JOURNAL OF APPLIED INTELLIGENT SYSTEMS & INFORMATION SCIENCES
VOL. 3, PP. 36-45, OCTOBER 2022.
Available at: www.journal.research.fanap.com
DOI: https://doi.org/10.22034/JAISIS.2022.362813.1048

AN EFFICIENT PATH PLANNING METHOD BASED ON THE CURVATURE OF CONTOURS FOR DRAWING ROBOTS

Majid Abedinzadeh Shahri[1,*] and Seyed Vahid Daei Niaki[1]

[1]*Research Center, FANAP Co., Tehran, Iran,*

ABSTRACT

Creating artistic robotic systems is among the most attractive activities that have recently become more interesting. For drawing robots, a contour graphic can be presented as a Raster graphic or a Vector graphic. Because vector graphics can create continuous and smooth strokes, vector graphics are more appropriate than raster graphics for drawing robots. Accordingly, in this paper, we propose a methodology for the smooth path planning of drawing robots. To achieve this purpose, in this work, firstly, we propose an approach for sorting the points of a raster graphic into several strokes. Secondly, a novel approach is proposed to identify the main points of the raster graphic and use them for Raster-To-Vector conversion. We use the obtained vector graphics for path planning. Finally, the obtained trajectories are compared to each other on a simulated drawing robot to show how a Raster-To-Vector approach affects the performance of a drawing robot. The results validate the applicability of the proposed approach.

Keywords: Drawing Robot, Path Planning Methods, Counter Drawing, Raster to Vector Conversion.

1. INTRODUCTION

Writing and drawing easily performed by humans, are two key activities that make robots look more like humans. Therefore, robotic drawing is becoming a popular human-robot interactive activity that is both attractive and fun for the public people (Wang et al., 2020; Gao, et al., 2020). The early history of the creation of drawing machines can be attributed to the works of art by Jean Tinguely and Harold Cohens Aron (McCorduck, 1991).

Generally, a robotic drawing system is usually a special machine that creates pieces of artwork either completely or partially (Coelho, A., 2018; Adamik et al., 2022). In a robotic drawing system, a robotic arm (Song, Lee, and Kim, 2018) or a humanoid robot (Calinon, Epiney, & Billard, 2005) is used in an interactive environment to draw a picture of images in front of human users. Due to the interactivity and entertainment, drawing systems applications have been extended in a wide range of scenarios such as children's education (Hood, Lemaignan, & Dillenbourg, 2015), psychological therapy (Cooney and Menezes, 2018), and social entertainment (Jean-Pierre & Sa¨ıd, 2012). Although these robotic systems may have been independent and autonomous components and their output may be similar to the works of human artists, developing a creative machine is still an open problem (Jeon, 2017; Yu & Chen, 2018).

Researchers have spent much effort in developing artistic robots that can draw sketch portraits (Gao et al., 2019), color images (Luo, Hong, & Chung, 2016), watercolors (Scalera et al., 2019), painting using the palette knife technique (Beltramello, 2020), etc. While It usually takes a robot several hours to draw color images (Scalera et al., 2019), drawing a portrait which mainly includes a set of lines usually takes a short period. Portrait drawing

* *Corresponding Author, Email:* m.abedinzadeh@ut.ac.ir

RECEIVED: 19 SEPTEMBER 2022; ACCEPTED: 06 OCTOBER 2022; PUBLISHED ONLINE: 10 OCTOBER 2022

robots thus allow active interactions between robots and users. In this work, in an attempt to have a portrait drawn in a short time, we focus only on drawing contours.

For a robot to draw the contours of a picture, the continuous contours of the desired picture need to be presented as digital data to be received by the robot. This digitalization process usually adds noise to the contours' features. On the other hand, after the robot planned its motion according to the received digital data, the robot should draw the contours as continuous lines on a sheet.

Nevertheless, a digitalized graphic can be presented in two frameworks (Orzan et al., 2008): raster graphics and vector graphics. For the raster graphics, the continuous lines of contours are presented as discrete points (Hsu, 2017). For a drawing robot, this presentation results in repetitive go-and-stop motions that would act jerky. On the other hand, vector graphics present lines with continuous mathematics formulas. Because vector graphics can create continuous and smooth strokes, for a drawing robot, it is preferred to receive a digitalized vector graphic (instead of a digitalized raster graphic).

