Adaptive Pixel Art using a Franka Arm*

Daksh Adhar[1], Bhaswanth Ayapilla[1], Sreeharsha Paruchuri[1], Parth Singh[1]

Abstract— **The travelling salesman problem is a classical example of a combinatorial problem and presents itself in various forms in the real-world, from room rearrangement to air traffic control. In this project, we explore its role in autonomously generating pixel art with a Franka robot. To achieve this result, we adopt a stamping mechanism which incrementally generates an image on the whiteboard placed below the robot arm. To generate the grid version of the input image, we develop an algorithm to classify the colour of each region in the desired grid. Moreover, the task demands accurate motion planning, force regulation and reliable stamping mechanism. To achieve efficient task execution, we address planning challenges such as colour-based stamp grouping and minimise arm travel using Travelling Salesman Problem (TSP) optimisation, specifically employing Christofides' algorithm. The system's variability is explored across three dimensions: input patterns, stamping grid resolution (demonstrated with 4x4 and 8x8 grids), and environmental adaptability to changes in ink pad locations. This work highlights the potential of robotic manipulators in executing intricate artistic applications with adaptability and precision.**

I. INTRODUCTION

Robotic arms have become increasingly adept at performing fine motor tasks traditionally reserved for human operators. Among these, artistic applications—such as painting, calligraphy, and stamping—pose unique challenges due to the need for precise force control, repeatability, and interaction with deformable surfaces.

Our work introduces the use of a robotic manipulator to create pixel art autonomously through a stamping mechanism. The aim is to demonstrate the ability of manipulators to perform precise and repeatable artistic tasks through visuo-motor control. We utilize the arm to pick up stamps, apply ink from a stamp pad, and imprint patterns onto a flat whiteboard to generate pixelated images. This process requires accurate motion planning, force regulation, and sequential stamping to recreate digital designs faithfully. Additionally, we are integrating a vision system to ensure the precise placement of each stamp and to compensate for variations in ink transfer.

To ensure efficient task execution, we address key planning challenges such as grouping stamps by color, minimizing arm travel using Traveling Salesman Problem optimization, and managing ink replenishment during stamping. To solve the TSP efficiently, we use Christofides' algorithm [4], which offers better tour quality guarantees compared to simpler heuristics like Nearest Neighbor or Greedy Insertion[5].

We are exploring the variability of the system in three key aspects:

- **Input Variability:** The robot can be tasked with stamping different patterns and coloring various articles, allowing for a diverse range of artistic outputs.
- **Grid Resolution:** The resolution of the stamping grid can be modified, enabling finer details in pixel art. For instance, our demo uses an 8×8 grid, and can be scaled up to higher grid dimensions as well.
- **Environmental Adaptability:** The system can adjust to changes in its environment, such as variations in ink pad locations, ensuring robustness in dynamic setups.

II. RELATED WORK

In this section, we provide an overview of related work on robotic painting technology and vision-guided painting.

Research from Carnegie Mellon University's Robotics Institute demonstrates how their system, Framework and Robotics Initiative for Developing Arts (FRIDA), uses AI to prepare images for robotic painting [1]. While not specifically focused on pixel art, the methodology provides valuable insights into image processing for robotic art production. The system can accept text prompts or image examples as input, then generates appropriate painting.

A related study on vision-based pen-and-ink drawing by a robotic manipulator provides a framework where the system first extracts the outlines from an image and delineates them with different width strokes according to their structural importance [2]. This methodology could be adapted for pixel art by modifying the extraction algorithm to generate pixel-appropriate grid patterns rather than continuous lines.

Complementing AI-driven techniques, 3D vision technology enables highly precise robotic painting. Several research teams have developed systems for automating art creation on canvas using 3D vision-guided robots. These systems can provide 1:1 replicas of original paintings while ensuring complete precision and consistency [3]. A notable implementation involves a collaborative ABB robot equipped with

*This work was supported by MRSD and Robotics Education Laboratory at Carnegie Mellon University

[1]Author is a Masters student in Robotic Systems Development at Robotics Institute, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA
{dadhar, bayapill, sparuchu, parthsin}@andrew.cmu.edu

a Photoneo MotionCam-3D Color camera to create precise replicas of original artworks through a structured workflow:

- Digitization and processing of the reference image
- Vision-guided scanning of both color palette and canvas
- 3D spatial mapping of the working environment
- Precise execution of the painting process

The system's 3D vision capabilities enable the robot to understand depth relationships and surface characteristics, allowing for more natural brush movements and paint application techniques.

