TSP-Bot: Robotic TSP Pen Art using High-DoF Manipulators

Daeun Song, Eunjung Lim, Jiyoon Park, Minjung Jung and Young J. Kim

Abstract—TSP art is an art form for drawing an image using piecewise-continuous line segments. We present TSP-Bot, a robotic pen drawing system capable of creating complicated TSP pen art on a planar surface using multiple colors. The system begins by converting a colored raster image into a set of points that represent the image's tone, which can be controlled by adjusting the point density. Next, the system finds a piecewise-continuous linear path that visits each point exactly once, which is equivalent to solving a Traveling Salesman Problem (TSP). The path is simplified with fewer points using bounded approximation and smoothed and optimized using Bézier spline curves with bounded curvature. Our robotic drawing system consisting of single or dual manipulators with fingered grippers and a mobile platform performs the drawing task by following the resulting complex and sophisticated path composed of thousands of TSP sites. As a result, our system can draw complicated and visually pleasing TSP pen art.

I. INTRODUCTION

With the tremendous growth of digital technologies, digital art has become one of the largest art fields since the early 1960s. Early pioneers of digital art were not only artists but also engineers, computer scientists, and mathematicians who challenged traditional art standards with new technologies. Typically, most of the work focuses on investigating the production of artistic images in virtual space, which enables a wide variety of expressive and aesthetic styles using computer algorithms.

Traveling Salesman Problem Art, abbreviated as TSP art, is one of the representative examples of creating artistic work using computer algorithms. It was first invented by mathematician Robert Bosh [1]. TSP art is an art piece that represents the original digital image with piecewise-continuous line segments. TSP art involves not only the creative process of computer algorithms but also fits the nature of a robotic task, whose fundamental mission is to follow a path accurately and efficiently.

As hardware technology advances, efforts have been made to bring these digitally generated artistic results into the physical space using machines [2], [3]. As robots are capable of delivering long and complex motions, we believe TSP art is best suited for the robotic drawing system, recognizing the original purpose of the robot. Our work focuses on a robotic TSP pen art system that is supported by complex and sophisticated motions. Our goal is not to supplant human artists but rather to aid and demonstrate the potential of interdisciplinary collaboration between robotics and art.

Main Results In this paper, we present a multi-color robotic pen drawing system, TSP-Bot, that transforms a digital raster image into long, continuous robotic paths that replicate the original image’s tone and color and draw the result on a planar canvas surface (Fig. 1). The system takes any raster image with color as input. In order to be reproduced by pens with a limited number of colors, the color image is channel-split into user-provided color spaces, such as the CMYK color space, and saved as separate image files. We use a stippling algorithm to displace points so that the points’ density represents the image’s tone. The system finds piecewise-continuous line segments that visit every point by solving TSP. We then perform path optimization with bounded curvature so the robot can follow smoothly. The drawing is rendered on a target canvas plane using our robotic hardware. We carry out drawing experiments using single and dual high degree-of-freedom (DoF) manipulators, the former with a mobile platform, to show that our system can create artistic and complicated TSP pen art in a physical space. We present diverse TSP drawing results.

In summary, our technical contributions include:

- A novel approach for processing color raster images suitable for limited-color-palette pens. This method involves splitting the input image into user-defined color spaces, such as CMYK, and using high-density TSP art to represent image tone accurately.
- A simple path optimization with bounded curvature to ensure smooth robot movement during the drawing process, coupled with the TSP solver.
• A novel drawing tool design with a tool-change mechanism to ensure robust pick and place using a 3-finger gripper. This new design enhances the reliability and versatility of the drawing system.

II. PREVIOUS WORK

A. TSP art

TSP art, initially introduced in [1], represents an image’s tonal quality with a single continuous path, formulated as a Traveling Salesman Problem (TSP), achieved by placing dots on the image and connecting them with piecewise-continuous line segments. The stippling technique, to effectively reproduce the image’s shading with the density of the points, has been explored by much research [4], [5]. Most recently, Weighted Linde-Buzo-Gray Stippling method [6] was introduced. It dynamically splits cells until achieving the desired number of representative vectors, and reformulates it to split and merge cells based on size, grayscale level, or image variance.