However, usually, a digitalized graphic is received as a raster graphic. Accordingly, we need an algorithm to convert a raster graphic to a vector one. In computer science, this process is called Raster-To-Vector conversion (Levachkine, 2003; Liu, 2017). Hence, this paper focuses on algorithm design for Raster-To-Vector conversion.

Nevertheless, as mentioned before, one of the main applications of drawing robots is entertainment. In other words, this type of robot (drawing robots) is very popular among toy designers. For a robotic toy to have more lifetime, energy efficiency is essential. Indeed, low-power design is a major desired feature for a drawing robot toy. Now, the question raises here is how a Raster-To-Vector approach affects the efficiency of a drawing robot.

To address this challenge, in this paper, firstly, we propose an approach for sorting the points of a raster graphic into several strokes. Secondly, as the main contribution of this work, a novel approach is proposed to identify the main points of the raster graphic. In this approach, we consider the curvature of contours as a guideline to exclude the points with less importance. The remained points are used for Raster-To-Vector conversion and path planning. Finally, the proposed approach is compared to a conventional one on a simulated robot to show how a Raster-To-Vector approach affects the performance of a drawing robot.

The rest of the paper is structured as follows; Section 2 states the problem of interest and presents the different approaches for path planning. Then, in Section 3, we introduce the case study in the simulated environment. Also, in this section, we present the obtained simulation results. The paper ends with discussions and conclusions in Section 4 and Section 5, respectively.

2. Problem Statement & Path Planning Methods

For drawing counters of a picture, consider a robot that received the raster contours' points. The problem addressed here is to design an efficient path planning algorithm for the robot's motion to obtain the trajectories of desired contours. Without loss of generality, we assume the desired contours points are presented in an M×M matrix (hereinafter this matrix is named *ConMat*) as illustrated in Fig. 1. In this figure, the filled cells present the contours' points.

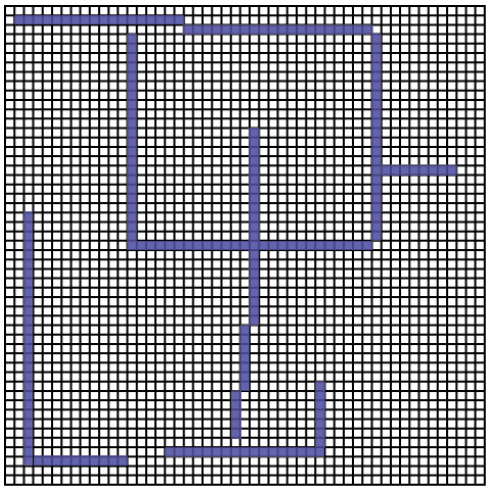


Fig.1: An example of contours' points on a matrix. In this figure, the cells related to the points are filled.

Generally, for beginning to draw the contours with a start point, we need an approach for sorting the contours' points. Accordingly, in the following, we introduce an approach to order the points in several strokes (each stroke includes several ordered points) to form the desired contours.

2.1. Sorting method

For this purpose, we compute a neighborhood matrix for each cell of *ConMat*. Each neighborhood matrix has 9 cells (3×3) and the central cell of each neighborhood matrix is located on a cell of *ConMat*. Accordingly, if each cell of the neighborhood matrix is located on a filled cell of *ConMat*, that cell takes value 1 (otherwise, it takes value 0), see Fig. 2. Computing these neighborhood matrices for all cells of *ConMat*, the neighborhood number of each cell is obtained by the sum of the values of its neighborhood matrix.

According to the neighborhood numbers, the contours' points are sorted and divided into several strokes. For this, the filled cell with a minimum neighborhood number is chosen as the first point of the first stroke. Thereafter, if that cell has at least an adjacent filled cell (according to the neighborhood matrix of the chosen cell), the adjacent cell with the maximum neighborhood number is chosen as the next point of that stroke. This procedure is repeated until the last chosen cell has no adjacent filled cell. In this case, if there exists at least a filled cell among the other cells, the presented procedure is repeated for the next stroke. Accordingly, the points of strokes are obtained (according to the location of chosen cells). To better understand this approach, see **Algorithm I** in Table 1.

