Report for Robotics Final Project Team 12

David Gasser
Technische Universität München

Jochen Luithardt
University of Tübingen

Matthias Pouleau
Technische Universität München

Thea Verburg
VU Amsterdam

***Abstract*—Artificial intelligence and robots have the potential to revolutionize the art world, but do they have the ability to create truly creative artwork? This report explores the possibilities of artificially created art through the creation of a system that is able to take speech input, generate and paint the input on a canvas with brushstrokes. Here we show the difficulties in replicating and generating artwork, along with technical details of our implementation.**

***Index Terms*—robotics, AI, art, painting**

I. Introduction

Are you interested in buying artwork produced by a robot? Why or why not? In this report, we explore the possibilities of artificially created art, our challenges, and our research in the creation of a robot that is able to paint. What components does an artist have? The physical body, the mind, the brain to allow for creating art. Yet this is contained in our research, as DALL-E [1] is an algorithm which has the ability to create art. By providing the algorithm with a physical body, the robot arm, we are providing the algorithm with a physical extension of itself. This system embodies an entire cohesive robot that is able to create art. The robot is able to pick up the paint brush itself, it can wet the paint brush and can select the correct paint and paint a brush stroke on the paper canvas. Therefore, our system should be able to create just as notable art as any human artist, right? Yet, it is important to consider that art is subjective, and it is hard to foresee how a viewer will interpret a piece of art. This brings the question of what defines art as art? It is possible to say that creating art requires a certain level of creativity, which is a quality that may not be fully reproducible in a machine. Notwithstanding that DALL-E is able to generate creative artwork, yet it is not clear whether DALL-E is truly creative in the same sense as a human artist. However, there are some notable benefits to using a robot to create art. One of these is that a robot is able to continuously work without needing breaks or rest, which makes a more efficient production process possible. Additionally, a robot could potentially produce artwork that is more consistent in style and quality, as it is not subject to the same variations in mood or skill level as a human artist. On the whole, a robot creating art raises interesting questions about the nature of creativity and the role of technology in the arts. While there are certainly challenges to overcome, the development of a robot that can create art has the potential to revolutionize the art world and change the way we think about the creative process.

II. Related Work

In [2] a new stroke-based rendering robotic drawing system is introduced, which notably does not require any user assistance. The main distinguishing feature of their work from the existing robotic drawing systems at the time of publishing, is that they created a stroke-based drawing and an accompanying stroke ordering plan within their system. They described previous research on stroke-based rendering (SBR) in their work, and also they note how the use of machine learning (ML) has advanced the field. Ilinkin et al. note that their work builds upon this history of SBR research and aims to further advance the capabilities of robotic drawing systems. Their algorithm works as such, when given an input image I and a desired number of strokes N, their system generates a stroke plan for a robotic painter. Their system had image segmentation and depth estimation algorithms integrated into the system. In this system, during stroke generation, they use AniPainter to generate the strokes. They mention that AniPainter [3] is a variation on the stroke generation algorithm proposed by [4]. This algorithm takes as input a reference image and reproduces it as a drawing by finding sets of brushstrokes, converting them into Bézier curves. Since AniPainter suited our project very well, we used this algorithm in our project, as it was able to generate strokes that were suitable for our system.

In the AniPainter project [5], they also used a robotic arm to paint. While in our project, we did adopt and alter the stroke-generation algorithm from this project. However, when we look at the video's from this project, we see that the robot's arm movement is more erratic and picking up the paint is more volatile. The downwards movement of their arm is less controlled. This is one area we aimed to improve in from this project. In our work, we methodically approached the picking up of the paint brush, putting the brush in water, cleaning the brush and retrieving paint. We wanted to show that it is possible to do this in a more controlled manner and ensure that paint would not spill everywhere.