III. METHODOLOGY

The forward process begins with a predefined image which can unambiguously be down-sampled into an 8x8 grid of pixel colours that fall into one of the five predefined categories - Red, Green, Blue, Black and White through a simple python script. The output of the aforementioned module consists of a list of 64 grid locations and corresponding colours. This list is then fed to the planning algorithm which optimally sequences the inpainting of grid locations in the robot base frame with the assigned colour in order to optimize for time taken. Moreover, this sequence is augmented with a 'stamp re-ink' waypoint after every step in order to re-coat the stamp pad with ink. This process has been experimentally verified to ensure proper coloring on the whiteboard without the ink density of the coating on the stamp reducing leading to a visually faded colour. Next, this sequence is then passed on to the motion planning block, which executes the plan by moving the end effector to each of the pixel locations as well as picking up the stamp and re-inking it.

It is important to ensure that the end-effector remains perpendicular to the stamping surface to ensure that the square shape is properly imprinted and the area is filled in a consistent manner. In order to achieve this, once every 9 stamps, the planner signals to return the stamp to its initial location and reset all joint values. This sends the manipulator to its predefined 'home' position which is useful to us as it orients the joint angles in such a way that the end-effector is perpendicular to the stamping surface below it. Once the robot executes this step, it resumes stamping according to the generated plan. This sums up the ability of the robot to autonomously make decisions regarding sequencing the waypoints derived from the input. Variability is introduced through ArUco tags placed on the stamp pads, which are detected by the overhead world-frame camera. The camera identifies the position of each pad and provides this information to the planner, allowing it to determine which color to start with or select next.

The reset process involves returning the stamps to the initial, hard-coded, locations and the robot arm retracting to a home location well above the generated pixel-art image.

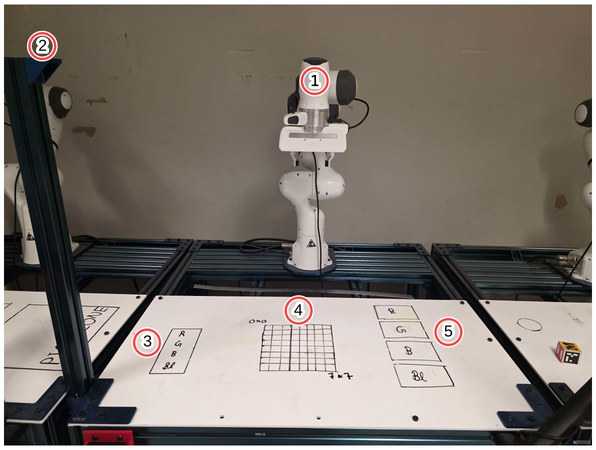

Fig. 1. Hardware Setup with Marked Placeholders

A. Hardware Setup

The hardware setup for this project is shown in figure 1 and the annotated components are labelled in table I. Our setup involves a Franka Panda robot arm which serves the role of the artist for painting the pixel art and the pixel grid, where the artwork is created, is positioned at the center of the arm's workspace, that is, directly below the end-effector. To paint the 8×8 pixel grid on the whiteboard surface, we have replaced the paint brush with 4 stamp rubbers (red, blue, green and black) which are attached to a custom 3-D printed holder that the manipulator can grab onto via the Franka Hand to perform the desired stamp operations. To refill the stamps with ink, on the right side of the workspace as seen in the figure 1, there are 4 ink pads corresponding to the stamp colors. The ink pads can be repositioned freely, provided they remain within the virtual wall constraints defined by FrankaPy. ArUco tags affixed to the pads are used to detect their locations, which are then supplied to the planner as input.