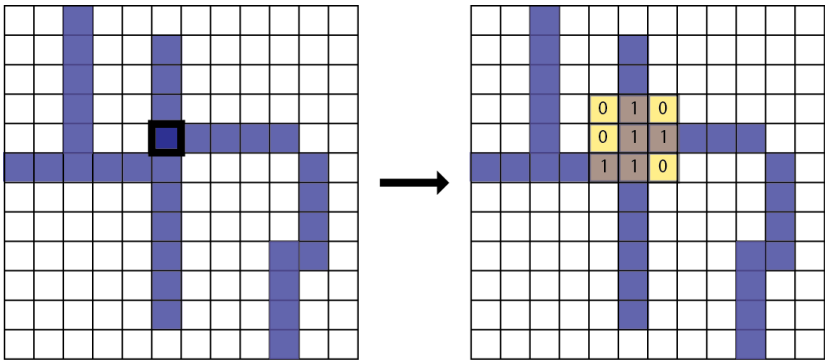


Fig. 2: An example of a neighborhood matrix: The selected cell for computing the neighborhood matrix is highlighted in the left subfigure. Also, the related neighborhood matrix is computed in the right subfigure.

Table 1: Sorting Contours' Points Algorithm

The algorithm I: Sorting Contours' Points
Input: initial *ConMat* according to the given raster points
Output: The strokes of the contours presented in *ConMat*
Compute the neighborhood matrices for all cells in *ConMat*;
While (exists at least a filled cell in *ConMat*)
Compute the neighborhood numbers for all cells in *ConMat*;
Create a new stroke and open that;
Choose the filled cell with a minimum neighborhood number;
Insert the location of the chosen cell in the stroke and make that cell blank in *ConMat*;
While (the last chosen cell has at least an adjacent filled cell)
Choose the adjacent cell with the maximum neighborhood number;
Insert the location of the chosen cell in the stroke and make that cell blank in *ConMat*;
Update the neighborhood matrices for all cells in *ConMat*;
End While
Close the stroke;
End While

For each obtained stroke, we need a path function to draw that. However, raster graphics (due to the digitalization process) may include noisy points. Hence, considering all raster points for path planning may be an inefficient solution. In the following, we introduce an approach to overcome this problem.

2.2. Separating method

Because the contours' points include noisy data, we propose an approach to remove the points which are less important for the path planning procedure. For this purpose, we introduce the stroke curvature as a criterion for labeling P as an important one or not: $C(.) = StrokeCurvature(.)$. For this, C is computed numerically for all points of each stroke. Thereafter, if for a point (e.g., P) we have: $C(P) > \mu$, that point is labeled as an important one. Indeed, if the chosen point has a large curvature, that is labeled as an important one. Note that, if μ is chosen with a small value, most of the points are considered important points, and on the contrary, if one chooses a large value for this parameter, most of the points are labeled as less important.

Once the main raster points are obtained, we use them for path planning. In the following, we introduce three approaches for Raster-To-Vector conversion and path planning.

2.3. Path planning methods

Now, we need an approach to present the points of each stroke (see Fig. 3a as a sample) in mathematical forms. Indeed, for a stroke (e.g., the i^{th} stroke), we need a path function as: $P_i = f_i(t)$, that converts the drawing time (notated as t) to the stroke position (i.e., P_i). $P_i = [P_i(1)\ P_i(2)]$ is a vector with two components (to present the point location on the X and Y axes). To achieve this purpose, we introduce three interpolation approaches to convert raster points to vector presentation.

1- *Dense piecewise*: Although we proposed an approach to remove the points with less importance, conventionally, all points are considered for interpolation. Hence, as a conventional method, we connect every two adjacent points of each stroke with two piecewise linear functions (one for the X and the other for the Y axes). Also, to control the drawing speed of the stroke, we consider the length of the traced path (for the j^{th} point of the i^{th} stroke notated as $s_{i,j}$) as the input of piecewise linear functions:

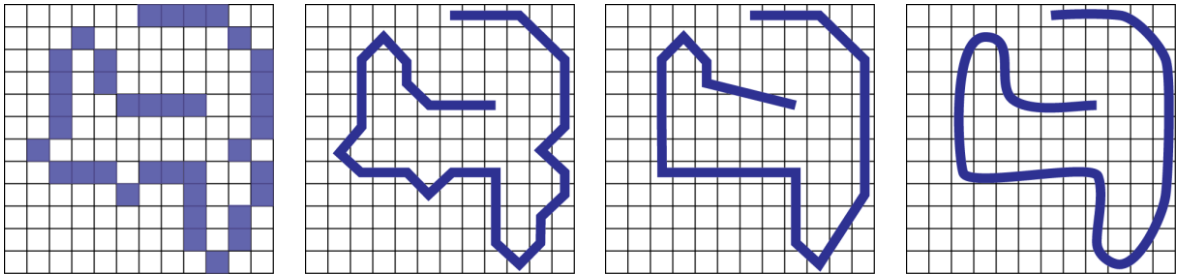
$$s_{i,j} = \begin{cases} 0 & j = 1 \\ \sum_{k=1}^{j} \sqrt{\left(P_{i,k}(1) - P_{i,k-1}(1)\right)^2 + \left(P_{i,k}(2) - P_{i,k-1}(2)\right)^2} & o.w. \end{cases} \quad (1)$$