In the research by [6], they experimented with a 6 DOF robotic arm painting, but they used watercolor instead. One of the main takeaways from this paper was that watercolor required many variables and techniques, like lightness, dilution level and the amount of water in the paint. Therefore, we decided to use Acrylic paint instead, as we thought our results would vary less than this work's research results. Also, acrylic paint meant that the canvas would dry more quickly than watercolor.

III. Methodology

The Robotics HW4 assignment meant it was possible to establish familiarity with the robot, such as possible movements the robotic arm could make. Then a calibration system for the camera was created, in order to establish where the robotic arm is located in the plane. Then multiple approaches were discussed on how best the system should be set up. First, the main focus was to teach the robotic arm to paint. This project is made up of 4 components; the speech recognition, the image generation, translating the image to strokes and the actual painting part.

The goal was to keep the interaction with the robot as natural as possible, so the interaction with the robot should be done via speech (part of Jochen's expertise). To achieve this goal, the robot must be able to understand, interpret, and respond meaningfully to spoken language. Additionally, the robot must recognize the user's character request from the user interaction. Since the Transformer architecture [7] seems to be the most suitable for these tasks, all neural networks used in this work are based on this architecture. To translate an audio signal into text, we used a model called Whisper [8], which is characterized by its high accuracy and processing speed. After translating the user's audio input into text, we use the GPT-3 model from OpenAI [9] to generate a meaningful user interaction. The model was given the context of a Rooter version of Leonardo Da Vinci and instructed to interact with the user in a friendly and helpful manner. Finally, the few-shot capability of GPT-3 was used to extract the user's intended image.

This extracted user intent was then processed into a prompt by adding the string "a black and white paint brush artwork" to the end of the extracted user intent. We use a model called DALL-E [10], which is capable of creating images from text descriptions. After 4 options were created, the user was asked to make a selection.

Next, the selected image needs to be transformed into a painting, which includes a set of instructions for brush strokes which can be given to the robot to perform. As time for the project was limited and designing such an algorithm would take up a lot of time, we decided to use Anipainter [3]. Anipainter is an algorithm which can take in a reference image and replicate it as a drawing by finding sets of brushstrokes, converting them into Bézier curves. The algorithm suited this project so well that only minor adjustments had to be made. The first main obstacle that emerged in this project was translating an image to a set of coordinates. The next challenge was getting the robot to move a paintbrush from one set of coordinates to the next. Then it was also necessary to change the paint brush's colors'. This project progressed systematically, making sure that every component worked correctly and was reproducible, before moving to the next.

IV. Experimental Setup

A. Speech Component

To perform speech recognition, audio input was first recorded and fed locally into the locally deployed Whisper model. Then, the user input was converted into a prompt and sent to the OpenAI GPT-3 API. Then the robot's response was translated into audio output using a text-to-speech engine. After this initial interaction, the robot extracted the type of art that the user requested. Afterward, this request was converted into a prompt. While the user interacts with the robot through a dialog, the robot sends this request to the OpenAI DALL-E API to create four different images. Finally, the user decided which artwork to have the robot draw.

B. Image and Stroke Generation

First, the image is discretized to several colors. Then, the starting point for a stroke is determined by finding the greatest difference between the reference image and the painting. From this starting point, new points of the same color are searched for. Once all points are found or a limit is reached, AniPainter applies PCA in order to transform the points into a Bézier curve. A stroke can still be rejected if it does not decrease the error between painting and image enough. Strokes are made until a prior specified maximum is reached, or no strokes are found to further decrease the error. Multiple stroke sizes can be used by layering their strokes on top of each other. To simplify paint mixing, it was decided to paint in black and white. Therefore, the pictures were always discretized to seven equally distributed colors between black and white, including the latter two. Moreover, the option to save the strokes in a CSV file as soon as they are created was implemented to enable the stroke generation script and the robot painting script in parallel. However, it was decided to not take this approach, as the stroke generation for the test images was fast enough. The brush strokes are Bézier curves given to the script controlling the robot arm in the form of a CSV file holding the x- and y-coordinates of a chosen number of points distributed equidistantly on the curve. Furthermore, the coordinates are in their pixel frame, ranging between values of 0 and 200. For every Bézier curve the brush size, or radius, as well as the color are given as integers and appended to the coordinates of the curve. Each stroke consists of a set of points to be approached by the end effector. Once these points have been determined, the image-coordinates have to be converted into robot-base coordinates. The points in robot-base coordinates are all fixed for four out of six dimensions: The orientation of the end-effector is always (0°, -180°, 135°) for the three rotation axes and the value of z is always the drawing height. To get the x and y values, you have to perform a 2D transformation, using a transformation matrix M.