Once the stipples are placed, finding the shortest possible path that visits every stipple exactly once is equivalent to solving a TSP. Much research has been carried out to efficiently solve TSP, known as NP-Hard. Concorde [7] is an optimization solver widely regarded as the fastest TSP solver for large instances [8]. The resulting single line that resembles the original image’s tone is considered an art form called TSP art. We observe that such complex and continuous paths are suitable for robotic drawing systems to reproduce them on a physical surface with its capability of sophisticated maneuverability.

B. Robotic Drawing

The early history of drawing machines traces back to Harold Cohen’s artistic work with AARON [9]. Recent advances in robotic hardware have spurred diverse artistic applications, exemplified by creations like Paul the robot, which produces portrait drawings by observing subjects and mimicking artistic styles [10], and eDavid, an industrial robot creating paintings with visual feedback and a Non-Photorealistic Rendering (NPR) algorithm [11]. Ongoing research explores color implementation for more vivid drawings [12], [13], alongside the emergence of robotic drawing systems incorporating machine learning techniques, such as stroke order planning based on image segmentation and depth estimation [14], and RoboCoDraw, a personalized avatar character drawing system using Generative Adversarial Network (GAN)-based style transfer [15].

Recent research on machine creativity emphasizes painterly rendering algorithms, which still lag behind human abilities, while new robotic pen drawing systems prioritize the robot’s capabilities. A system for drawing on non-planar surfaces using manipulator impedance control was introduced [16], [17], expanded by SSK [18] for larger surfaces with mobile manipulators. Chitrakar [19] proposes a robotic system that autonomously converts a human face image into a non-self-intersecting curve by solving TSP with a stippling method. We take a similar approach, but our system can draw more colorful drawings and is more suitable for autonomous robotic drawing with an optimized drawing path and a pen-change mechanism.

III. SYSTEM OVERVIEW

We present the overview of our TSP robotic drawing system in Fig. 2. Given a target image $I$, in order to make it drawable for robots, we require a method to map it into the robot’s configuration space. This paper employs TSP art to map the image to piecewise-continuous line segments within the canvas space. The path optimizer smoothens the trajectory by fitting cubic Bézier spline curves with bounded curvature. Subsequently, we determine the corresponding path in the configuration space by solving the path-wise Inverse Kinematics (IK) problem.

We generate TSP art using two main stages as follows:

1) Stippling: From $I$, generate and place a set of points $P \subset \mathbb{R}^2$ to replicate the tone of the original image.

2) TSP Solving: Find a cycling path $\mathcal{X} \subset \mathbb{R}^2$ that visits every point $P$ once and returns to the first point.

In order to reproduce the original image’s color, we split the image $I$ into predefined $n$ color channels $I_i$, $i = 0, \ldots, n-1$, where $n$ is the number of colors. We repeat generating TSP art for each $I_i$. In this paper, we follow a modern, four-color-process printing technique called CMYK [20] that splits the color space of the image into four color channels, cyan, magenta, yellow, and black. Afterward, we perform path optimization. We begin by applying the Ramer-Douglas-Peucker algorithm [21] to simplify the piecewise linear paths with bounded Hausdorff distance between the original curve and the simplified curve. Then, we interpolate the linear paths using cubic Bézier spline curves $\mathcal{X}_* \subset \mathbb{R}^2$ with bounded curvature. We map the path to the end-effector’s configuration $\mathcal{X}_* \subset \mathbb{R}^2 \rightarrow \tilde{\mathcal{X}} \subset SE(3)$ by projecting $\mathcal{X}_*$ onto the target 3D planar canvas space, with end-effector orientation perpendicular to the canvas.

We finally find the joint configurations $\xi$ corresponding to $\tilde{\mathcal{X}}$, which is fed into robots to perform drawing. Our system also considers the manipulator’s reachability to decide the size of the drawable canvas space. We also designed a new pen-drawing tool for a 3-finger gripper to quickly and robustly switch between colored pens.