TABLE I
HARDWARE SETUP COMPONENTS

S.No.	Component
1	Franka Emika Panda Manipulator
2	Intel Realsense D435i Camera
3	4 Different Coloured Stamps
4	8x8 Pixel Grid
5	4 Different Colored Ink Pads

B. Input Processing Subsystem

Our input processing subsystem reads a .png file as an input which it converts into an n × n grid of colours where 'n' is a user-defined input. We define it to be 8 for our task as this value respects the virtual walls of the Franka Arm and runs within a time limit that's suitable for demonstration. Each grid cell has properties that define its location in the 3D world, as defined by the robot base-frame, and a capital letter denoting the classified colour of the cell.

So, from an d × d × 3 RGB image we go to an n × n × 4 grid where the 4 dimensions are denoted by X,Y,Z,C.

The classification process operates in the HSV color space, using predefined color ranges to ensure robustness against lighting variations. Each grid cell is mapped to real-world spatial coordinates using a scaling factor that converts pixel measurements into metric units. A constant Z-height is maintained across all grid locations to preserve a uniform reference plane, we set its value to 2cm through experimental validation. The intuition behind this value is grounded in the fact that an impedance controller will try to go to a desired location and will exert force into the environment in order to reach the waypoint. Setting the Z value to 2 centimeters allows for the stamps to be pressed into the stamping surface with adequate enough force to transfer ink from the stamp onto the stamping surface. Once we have computed the X,Y displacement of one of the corners of the grid cells in the robot base frame as a simple, the center of each grid cell can be computed in the robot base frame as a simple addition of the grid dimension in X and Y. This displacement was computed using the *guide_mode.py* script - moving the end effector to the centroid of closest corner grid cell to the robot base.

To enable processing by the Task Planning subsystem, the output of this script is saved as a `.npy` binary file that preserves the grid structure of the intended output.

C. Task Planner

The task planner defines the high-level sequence of actions required to complete the pixel art stamping efficiently. After the input processing subsystem provides the set of pixel locations along with their target colors, the task planner groups the pixels by color and determines the order in which the colors are processed. To decide the color ordering, the robot computes the Euclidean distance from its predefined home pose to the corresponding stamp pad positions. The positions of the stamp pads are tracked using ArUco tags, allowing them to be placed arbitrarily within the workspace while still enabling efficient planning. The colors are then sorted based on ascending distance to reduce unnecessary movements across the workspace.

For each color, the robot performs the following sequence: it first picks up the corresponding stamp tool from the stamp rack, dabs it onto the ink pad to load ink, and then proceeds to stamp all pixels of that color. To minimize the total travel distance when stamping pixels of the same color, we pose the problem as a Traveling Salesman Problem (TSP). Given that finding the exact TSP solution is NP-hard [6] and computationally expensive, we apply Christofides' algorithm [4], a polynomial-time approximation method that guarantees a solution within 1.5 times the optimal tour length. Christofides' algorithm works by first constructing a minimum spanning tree over the pixel locations and then connecting them to form a near-optimal cycle. We use the Christofides implementation from the `NetworkX` library to generate this approximate tour efficiently at runtime.

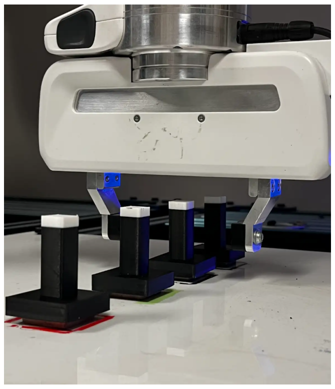
Fig. 2. Robot Gripping the Stamps

Algorithm 1 Traveling Salesman

```
1:  procedure TravelingSalesman(startPosition, pixels)
2:      Initialize graph $G$
3:      $points \leftarrow$ extract $(x, y, z)$ from each pixel
4:      for each point $i$ in $points$ do
5:          Add node $i$ to $G$
6:      end for
7:      for each pair $(i, j)$ do
8:          Add edge with Euclidean distance as weight
9:      end for
10:     $startIdx \leftarrow$ index of closest point to $startPosition$
11:     $tspCycle \leftarrow$ Christofides(G)
12:     Reorder $tspCycle$ to start from $startIdx$
13:     return pixels ordered by $tspCycle$
14: end procedure
```

After completing the stamping of one color, the robot places the stamp back in the stamp area, returns to the home position and transitions to the next color, repeating the procedure until the entire pixel art is completed.