where n_i is the number of points in the i^{th} stroke and $P_{i,j}$ presents the location of the j^{th} point of that stroke the stroke with n_i points is formalized with $2n_i$-2 piecewise linear functions as:

$$P_i(1) = F_{i1,1}(s) = \begin{cases} h_{i1,1,1}(s) & 0 \le s < s_{i,2} \\ h_{i1,1,2}(s) & s_{i,2} \le s < s_{i,3} \\ \dots & \dots \\ h_{i1,1,n_j-1}(s) & s_{i,n_j-1} \le s < s_{i,n_j} \end{cases}$$
$$P_i(2) = F_{i1,2}(s) = \begin{cases} h_{i1,2,1}(s) & 0 \le s < s_{i,2} \\ h_{i1,2,2}(s) & s_{i,2} \le s < s_{i,3} \\ \dots & \dots \\ h_{i1,2,n_j-1}(s) & s_{i,n_j-1} \le s < s_{i,n_j} \end{cases} \quad (2)$$

where $h_{i1,1,.}$ and $h_{i1,2,.}$ present the linear functions fitted with the stroke points for the X and Y axes, respectively. See Fig. 3b as an example of this approach.

2- *Sparse piecewise*: In this approach, after choosing the points with the most important points from each stroke (by the approach presented in the "*Separating method*"), we connect these points with piecewise-linear functions. Similar to the latter method, the length of the traced path in the chosen points is considered as the input of piecewise linear functions. It is worth mentioning that to compute the length of the traced path for the chosen points, the points with less importance should be considered for the summation operator in Eq. 1. Also, the formulation of this approach is similar to Eq. 2, except this only includes the linear functions between the main points. Fig. 3c tried to illustrate an example of this approach.
3- *Sparse spline*: This approach is similar to the previous one in all cases, except that this approach uses spline functions between every two main points, see Fig. 3d.



(a) (b) (c) (d)

Fig. 3: Dense-To-Vector methods: The filled cells in (a) presents the stroke points. (b), (c), and (d) are vector presentations of the desired stroke obtained by *Dense Piecewise*, *Sparse Piecewise*, and *Sparse Spline* methods, respectively.

Now, for each stroke, we have three mathematical functions: $P_{i1}=F_{i1}(s)$, $P_{i2}=F_{i2}(s)$, and $P_{i3}=F_{i3}(s)$, where P_{i1}, P_{i2}, and P_{i3} denote the stroke point measured by *Dense Piecewise*, *Sparse Piecewise*, and *Sparse Spline* methods, respectively. Also, F_{i1}, F_{i2}, and F_{i3} are the designed stroke trajectories' functions by *Dense Piecewise*, *Sparse Piecewise*, and *Sparse Spline* methods, respectively.

It should be mentioned here that for drawing a stroke with respect to the time, we need to define the desired tracing length as a function of the time as $s = g(t)$, where g denotes the tracing function and t denotes the traced time.

3. Case Study & Simulation Results

In this section, we demonstrate the applicability of the proposed approach by simulating a drawing robot. The considered robot includes a pen (for drawing on a sheet), a manipulator (for horizontal movements of the pen), and a body (for vertical movement of the manipulator), see Fig. 4. This robot has 3-DoFs (two ones for horizontal movements and the other one for vertical movement). The horizontal movement of the pen is achieved by two revolute actuators on the serial manipulator. Also, a linear actuator is used for vertical movement.