IV. MULTI-COLOR TSP ART

In this section, we introduce our approach to map the input image $I$ into a drawable robotic path $\mathcal{X}$. We take the TSP art idea with an additional color processing stage to reproduce the color likeness of the original image.

A. Color Processing

We begin by segmenting the color image into distinct color channels $I_0, \ldots, I_{n-1}$. Each channel is then independently processed to generate TSP art pieces. When reassembled, these TSP art fragments collectively replicate the color similarity of the original image. The method for channel separation can be controversial: we adopt the CMYK approach, commonly used for color printing, which separates images...
into cyan, magenta, yellow, and black channels. CMYK is effective due to its analogy to subtractive mixing, prevalent in both print and pen ink application. However, CMYK model may struggle to faithfully reproduce the original image color when working with fewer points. Further insights into alternative approaches are discussed in Sec. VII-C.

B. Point Generation

We perform stippling for each color channel individually, which involves transforming an image into a set of points \( P \subset \mathbb{R}^2 \). This process results in denser point placement in darker regions and fewer points in brighter regions. Numerous stippling methods have been explored to achieve effective image reproduction. In this paper, we adopt the weighted LBG Stippling method\cite{6}. The method progressively divides Voronoi cells from a single point until the desired point density, determined by the weight function, is reached. Additionally, the method allows the merging of the neighboring cells once the point density becomes excessive.

C. TSP Path Generation

The generated points \( P \) can already form an art piece, feasible for robotic motion. However, repetitive up-and-down motions are time-intensive. To address this, we connect all the points in \( P \) to establish a singular path \( X(t) \subset \mathbb{R}^2 \), which makes the robot easier to follow. Finding this path is equivalent to solving TSP, a widely known NP-hard problem. Significant research has been directed towards efficient approximated solutions for TSP. In this work, we utilize the Concorde solver, which relaxes the problem into a Linear Program (LP) and iteratively fixes potential fractional solutions through a cutting plane algorithm\cite{22}. Using the Lin–Kernighan heuristic\cite{23}, which iteratively improves a tour by exchanging pairs of edges while utilizing edge removal, insertion, and recombination to minimize edge crossings, we can generate a path without any edge crossings.

V. ROBOTIC DRAWING

This section outlines the optimization of the drawing path and describes selecting the appropriate canvas dimensions based on the manipulator’s reachability. Furthermore, we detail the robotic curve rendering method, which involves determining joint configurations to execute the drawing task.

A. Path Optimization

Tracing a TSP path composed of piecewise linear segments \( X(t) \) that interpolate \( P \) is feasible for an end-effector to follow, but it is not optimal for realizing the robot motion. We optimize the path \( X^*(t) \) to be better suitable for robotic tracing while approximating the original path with bounds on distance and curvature. The Ramer-Douglas-Peucker algorithm simplifies the linear path by decimating some sub-path that does not significantly contribute to the curve’s shape. This process involves recursively subdividing the path, checking if the distance between a sub-path and the original path is below a threshold \( d_e \), and then discarding it.

After acquiring a simplified piecewise linear path that interpolates a reduced number of points \( P^* \), we interpolate them using cubic Bézier spline curves. The curvature of these curves is bounded by \( \kappa_c \), determined by the maximum acceleration of the end-effector. The entire path comprises \( |P^*| - 1 \) spline curves, each with four control points. Fig. 3 illustrates two spline curves connected at \( p_2 \). The curvature of the Bézier spline curve at \( p_3 \) is evaluated as:

\[
\kappa = \frac{2d}{3c^2},
\]

where \( d = |p_2 - p_3| \) and \( c \) represents the distance between \( p_3 \) and the line formed by \( p_2 \) and \( p_3 \)[24]. Thus, \( \kappa \) can be controlled by fixing \( c \) and increasing \( d \) by displacing \( p_2 \) along \( p_2 - p_3 \) with some scaling factor \( s > 0 \). The final path \( X^* \) is obtained by minimizing \( s \) such that \( sk_i > \kappa_c, \forall i \in |P^*| - 1 \) where \( \kappa_i \) denotes the curvature for \( i \)th spline.

