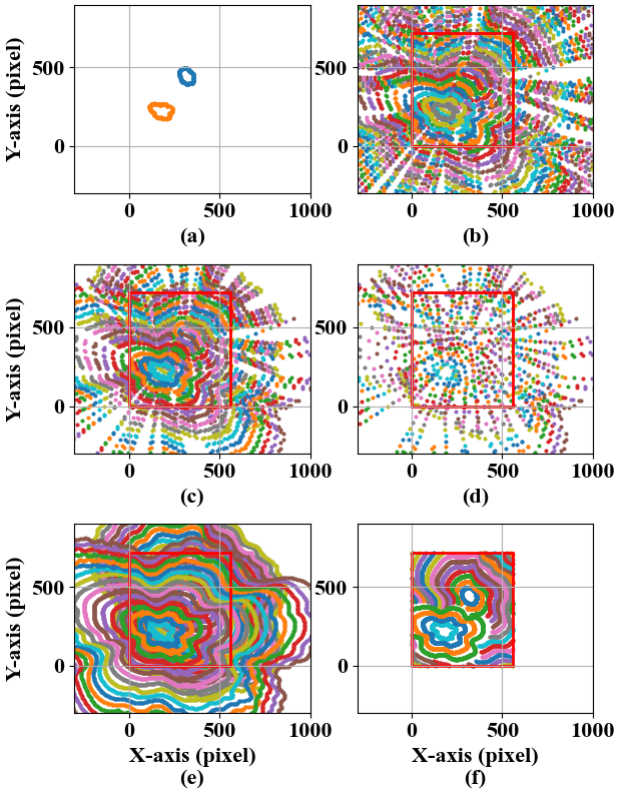

Fig. 9. The process for creating patterns with separated objects includes: (a) obtaining the contour input, (b) scaling and framing the contour, (c) maintaining the scaled contour within the frame, (d) downsampling, (e) smoothing the contour, identifying primary and secondary contours, and (f) removing points outside the frame.

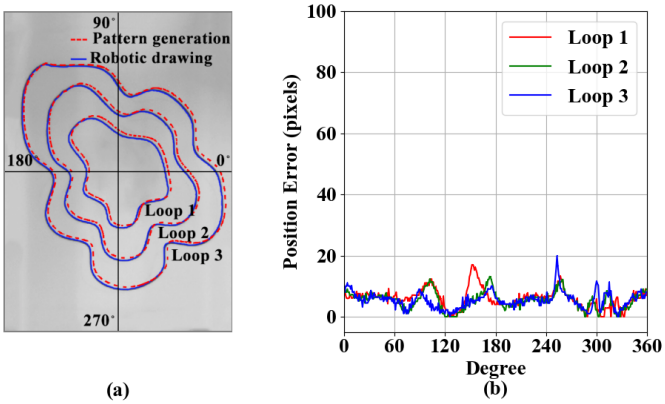

Fig. 10. Position error verification for a single object involves: (a) overlaying the generated pattern with the robotic drawing pattern on A4 paper, and (b) analyzing the correlation between position error and the direction of key points in degrees.

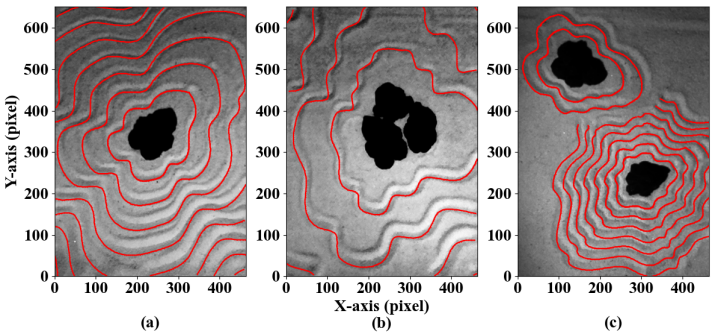

Fig. 11. The comparison between the generated patterns and robotic drawing patterns on sand is performed for (a) a single object, (b) overlapping objects, and (c) two separated objects.

and generates aesthetically pleasing patterns. Using contour detection and adaptive pattern generation algorithms, it adapts dynamically to variations in object placement, size, and shape. Experimental results demonstrate high detection accuracy, supported by strong precision and recall metrics. Moreover, the system produces smooth, fluid patterns that closely match the robotic-drawn designs with minimal positional error. The proposed solution provides an efficient, cost-effective method for combining robotics with artistic automation in Zen garden design.

ACKNOWLEDGMENT

This work was supported by Silpakorn University Research, Innovation and Creative Fund.

REFERENCES

[1] David and Michiko Young, The Art of the Japanese Garden, Tuttle Publishing, 2005.
[2] DIYODE Magazine, "Building a Robotic Zen Garden," DIYODE Magazine, Issue 48, pp. 56-60, 2022. [Online]. Available: https://diyodemag.com. Accessed: Oct. 5, 2024.
[3] M. S. K. Yeo, S. M. B. P. Samarakoon, Q. B. Ng, M. A. V. J. Muthugala, and M. R. Elara, "Design of Robot-Inclusive Vertical Green Landscape," Buildings, vol. 11, no. 5, p. 203, May 2021.
[4] A. Patel, S. Shah, and P. Mehta, "A Review of Object Detection Techniques," International Journal of Computer Applications, vol. 182, no. 30, pp. 22-28, Sep. 2018.
[5] X. Zhang, Y. Wang, and L. Li, "Research on Process Control of 3D Printing Technology Based on Multi-sensor Technology," Journal of Physics: Conference Series, vol. 1234, no. 1, pp. 012-345, 2020.
[6] Y. Singh, M. K. Sharma, and R. Kumar, "An Overview of Smart Garden Automation," International Journal of Engineering Research and Technology (IJERT), vol. 9, no. 12, pp. 432-438, Dec. 2020.
[7] P. Prusinkiewicz and A. Lindenmayer, The Algorithmic Beauty of Plants. New York, NY, USA: Springer-Verlag, 1990.
[8] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd ed. Chichester, U.K.: Wiley, 2000.
[9] A. Gray, Modern Differential Geometry of Curves and Surfaces with Mathematica, 2nd ed. Boca Raton, FL, USA: CRC Press, 1997.
[10] K. Perlin, "An image synthesizer," ACM SIGGRAPH Comput. Graph., vol. 19, no. 3, pp. 287–296, Jul. 1985.
[11] G. Farin, Curves and Surfaces for Computer-Aided Geometric Design, 4th ed. San Diego, CA, USA: Academic Press, 1996.
[12] Raspberry Pi Foundation. (2020). PiCamera Documentation. Raspberry Pi.
[13] Otsu, N. (1979). "A Threshold Selection Method from Gray-Level Histograms." IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62-66.
[14] Suzuki, S., & Abe, K. (1985). "Topological structural analysis of digitized binary images by border following." Computer Vision, Graphics, and Image Processing, 30(1), 32-46.