D. Motion Planner

The motion planner is responsible for converting the high-level sequence of the task planner into precise robotic movements. After the color to be stamped is decided by the task planner, the local control functions such as the `move_to_pad()`, to guide the robot to ink pads, `move_to_stamp()`, the location of stamps, `move_to_pixel()`, pixel locations, and `,` `return_to_stamp()`, back to stamps, are implemented using the FrankaPy library and ROS. These functions leverage position and force control of the Panda robot. Each movement involves calculating approach and retract poses using `RigidTransform` objects. For example, as seen in the video, for one stamp motion the robot goes to a pre-grasp pose which is 5cm above the actual stamp pose, using position control. It then moves linearly down the z-axis to the stamp and picks it up by closing the gripper, this motion is achieved by impedance control. Subsequent actions—such as dabbing the stamp onto the ink pad and pressing it onto the paper grid—are similarly split into a pre-grasp approach followed by a linear motion, both executed using force control. Force control is used consistently during all stamping interactions involving contact with the ink pad or paper to ensure uniform and complete stamping results.

Force-controlled stamping is achieved where the end effector applies downward pressure with specific cartesian impedance settings:

$$cartesian\_impedances = [2000, 2000, 1000, 100, 100, 100]$$

This ensures that we achieve stable interactions with deformable surfaces like the stamps and ink pads. The script orchestrates the overall execution by initializing the robot's home pose and loading pixel goals from an input image.

IV. Evaluation

To evaluate the performance and effectiveness of our pixel art system, we ran a few experiments focused on how accurate, effective and robust the stamping process is. These tests were designed to see how close we could get to the intended design, the overall time taken, and how well the system handled different setups.

A. Output Quality

We tested the system on several input images, from very simple colored sequences of low resolution such as `224 x 224` to more complex complete image structures of higher resolution such as `1200 x 1200`, such as the Minecraft Creeper as shown in Fig 3. In all cases, the robot was able to stamp very close to the center of each grid cell, with only a few minor deviations, usually no more than 3mm off. The colors matched well, though we did notice that after a few stamps, the color intensity faded because of the ink pads drying up.

We also tested the system with both 4×4 and 8×8 grids. The 4×4 version finished in around 20 minutes and looked good for basic patterns. The 8×8 grid took about 54 minutes but gave much finer results and showed that the system can scale up if needed. The added detail made the final pixel artwork look a lot more crisp.

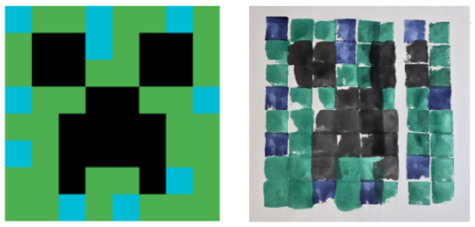

Fig. 3. Input image compared to the the pixel art output

B. Task Planning Efficiency

To see if our planning approach made a difference, we compared our Christofides' TSP-based path to a simple "go row-by-row" strategy. Across multiple runs, Christofides consistently reduced the robot's travel distance between pixel locations by around 38%. This implies lesser time is spent moving around and lesser stress on the Franka arm itself. This motion also looks more seamless with fewer unnecessary zigzags.

C. Re-inking Strategy

Right now, we re-ink the stamp after every three stamping actions. We tested whether that was the right frequency by printing without re-inking for longer stretches. After a few stamps, the ink usually starts to fade, confirming that 3 is a good number. However, this makes the entire process slower and it would be better to stamp after every 5 or even 10 stamps, which would require us to change the stamping material itself. Moreover, some stamp materials hold ink better than the others, so better hardware could help improve efficiency in the future.

D. Repeatability and Robustness

We ran the same stamping task three times in a row to see how consistent the entire system was. The results looked almost identical each time, which was a great sign. The stamp positions stayed accurate and the robot didn't miss any pixels. The only noticeable difference was in the ink intensity, which varied slightly due to drying up.