The Matrix M can be found, by comparing the robot pose from 6 different known image positions. The poses of the points in image coordinates and robot-base coordinates can be assembled into the matrices P_img and P respectively, with each point being a line vector. Our image point coordinates are in the form P_img[i] = $\begin{pmatrix} x_img(i) & y_img(i) & 1 \end{pmatrix}$ for i = {1, .., 6} and the robot-base coordinates are in the form P[i] = [x_i y_i] for I = {1, .., 6}. This gives us the following linear equation:

By using a linear solver, we got $M = \begin{pmatrix} -0.5482 & -0.5752 \\ 0.5826 & -0.5559 \\ 225.322 & 288.739 \end{pmatrix}$

The final form of the pose is now the following: (x, y, drawing_height, -180°, 0°, 45°)

C. Navigation and Routing

As the robot should be able to handle a case with different brush sizes, all brushes were placed onto a self-built brush-holder. When the robot needs to use a brush, it must be able to pick up the brush from this holder. For this, the robot first needs to move to an approach point above the pick-up pose and then move linearly to the pick-up pose. Once the brush is grabbed, the end-effector moves to the preposition, which is shifted by 50 mm on the x- and y-axis. Once the robot is in that position, it can move again freely in PTP-mode, without risking colliding with the brush-holder.

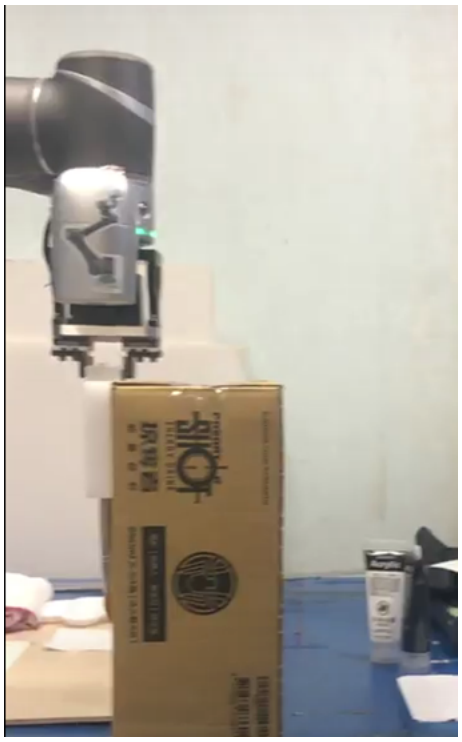

Fig. 1. Robot Gripper grabbing the brush from the brush-holder.

The processes of cleaning and then drying the paintbrush must be done each time after the robot grabbed a brush, when the color is changed and when the robot is done painting. To clean the brush, the end of the brush must touch the ground of the water container, so that the paint gets out more easily. Once the robot is in that position, it carries out a rotational motion for several loops, to increase the friction with the water and the ground, so that the paint gets off the brush. When the brush is clean, the robot moves the brush out of the water. While it is still in the container, it performs another rotational motion, to have most of the water fall off the brush, in order to avoid having water fall on the nearby color palette. After this, the robot moves the brush to the drying cloth and performs several linear movements on it, with a random starting position, to dry the brush.