![Fig. 3. Interpolating cubic Bézier curve. \( p_i \)'s represent the control points for the first Bézier spline that interpolates \( p_0, p_3 \).](image)

Optimizing a path of more than 50,000 control points may require a significant amount of computation time. Thus, we employed rather simple path optimization and smoothing techniques to save time. Moreover, since the preceding TSP solver already produced spatially coherent and optimal
drawing routes, our simple approach still produces a highly optimal path and prevents abrupt robot motion.

B. Maximum Size of Canvas Space

After generating the final 2D drawing path, the next step before executing the robotic task is to map the path onto the 3D real-world space. We determine the maximum size of the drawing canvas based on the robot’s reachability. The reachability of the robot is defined by discretizing the robot’s Cartesian workspace and solving the inverse kinematics problem for each discrete point to check if it is reachable [25]. The reachable points are saved as shown in Fig. 4 with colored spheres. The intersection between the reachable point set and the target surface is used to determine the canvas dimension. For dual-arm, we separate the drawing task by color channels and find the intersection between the reachable point set by both arms and the target surface. For mobile manipulator, we split the canvas into sub-canvas based on the manipulator’s drawing size. Then, we repeat the process of drawing and shifting to the next sub-canvas to complete the drawing. After determining the canvas space, we obtain the target drawing poses by projecting the drawing path onto the planar canvas space with fixed end-effector orientation in the opposite direction of the surface normal: $X^* \subset \mathbb{R}^2 \rightarrow \tilde{X} \subset SE(3)$.

C. Robotic Curve Rendering

Once the target drawing poses $\tilde{X}$ are decided, they are fed to robots. Robotic drawing is equivalent to finding a continuous path in the robot’s configuration space so that its end-effector follows the given path; this problem is also known as path-wise IK [26]. Moreover, path-wise IK is suitable for setting a robot’s kinematic and dynamic constraints while following the robot trajectory. In the case of robot drawing, achieving a feasible motion with no sudden jumps is crucial. Plus, it is essential to follow the given end-effector poses accurately so as not to ruin the resulting drawing. Thus, we solve the path-wise IK problem by iteratively solving the IK for the end-effector poses $\tilde{X}$ with the minimum distance objective in the configuration space.
• **Dual Manipulators** consist of two UR5e manipulators, each equipped with a Robotiq 3-finger adaptive gripper. We designed a new pen tool that allows our gripper to hold the pen firmly despite sensing and mechanical errors. The drawing process is fully automated thanks to the gripper and our pen tool change mechanism.

• **Mobile Manipulator** consists of KUKA LBR iiwa 7 R800 manipulator mounted on top of the omnidirectional mobile platform, Ridgeback. A 3D-printed pen tool is rigidly attached to the robot’s end-effector, which requires a manual change of colors. Thanks to its mobility, it is not limited to the drawing canvas size.

We use Robot Operating System (ROS) Melodic framework under Ubuntu 18.04 operating system to communicate with the robots and perform the drawing task. We use C++ and Python for programming, which runs on a PC equipped with Intel i7 CPU and 32 GB memory. We use MoveIt! [27] with TRAC-IK [26] inverse kinematics solver for computing the robot trajectory following the Cartesian path.

### B. Drawing Results

The statistics for the experimental drawings shown in Figs. 1 and 8 are provided in Table I. These statistics include the size of the drawing and the number of stippled points used to solve TSP. Additionally, the time consumed to run the stippling algorithm, solve TSP, and execute the robot is presented. The number of points and the times in the statistics represent the sum of values for every color. Regarding TSP solving time, we impose a time-bound to ensure that the data generation process does not take an excessive amount of time. During the drawing process, we operate the robot at a low speed, specifically 20% of the maximum joint velocity. This is to ensure the safety of both the robot and any collaborators, such as humans.

Fig. 7 shows the digital drawing result of the Heart before and after the path optimization. Note that the TSP path in Fig. 7(a) without path optimization produces a similar result like Chitrakar [19]. The number of points decreased from 21,664 to 16,720, which means about 15% of the points were removed while the shape of the result is nondistinguishable between the two. We experimentally set the parameters of $d_e$ and $\kappa_e$ to 0.5 and 2.0, respectively, given the simplification rate of 15% and the bound for maximum end-effector acceleration.