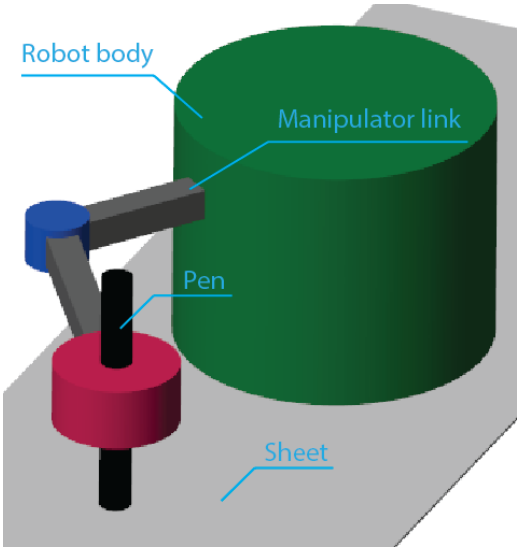


Fig. 4: The drawing robot includes a body, a manipulator (with two series links), and a pen.

We consider a mass of 100 grams for both links. The pen simulates a mass of 10 grams. Accordingly, all movable parts weigh 410 grams. All robot joints have a small viscous friction torque with a coefficient of 0.00001 N.m/rad.s. The friction force between the pen and sheet is simulated by a Coulomb friction force with a magnitude of 0.001 N.

The desired contours with 1498 points are illustrated in a matrix with 255×255 cells, as shown in Fig. 5. The proposed method for sorting the contours' points resulted in 14 strokes. For each stroke, we design three trajectories according to *Dense Piecewise*, *Sparse Piecewise*, and *Sparse Spline* approaches. For *Sparse Piecewise* and *Sparse Spline* methods, we consider μ=0.09. Also, for each case, we define the tracing function as $g(t)$=0.1t. Accordingly, we have a constant tracing speed (0.1 m/s) for all trajectories.

Fig. 5: The desired contours

As mentioned before the designed strokes' trajectories present the points' location on *ConMat*. Hence, we should map the strokes' points' location on the robot workspace. For this purpose, we define the drawing workspace as a square with a width and a height of 12×12 centimeters in the front of the robot's body. Both links of the robot have a length of 0.1 m. Accordingly, we use the inverse kinematics relationships of the manipulator to find the appropriate angles of revolute actuators to reach a target point in the drawing workspace.

After drawing each stroke, the manipulator is stopped and moved up. Then, it begins to move to the initial position of the next stroke with a determined speed (4.5 cm/s). Thereafter, the manipulator is moved down to begin drawing the next stroke. For each vertical movement, we consider a delay time (0.1 seconds).

We have implemented the robot model in the MATLAB SimMechanics toolbox. In the simulations, we assume all actuators have perfect controllers. This means that in the simulation, we assume the trajectory tracking error equals zero.

According to the mentioned conditions, we simulated the drawing task for the three designed trajectories. The simulation results showed that the robot with the first method (F_{i1} with 1497 piecewise linear functions and an overall length of 0.84 m) consumed 0.9 joules of energy in 8.42 seconds. Also, the drawn contours obtained with this method are shown in Fig. 6a. The robot with the second method (F_{i2} with 72 piecewise linear functions and an overall length of 0.77 m) consumed 0.14 joules of energy in 7.69 seconds. The contours drawn with this method are illustrated in Fig. 6b. Also, the third method (F_{i3} with 72 spline functions and an overall length of 0.78 m) resulted in 0.22 energy consumption in 7.8 seconds. The contours drawn with this method are illustrated in Fig. 6c.

4. DISCUSSIONS

According to the obtained results, we can conclude that the robot with the first method (*Dense Piecewise*) needs the highest energy to draw the desired contour. Meanwhile, because the digital raster graphic includes noise, the obtained drawn contours include wave-like noise, too. Indeed, these undesirable movements, which are the result of noisy data, increased the energy consumption of the drawing task. Hence, considering all raster points for path planning is an inefficient approach for such a case. Especially, for raster graphics with significant errors.

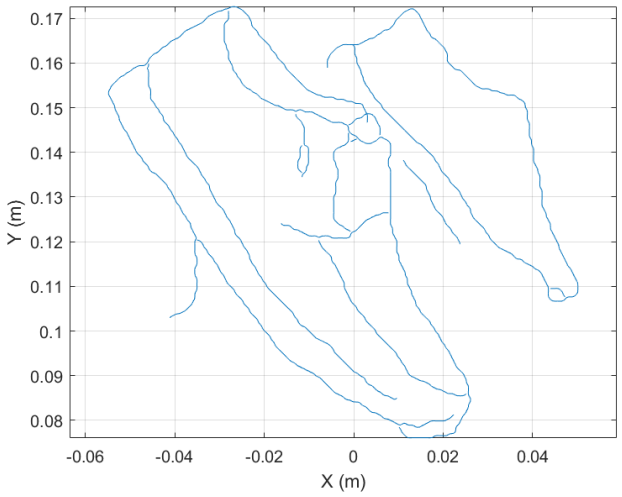
Accordingly, in the second approach (*Sparse Piecewise*), we used the proposed method to remove the less important points to exclude parts of noisy data. The results indicate that the second approach has the least energy consumption. However, because the remained points are connected with piecewise linear functions, the obtained drawn lines are very simplified in comparison to the desired contours. In other words, the fitted functions are not so flexible to yield the desired contours.