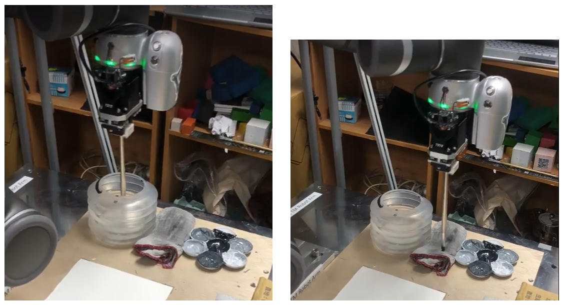

Fig. 2. Paintbrush getting cleaned and dried

The robot needs to get new paint on the brush, when the color changes between two strokes or after three consecutive strokes of the same color, as the brush would be out of color. In order to limit the amount of color changes, the strokes of the same color and width got grouped together in the CSV-file containing all generated strokes. To get the paint, we had to teach 7 points, for each center of the color areas. In order to reduce the repetition error, when rebuilding the robot setup, each point is defined relative to the center of the color area 0 (black). That way, if there is some imprecision after a rebuild, only the center of area 0 had to be retaught. To get the color, the robot goes to the center of the area and then performs a rotational motion around it, so the robot doesn't always get the paint from the same spot.

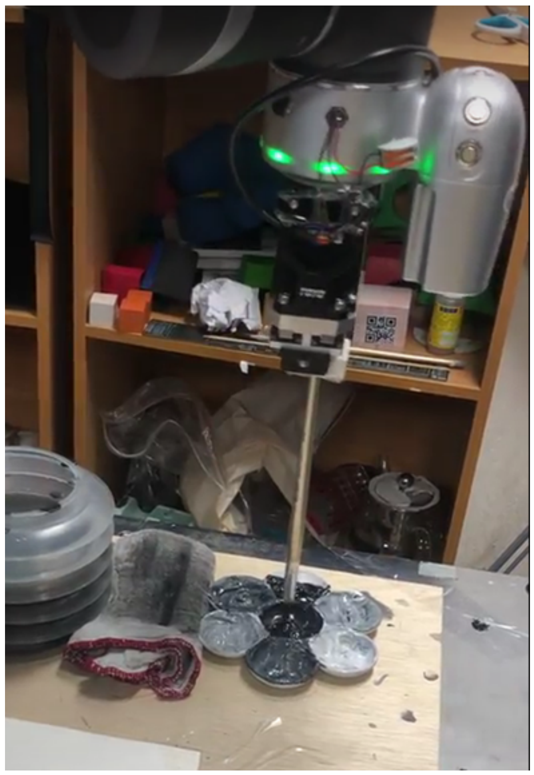

Fig. 3. Getting painting color from the color palette.

1) Painting: Painting the strokes was by far the biggest issue we encountered with our project. Before working with colors and water, we performed many dry runs to make sure that there are no bugs in our code and that the behavior of the robot does not change over time. The robustness of the code was of the utmost importance to us, as irregular movements when working with color and water could entail damage to the robot or the surrounding computers. During these dry runs, we closely followed the movement of the end effector while simulating the brush strokes and were able to see that the end effector did all movements as instructed. The robot was able to follow the Bézier curve in an acceptable time once the number of points was decreased to 20. During the first few wet runs, a major problem became apparent quite quickly. Due to the friction between brush and paper, all brushes did not follow a given Bézier curve correctly. We found that this can be attributed to two factors.

1) The hairs of the brush are pulled to one side and aren't directly underneath the brush as soon as the brush is being moved in one direction.
2) The approach of fixing the end of the paint brush to erasers for the robot to grip them turned out to be too flexible.