To further evaluate adaptability, we repositioned the ink pads prior to execution and re-ran the task. The results demonstrated that vision-based tracking significantly enhances the system's robustness and flexibility.

E. Verification

To enhance the reliability of the stamping process, we plan to integrate a verification step using the overhead table-mounted camera. This module will perform post-stamping inspection by analyzing the final artwork pixel by pixel. Each stamped pixel will be evaluated on the basis of the proportion of the grid cell covered with the intended color versus unmarked (white) areas as observed from the viewing angle in 4. A pixel will be considered successfully stamped if the color coverage exceeds a predefined threshold. Otherwise, the pixel will be flagged for re-stamping. The robot will then revisit and re-stamp all flagged locations. This verification

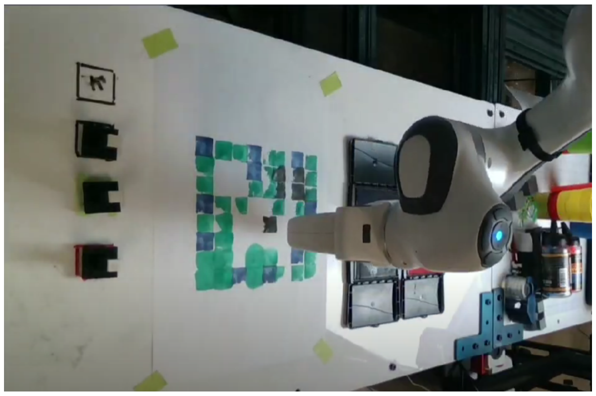

Fig. 4. RealSense camera view

and correction loop will continue until all pixels meet the success criteria. Implementing this step will significantly improve output consistency and reduce the impact of faded ink or minor misalignments over long runs.

V. Challenges

A. Control Methods

Initially, we attempted to use impedance control by specifying target poses slightly beneath the table surface to ensure full contact between the stamp and the board. However, this led to unintended behavior—since the robot could not physically reach the specified pose due to the table, it tilted aggressively in an attempt to comply, compromising the stamp angle and quality.

As a fallback, we adopted a Cartesian control approach, which allowed us to complete the tasks but with limitations. Notably, the end-effector does not maintain a vertical orientation during movement. Once the basic implementation was completed, we shifted to force control methods to trul make our system robust.

B. Hardware

One of the most persistent issues was ink drying. Since the ink pads dry out quickly, especially for less-used colors, frequent re-inking was required to maintain color intensity. Moreover, the rubber stamp material currently in use does not retain ink well across multiple stamps—it typically fades after 1–2 actions. This forced us to re-ink nearly every time, slowing down the overall process. A sponge or velvet based stamping material was then implemnted which increased our re-inking step from one to three.

Another hardware limitation was that the stamp did not lay flat on the surface during contact. This was a result of both imperfect approach trajectories and the rigid nature of the stamp. Consequently, some parts of the stamp did not touch the board fully, causing incomplete prints. We solved this by adding a foam based pad at the ends of the stamp. This meant that a better contact could be established and the ink could print better on paper.

C. Stamping Time

The overall time taken to complete a single pixel art is currently quite high—around 54 minutes for an 8×8 grid. This is due to the combined effects of pixel re-inking, long path traversal, and sequential stamping logic. While our Christofides-based task planning helps reduce travel distance, it does not fully compensate for the overhead caused by hardware limitations. The high runtime also limits how frequently we can test with full setups, especially since ink drying and mechanical fatigue introduce inconsistencies over time.

VI. Future Work

Future versions could work on improving the stamping quality of our current setup. This mostly depends on getting better hardware - specifically stamps that are made of better material. As a result, we would expect to see ink remain on the stamp for more than three 'dab' action and thus we would be able to re-dab at a lower frequency which will bring down the time taken for the process. Additionally, whiteboard inks that are denser in nature can be used which will allow the entire stamp to be coated uniformly with ink, resulting in well-defined square stamps on the whiteboard. Together, the above improvements can make the system more time-efficient.

Moving forward, we can look to incorporate visual feedback into our system. Either for added variability in detecting using Aruco markers where the stamp pads or to perform validation followed by error correction. This would make the system more reliable for all possible output resolutions.