To address the mentioned problem, in the third approach (*Sparse Spline*), we used spline functions to connect the main points. The results showed this approach with a little more energy consumption than the second one, achieved the most acceptable curvatures.

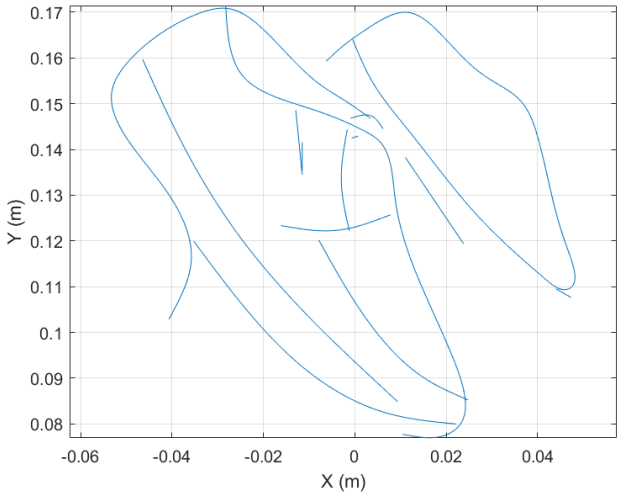
Needless to say, each method has its advantages and disadvantages. The main advantage of the *Dense Piecewise* approach is considering all given raster points (without neglecting any data) in the path planning procedure. This is good for cases with negligible noise. On the other hand, this feature may be a disadvantage for significant noisy data. Besides, this needs more time to obtain the output.

The advantage of *Sparse Piecewise*, in addition to excluding some noisy data, is its simplicity and speed. In other words, for cases where fast drawing is desirable, this approach can be an ideal candidate. However, its disadvantage is that it is not sufficiently smooth to obtain curvy lines.

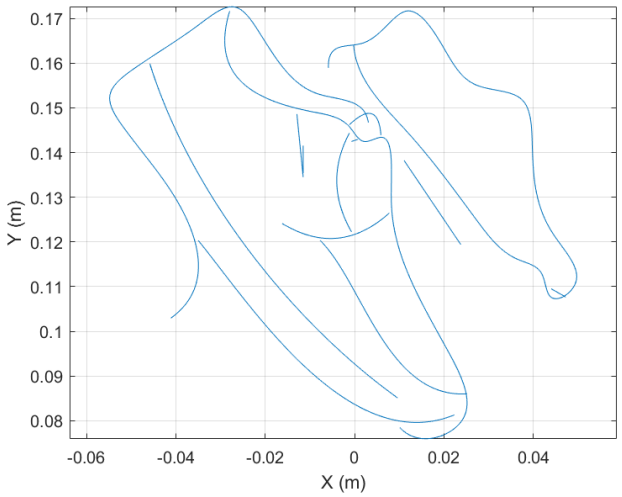
Nevertheless, the *Sparse Spline* stands between *Dense Piecewise* and *Sparse Piecewise*. Indeed, although this approach includes less noisy data than *Dence Piecewise*, its output has proper flexibility to achieve curvy contours. Hence, we think the proposed approach is a simple, fast, and efficient approach for path planning to draw contours. However, the disadvantage of this approach is excluding parts of given data, which may result in undesirable contours in cases with negligible noisy data.



(a)



(b)



(c)

Fig. 6: The drawn contours in the simulation: (a), (b), and (c) presents the obtained contours by *Dense Piecewise*, *Sparse Piecewise*, and *Sparse Spline* methods, respectively.

5. CONCLUSIONS

For path planning to draw contours, this work firstly proposed a method to sort the points of desired contours in several strokes. Secondly, a novel method was proposed to identify the main points of each stroke. Accordingly, to show the applicability of the proposed method, three approaches were introduced for Raster-To-Vector conversion: *Dense Piecewise*, *Sparse Piecewise*, and *Sparse Spline*. Finally, considering energy efficiency as a desired feature for drawing robots, in the simulations, we studied the effect of using the proposed approach on the path planning of a simulated drawing robot. The simulation results showed the superiority of the proposed method in comparison with a conventional method.