The first factor is an inherent characteristic of brushes. Furthermore, the second factor, the flexibility, seemed like an important fail-safe to avoid damaging materials or machinery. Therefore, ultimately it was decided to not address these factors, but rather take them as a given. As friction was found to be the main cause of the problem, multiple strokes on different drawing heights were compared. It became obvious that the lower the friction, the better the brush stroke. However, as this did not alleviate the problem completely and only thin lines were able to be drawn that way, we decided to draw each line again starting from the other side. This approach showed major improvements, as drawing the full curve and multiple brush sizes was now possible. Yet, solving this problem entailed the next issue. Moving the brush in one direction slightly deformed the brush head so that the tip was behind the base of the hairs. When running the line from the other side, the brush tip was still pointed in the direction of movement, creating an unclean start of the line. To redirect the brush tip in the opposite direction at the very beginning of the line requires fine motor skills even for humans. After further testing, a sloping path for the z-coordinates was implemented. With this, the brush comes from an extended line between the last two points and only lightly touches the paper for the first few coordinates. Consequently, the brush tip reverses its directions without creating an unclean line. Figure 4 and 5 illustrate this technique.

2) Switching to different stroke size: The program and the setup were initially made to be able to use several different brushes, but in the end, only one of the three brushes was fit for the task, as the other two didn't behave the same way in all directions. To still be able to change the stroke size, we only had to modify the height of the brush, when painting, as

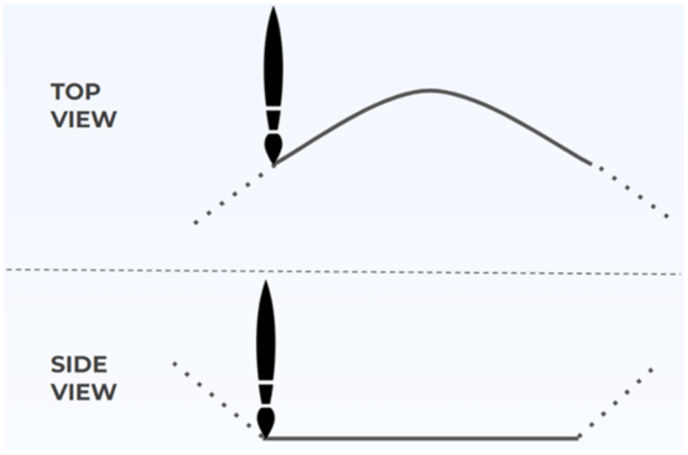

Fig. 4. Paint brush perspectives

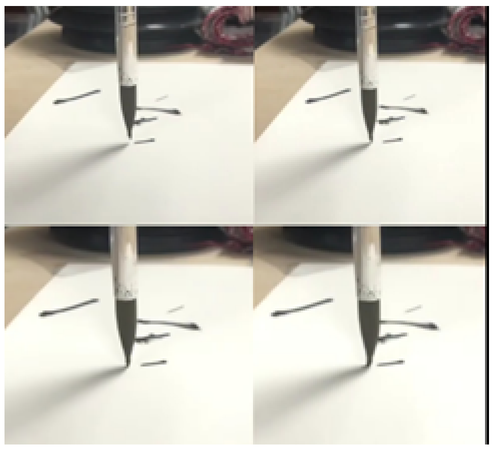

Fig. 5. Movement when the paint brush hits the canvas

the stroke would get larger, when the brush goes lower. In the end, we were able to have two stroke-sizes, 3mm and 5mm. The brush needed to be 5mm lower when drawing a 5mm thick stroke, than for a 3mm stroke.

V. Results

In the end, given the final code and setup created, it should be possible for our robot to paint a painting with a given set of strokes, from a voice command. Unfortunately, there was limited amount of time with the robotic arm, which made it impossible to carry this out completely, as the painting process would take 2–4 hours to finish. This duration is not unusual, as a human would also take a similar time to paint a painting with hundreds of strokes. In our test run, the robot painting for a bit more than one hour and had approximately one third of all strokes drawn. During this time, the robot did not encounter any setbacks.