Fig. 8(a) and (b) show the robotic pen drawing results drawn with a dual-arm robotic setup. Fig. 1, Fig. 8(c), and (d) show the robotic pen drawing results drawn with the mobile manipulator setup. The Heart is drawn in one place with no mobile platform moving. The Violet drawing was split into three regions, and the Ewha Womans University (EWU) graffiti was split into nine regions according to the appropriate canvas space measured in Sec. V-B.

### C. Discussions

**Color Splitting Methods:** We utilize CMYK color channels, commonly used in printing, for our robotic pen drawing system to replicate colors similar to the original image, albeit with a limited color range. However, due to the fewer stippling points used compared to printing, color reproduction may vary depending on the input image’s spectrum. An alternative approach involves generating a custom color palette by clustering raster image pixels into $k$ groups using algorithms (i.e., K-Means, Agglomerative, or DBSCAN), providing representative colors. This method may offer more accurate colors with fewer points, though there’s a risk of mismatch between the pen’s color and the obtained color.

**Robotic Hardware:** We employ two distinct robotic setups, each with unique advantages. The mobile manipulator excels in creating expansive artworks on a larger scale but requires manual color changes, while the dual-arm setup offers autonomous tool-changing capabilities. Our design choice involves constraining the drawing canvas within the shared reachable range of both arms, resulting in sequential drawing by each arm. Alternatively, dividing the task based on arm reachability could enable both arms to draw simultaneously, necessitating duplication of drawing resources like color palettes and tools to maximize concurrency and reachability. This approach is a focus of our future research.

### VIII. Conclusion

We introduce TSP-Bot, a multi-color robotic pen drawing system that translates digital raster images into continuous paths on a physical surface, replicating the original colors by segmenting the image into predefined color spaces. Our robotic hardware executes this intricate task by following paths comprising thousands of points, suitable for robots with high accuracy and repeatability. Demonstrating its capabilities, we utilize both a dual-arm robotic setup with a color change mechanism and a mobile manipulator setup, showcasing flexibility beyond canvas size. Our system produces colorful and aesthetically appealing TSP pen art.

Future directions involve further path optimization to enhance robotic drawing performance, including investigating the effects of varying parameters like $d_e$ and $\kappa_e$ on robot joints and resulting drawings. Additionally, immediate plans include further exploration of the dual-arm drawing setup. Distributing drawing tasks between arms based on reachability to address limitations of restricted canvas space and necessitating efficient collision-aware motion planning methods with new drawing strategies will be required.

### REFERENCES

[1] C. S. Kaplan, R. Bosch et al., “Tsp art,” in Renaissance Banff: Mathematics, music, art, culture. Bridges Conference, 2005, pp. 301–308.
### TABLE I
ROBOTIC DRAWING EXPERIMENTAL STATISTICS

| Robotic Hardware | Dual Manipulators (Fig. 8(a), (b)) | Mobile Manipulator (Figs. 1, 8(c), (d)) |
|------------------|-----------------------------------|----------------------------------------|
|                  | Stary Night                        | Big Ben                                |
| Canvas size (mm²)| 315 × 250                          | 214 × 300                              |
| # of Stippled Points | 81,591                             | 76,257                                 |
| TSP Stippling Time (sec.) | 32                                  | 34                                     |
| TSP Solving Time (sec.) | 6                                   | 11                                     |
| Drawing Time (min.) | 124                                 | 63                                     |
|                  | Heart                              | Violet                                 | EWU                                    |
| # of Stippled Points | 81,591                             | 76,257                                 |
| TSP Solving Time (sec.) | 32                                  | 34                                     |
| Drawing Time (min.) | 124                                 | 63                                     |

**Fig. 8.** TSP pen art results produced by the dual manipulators (left: input image, right: TSP drawing)

[1] P. Cohen, “Harold cohen and aaron,” *Ai Magazine*, vol. 37, no. 4, pp. 63–66, 2016.