Another promising direction is improving motion planning. While Christofides' algorithm has significantly reduced travel distance, it does not currently optimize for end-effector orientation or avoid contact-induced drift. We plan to explore task-space motion planning with constraints that maintain vertical alignment of the stamp and incorporate trajectory smoothness for faster, more efficient execution.

We can also look into generalizing our system to handle arbitrary image inputs with dynamic grid resolutions and color sets. This would involve automated preprocessing to convert any image into stampable pixel art, as well as dynamic color management and on-the-fly assignment of available inks. Together, these upgrades would make the system fully autonomous and scalable for larger, more complex designs.

Finally, another potential direction can be integrating learning-based optimization for adaptive stamping. By collecting data over multiple stamping runs—including force profiles, stamp quality metrics, and execution times—a learning model could be trained to fine-tune parameters such as approach speed, force thresholds, and re-dab frequency based on real-time context. This would enable the system to autonomously adapt to variations in stamp material wear, ink consistency, or surface conditions, thereby improving performance without manual recalibration. Over time, such a feedback-driven learning loop could lead to more consistent stamp quality and further reductions in execution time.

VII. Conclusion

This work demonstrates the feasibility and versatility of using a Franka Emika Panda robotic arm to autonomously generate pixel art through adaptive stamping. By integrating robust vision-based input processing, efficient task planning using Christofides' algorithm for TSP optimization, and precise force-controlled motion execution, our system achieves consistent and repeatable artistic outputs across varying grid resolutions and environmental configurations. The experimental results highlight both the accuracy of the stamping process and the substantial reduction in robot travel distance compared to naïve planning strategies, validating the effectiveness of our approach.

Despite hardware limitations-such as ink retention and stamp-surface contact-our iterative improvements, including material changes and vision-based adaptability, have enhanced output quality and system robustness. The modular design also allows for straightforward extension to higher grid resolutions, arbitrary input images, and dynamic workspace layouts.

References

[1] Schaldenbrand, Peter, et al. ”Cofrida: Self-supervised fine-tuning for human-robot co-painting.” 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024

[2] Lu, Yan, Josh HM Lam, and Yeung Yam. ”Preliminary study on vision-based pen-and-ink drawing by a robotic manipulator.” 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. IEEE, 2009.

[3] https://www.automate.org/case-studies/automating-art-robotic-painting-on-canvas.

[4] Christofides, Nicos. ”Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem.” *Operations Research Forum*, vol. 3, no. 1, 2022, pp. 20. DOI: 10.1007/s43069-021-00101-z.

[5] Pop, Petrică C., Ovidiu Cosma, Cosmin Sabo, and Corina Pop Sitar. ”A comprehensive survey on the generalized traveling salesman problem.” *European Journal of Operational Research*, vol. 314, no. 3, 2024, pp. 819-835. DOI: https://doi.org/10.1016/j.ejor.2023.07.022

[6] Garey, Michael R., and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979. ISBN: 0716710447.

[7] Sten. ”Robot printer made of LEGO bricks can produce any pixel art using OpenAI's DALL-E 3.” Creative Mindstorms, 2024. [Online]. Available: https://www.designboom.com/technology/robot-printer-lego-bricks-pixel-art-openai-dall-e-3-sten-creative-mindstorms-06-21-2024/

[8] Santos, M., Twu, Y., Egerstedt, M., Diaz-Mercado, Y. ”Interactive Multi-Robot Painting Through Colored Motion Trails.” *Frontiers in Robotics and AI*, vol. 7, 2020, Article 580415. DOI: 10.3389/frobt.2020.580415

[9] Kim, J., Lewis, C., Sturdee, M. ”Encountering Robotic Art: The Social, Material, and Temporal Processes of Creation with Machines.” In *CHI Conference on Human Factors in Computing Systems (CHI '25)*, Yokohama, Japan, 2025. DOI: 10.1145/3706598.3713327

[10] McCormack, J., Gifford, T., Hutchings, P. ”The Robot is Present: Creative Approaches for Artistic Expression with Robots.” *Frontiers in Robotics and AI*, vol. 8, 2021, Article 710919. DOI: 10.3389/frobt.2021.710919