The first problem that presented itself was that it was necessary to construct and dismantle the setup each time. This meant many accuracy issues at the start, and it meant losing a lot of time. In the end, the program was as independent as possible from the actual accuracy of the dimensions of the setup. Another challenge was translating the stroke itinerary into a fitting robot motion, because of the flexibility of the brush tip. It meant adding several improvements to our base motion, in order to make it work. Since it takes a long time for the robot to paint a full painting, this made it harder to test our program, therefore it was necessary to decompose it into smaller sub steps. This was exacerbated by the fact that even a highly simplified image, as our test run, takes around 500 – 1000 strokes to complete.

Ultimately, due to a lack of time, it was not possible to paint a complete painting with the complexity we were hoping for. Since the code is working, a test run of a more complex painting with over 1000 strokes, would be a good point for future work to continue on. However, the problem with this is that the robot would need to draw for several consecutive hours.

VI. Conclusion

To conclude, even with a limited time span, we were able to show that our system, could recognize a speech command to generate an image, and start to paint this image. With the capability of AI and a robotic arm, we proved that a robot can create art. Sadly, it was not possible to obtain a fully finished painting with the goal complexity within the limited amount of time our team had with the robot. In the future, when given more time, certainly the system could create a fully realized painting. This has the potential to revolutionize the way that we think about art and the creative process, and could lead to new and exciting developments in the field of robotics. Furthermore, it can be argued that more research should be done in this area of research, especially to note the role of technology in the arts.

VII. Work Division in Detail

David

- Finding an algorithm to convert an image into a set of brush strokes.
- Found Anipainter, figured out how it works and how it had to be adjusted for our purposes. Made the adjustments.
- Experimented with different painting prompts for DALL-E and read into the basics of certain drawing styles to understand which image generation prompts work best for the conversion into brush strokes.
- Experimented with Anipainter parameters to find the best options for stroke generation
- In general, the team leader.

Jochen

- Building voice interface, speech recognition and dialog interaction
- Building a react frontend for dialog interaction and image selection
- Intent detection with GPT-3
- Creating a framework code for picking up, cleaning and drying the pencil as well as picking the right color.
- Experimenting with DALL-E to create meaningful prompts.
- Experimenting with GPT-3 to create a meaningful dialog interaction.

Matthias

- Teaching of points for cleaning pencil, grabbing pencil and getting color.
- Getting Transformation from image to robot-base coordinates.
- Helping David and Jochen to implement the proper end-effector movement in order to get a fitting stroke on the sheet.
- Working with the robot, resetting it, putting it in a certain position etc. while the algorithms got adjusted for the next run.

Thea

- Unfortunately, I have less knowledge than my teammates, due to being the only Bachelor student in my team. This meant that I cannot always contribute as much as the others on certain topics.
- Lead charge of presentations and research where relevant.
- All-round contribution, and I just jumped in where I could.
- Getting equipment, finding paint brushes.
- Taking measurements.
- Preparing the paint brushes with glue and erasers for the pick-up from the robotic arm.
- Finding relevant other projects for methodology.
- Gave feedback, help to set up and clean up.
- Made sure we had 6 distinct gray tones of paint, from light to dark.

References

[1] OpenAI. Dall-e: Creating images from text, Jun 2022.

[2] Ivaylo Ilinkin, Daeun Song, and Young J Kim. Stroke-based rendering and planning for robotic performance of artistic drawing. *arXiv preprint arXiv:2210.07590*, 2022.

[3] P. Schaldenbrand. pschaldenbrand/anipainter: Animate a given video by using a robot to paint each frame. frames are converted into brush strokes using a novel stroke-based rendering algorithm. — github.com. https://github.com/pschaldenbrand/AniPainter, 2020. [Accessed 26-Dec-2022].

[4] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460, 1998.

[5] Anipainter.

[6] Lorenzo Scalera, Stefano Seriani, Alessandro Gasparetto, and Paolo Gallina. Watercolour robotic painting: a novel automatic system for artistic rendering. *Journal of Intelligent & Robotic Systems*, 95(3):871–886, 2019.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[8] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.