[2] T. Lindemerje, J. Metzner, L. Pollak, and O. Deussen, “Hardware-based non-photorealistic rendering using a painting robot,” in *Computer graphics forum*, vol. 34, no. 2. Wiley Online Library, 2015, pp. 311–323.

[3] O. Deussen, S. Hiller, C. Van Overveld, and T. Strothotte, “Floating points: A method for computing stipple drawings,” in *Computer Graphics Forum*, vol. 19, no. 3. Wiley Online Library, 2000, pp. 41–50.

[4] A. Secord, “Weighted voronoi stippling,” in *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, 2002, pp. 37–43.

[5] P. J. Campbell, “The traveling salesman problem: A computational study,” *Mathematics Magazine*, vol. 80, no. 3, p. 238, 2007.

[6] S. A. Mulder and D. C. Wunsch II, “Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks,” *Neural Networks*, vol. 16, no. 5-6, pp. 827–832, 2003.

[7] P. McCorduck, AARON’S CODE: Meta-Art, Artificial Intelligence, and the Work of Harold Cohen. W. H. Freeman & Co, 1990.

[8] P. Tresset and F. F. Leymarie, “Portrait drawing by paul the robot,” *Computers and Graphics*, vol. 37, 2013.

[9] T. Lindemerje, S. Pirk, and O. Deussen, “Image stylization with a painting machine using semantic hints,” *Computers and Graphics*, vol. 37, 2013.

[10] A. Karmov, E. Kopets, S. Leonov, L. Scaler, and D. Butusov, “A robot for artistic painting in authentic colors,” *Journal of Intelligent & Robotic Systems*, vol. 107, no. 3, p. 34, 2023.

[11] L. Scalera, A. Gasparetto, S. Seriani, and P. Gallina, “History of drawing robots,” in *International Symposium on History of Machines and Mechanisms*. Springer, 2024, pp. 3–17.

[12] I. Ilinkin, D. Song, and Y. J. Kim, “Stroke-based rendering and planning for robotic performance of artistic drawing,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 8062–8068.

[13] T. Wang, W. Q. Toh, H. Zhang, X. Sui, S. Li, Y. Liu, and W. Jing, “Robocodraw: robotic avatar drawing with gan-based style transfer and time-efficient path optimization,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10402–10409.

[14] D. Song, T. Lee, and Y. J. Kim, “Artistic pen drawing on an arbitrary surface using an impedance-controlled robot,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4085–4090.

[15] D. Song and Y. J. Kim, “Distortion-free robotic surface-drawing using conformal mapping,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 627–633.

[16] D. Song, J. Park, and Y. J. Kim, “Ssk: Robotic pen-art system for large, nonplanar canvas,” *IEEE Transactions on Robotics*, 2023.

[17] A. Singhal, A. Kumar, S. Thukral, D. Raina, and S. Kumar, “Chitrakar: robotic system for drawing jordan curve of facial portrait,” *arXiv preprint arXiv:2011.10791*, 2020.

[18] F. A. Baqui, J.-H. Lee, A. U. Agar, and J. P. Allebach, “Digital color halftoning,” *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 87–96, 2005.

[19] U. Ramer, “An iterative procedure for the polygonal approximation of plane curves,” *Computer graphics and image processing*, vol. 1, no. 3, pp. 244–256, 1972.

[20] G. Dantzig, R. Fulkerson, and S. Johnson, “Solution of a large-scale traveling-salesman problem,” *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[21] D. Karapetyan and G. Gutin, “Lin–kernighan heuristic adaptations for the generalized traveling salesman problem,” *European Journal of Operational Research*, vol. 208, no. 3, pp. 221–232, 2011.

[22] G. Farin, *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.

[23] O. Porges, R. Lampariello, J. Artigas, A. Wedler, C. Borst, and M. A. Roa, “Reachability and dexterity: Analysis and applications for space robotics,” in *Workshop on Advanced Space Technologies for Robotics and Automation-ASTRA*, 2015.

[24] P. Beson and B. Ames, “Trac-ik: An open-source library for improved solving of generic inverse kinematics,” in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 928–935.

[25] S. Chitta, I. Sucan, and S. Cousins, “Moveit![ros topics],” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.