In this work, we consider the same tracing function for comparing the presented approaches. Nonetheless, designing a proper tracing function may affect the energy efficiency of the robot. This will be studied in our future works. Also, in future works, we will design and implement a drawing robot in an attempt to study the proposed approaches on a real robot.

REFERENCES

Adamik, M., Goga, J., Pavlovicova, J., Babinec, A., & Sekaj, I. (2022). Fast robotic pencil drawing based on image evolution by means of genetic algorithm. Robotics and Autonomous Systems, 148, 103912.

Beltramello, A., Scalera, L., Seriani, S., & Gallina, P. (2020). Artistic robotic painting using the palette knife technique. Robotics, 9(1), 15.

Calinon, S., Epiney, J., & Billard, A. (2005, December). A humanoid robot drawing human portraits. In 5th IEEE-RAS International Conference on Humanoid Robots, 2005. (pp. 161-166). IEEE.

Coelho, A., Branco, P., & Moura, J. M. (2018, November). A brief overview on the evolution of drawing machines. In International Conference on Intelligent Technologies for Interactive Entertainment (pp. 14-24). Springer, Cham.

Cooney, M. D., & Menezes, M. L. R. (2018). Design for an art therapy robot: An explorative review of the theoretical foundations for engaging in emotional and creative painting with a robot. Multimodal Technologies and Interaction, 2(3), 52.

Gao, Q., Chen, H., Yu, R., Yang, J., & Duan, X. (2019, February). A robot portraits pencil sketching algorithm based on face component and texture segmentation. In 2019 IEEE International Conference on Industrial Technology (ICIT) (pp. 48-53). IEEE.

Gao, F., Zhu, J., Yu, Z., Li, P., & Wang, T. (2020, October). Making robots draw a vivid portrait in two minutes. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 9585-9591). IEEE.

Hood, D., Lemaignan, S., & Dillenbourg, P. (2015, March). When children teach a robot to write: An autonomous teachable humanoid which uses simulated handwriting. In Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction (pp. 83-90).

Hsu, C. F., Kao, W. H., Chen, W. Y., & Wong, K. Y. (2017, June). Motion planning and control of a picture-based drawing robot system. In 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS) (pp. 1-5). IEEE.

Jean-Pierre, G., & Saïd, Z. (2012, March). The artist robot: A robot drawing like a human artist. In 2012 IEEE International Conference on Industrial Technology (pp. 486-491). IEEE.

Jeon, M. (2017). Robotic arts: Current practices, potentials, and implications. Multimodal Technologies and Interaction, 1(2), 5.

Levachkine, S. (2003, July). Raster to vector conversion of color cartographic maps. In International Workshop on Graphics Recognition (pp. 50-62). Springer, Berlin, Heidelberg.

Liu, C., Wu, J., Kohli, P., & Furukawa, Y. (2017). Raster-to-vector: Revisiting floorplan transformation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2195-2203).

Luo, R. C., Hong, M. J., & Chung, P. C. (2016, October). Robot artist for colorful picture painting with visual control system. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 2998-3003). IEEE.

McCorduck, P. (1991). Aaron's code: meta-art, artificial intelligence, and the work of Harold Cohen. Macmillan.

Orzan, A., Bousseau, A., Winnemöller, H., Barla, P., Thollot, J., & Salesin, D. (2008). Diffusion curves: a vector representation for smooth-shaded images. ACM Transactions on Graphics (TOG), 27(3), 1-8.

Scalera, L., Seriani, S., Gasparetto, A., & Gallina, P. (2019). Watercolour robotic painting: a novel automatic system for artistic rendering. Journal of Intelligent & Robotic Systems, 95(3), 871-886.

Song, D., Lee, T., & Kim, Y. J. (2018, May). Artistic pen drawing on an arbitrary surface using an impedance-controlled robot. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4085-4090). IEEE.

Yu, D., & Chen, H. (2018, July). A review of robotic drawing. In 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER) (pp. 334-338). IEEE.

Wang, T., Toh, W. Q., Zhang, H., Sui, X., Li, S., Liu, Y., & Jing, W. (2020, April). 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, pp. 10402-10409).