A drawing robot pipeline with artist-inspired execution

Jimena Arruti
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
jimena.arruti@fing.edu.uy

Martín Ottavianelli
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
martin.ottavianelli@fing.edu.uy

Alfredo Solari
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
alfredo.solari@fing.edu.uy

Pablo Monzón
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
monzon@fing.edu.uy

Pablo Musé
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
pmuse@fing.edu.uy

Juan Pablo Oliver
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
jpo@fing.edu.uy

***Abstract*—This paper presents a novel pipeline for vision-based robot drawing, which mimics the artists' execution process. The pipeline takes an image as an input and outputs the result of the drawing process, or optionally, the position and velocity needed for each motor of a planar robotic arm to execute the aforementioned drawing.**

***Index Terms*—robot, image processing, art, drawing, sketching, robotic art, planar arm**

I. Introduction

Since the rise of cybernetics, and the dawn of artificial intelligence, many artists began to incorporate computational elements into their work [1]. In 1961, Gordon Pask, inventor and author of a vast literature on cybernetics [2], presented a series of *Learning Machines* [3]. Among them, Eucrates, which is considered to be the first computational artwork, gave birth to a new era of cybernetic art. Regarding visual arts and technology specifically, the work of Jean Tinguely is of paramount importance. In the mid-1950s, he began producing a series of generative works called Métamatics [4], consisting of machines that produced abstract visual artworks in collaboration with the artist, spectators and restorers.

Since then, technological advances have continued to interact with the world of the arts, and to this day there are several works whose motif consists of a robotic system creating a visual artwork. Examples of this are AARON [5] or eDavid [6].

The main precedent in terms of drawing robots is the work of Patrick Tresset [7]. His art pieces are mainly still life scenes or portraits, being drawn by one or more robots in different settings. Each such robot consists of a camera and a robotic arm holding a pen. A particularity of his work is that the robotic arms has an anthropomorphic disposition, without seeking to fully emulate a human arm in its appearance, presenting three joints (shoulder, elbow and wrist). In addition, throughout the performance, all the movements they perform are perceived as natural.

Fig. 1. Sketches produced by PARRA, the drawing robot pipeline presented in this work.

In this context, this paper presents the signal processing pipeline of an artistic installation that uses an image as input and outputs a drawing executed by means of a planar robotic arm and a pen. The robotic arm will not be discussed here, but will be assumed as a speed and position controlled, 3-

axis planar arm [8], with an extra motor at the tooltip for pen-up and pen-down binary positions.

This hardware setup is inspired in the design from Paul the robot [9] [10]. However, the developed software arises from a different perspective. In particular, to our knowledge the approach taken for the shading process was not previously reported, and therefore it is presumed to be a novel contribution to the field. Two examples of achieved results by our drawing robot pipeline (*PARRA*) are presented in Figure 1.

II. Image Processing

The main restriction of the robot is that the drawing has to be monochromatic, as only one drawing tool (e.g. a pen) is used. Moreover, the pen is either pressing the paper or lifted, without considering pressure variations to change saturation of ink in the paper. This limits the drawing strategies that can be considered in the pipeline for boundaries and shading.

The first step of the processing pipeline consists of abstracting an image into a set of ideal curves that compose the sketch, each represented as a list of coordinates. When sketching, the robot takes into account two strategies to represent the picture: drawing edges, to distinguish between different objects -and different features in the same object-, and shading, for lighting information. The shading techniques are inspired by classical engravings procedures, and are believed to be new in the field.

A. Edge Detection

A Canny-Devernay algorithm implementation [11], [12] is used to detect the curves that compose the edges of the image. Its parameters (a lower and an upper threshold for the image gradient, and the standard deviation σ of the Gaussian filtering) are also parameters of the drawing robot pipeline. Different outcomes of this procedure are shown in Figure 2.

B. Shading Strategies

An image shading process is designed, resembling well known drawing techniques used in the visual arts, and taking into consideration the aforementioned restriction of the system to handle different pressure values in its strokes.

After exploring several techniques employed by visual artists, it was decided to design a shading process that emulates the technique of hatching (or *hachure* in French) [13], which is often used in ink drawings and engravings to generate different levels of darkness. This technique consists of creating different intensities of gray by drawing parallel lines. The perceived variation in intensity is the result of varying the amount, thickness and spacing of those lines, and even by the superposition of two or more layers of lines that are not parallel to each other. In addition, contrast between areas can be achieved by changing the angle of the shading layers. Therefore, a possible first approach consists of shading the image with parallel and equidistant line segments, achieving the effect of darkness or lightness by altering the local density of the segments.

Fig. 2. From left to right, downwards: original image; edges computed with the Canny/Devernay implementation described in [12], changing the value of σ and mantaining the values for the lower and upper threshold (4 and 5 respectively). From left to right, downwards: original image; edges detected with $\sigma = 2$, with $\sigma = 3$ y con $\sigma = 4$. As σ increases less edges are detected. $\sigma = 3$ is chosen as the default value for the system.

In order to represent the intensity value by the level of concentration of the lines, first the image must be quantized to delimit regions that are distinguishable in the shading. Secondly, it is also necessary to associate the varying spacing of the lines with each level of quantization, and to choose the direction in which the lines will be drawn in each area.

A second and final approach incorporates the shading method proposed in the initial approach to an iterative process of hatching, where the image is previously filtered with a median filter and quantized into N gray levels, to delimit different shading regions. Then, those regions are shaded by overlapping layers in an iterative process, in which any new iteration excludes the lightest region shaded in the last one. The lightest region in the image is never shaded. Thus, each iteration overlaps layers of parallel lines, each one with a different orientation, progressively reaching a higher density of lines for the darker areas. This process is summarized in Algorithm 1.

Algorithm 1 can be easily modified to only apply the shading process to the M darkest quantization levels of the images, adding variety to the possible outcomes. This is shown in Figure 3.

Algorithm 1 Hatching procedure

Require: quantized_image
1: shading_curves = []
2: angles = $[i2\pi/(N-1)]$ con $i \in [0, N-2]$
3: shuffle angles
4: **for** i in $[0, N-2]$ **do**
5: region_to_hatch = quantized_image <= i
6: shading_curves.append(Hatch(region_to_hatch, angles[i]))
7: **end for**

Fig. 3. From left to right, downwards: original image; 3 bit quantization; complete shading process; shading process for the 5 darkest quantization levels.

III. Drawing Curves

The previously mentioned stages output the ideal curves that would compound the final work. Each one of these curves is stored as a list of connected coordinates.

As stated in the introduction, the system is made for a planar arm with three joints. The fact that the arm is planar implies that it is more restricted than an actual human arm, which allows three-dimensional movement. Nevertheless, although the system is physically constrained, it still needs further restrictions, on account of the copious amount of possibilities for moving from one point to another, even when having the angles of the joints and the speed of the motors quantified.

A. Restrictions

It is intended that the arm moves as similarly as possible to a human arm, thus defining the range of movement for each joint, and suggesting the goal of a soft movement. In addition, since the three degrees of freedom system allows several possible arm configurations for drawing the same stroke, the use of additional restrictions simplifies the computation. Ultimately, the following restrictions are imposed:

- Each motor's range of motion is limited to certain values to mimic the functioning of the corresponding joint of a human arm.
- Each motor moves at constant velocity while drawing a curve.
- The three motors start and end their motion simultaneously.

B. Obtaining angle and speed values

Once the image processing abstraction is completed, a method for translating each ideal curve into commands for the arm's motors is needed. Considering the previously mentioned restrictions, and the speed and position of each motor as the variables, the following two-step procedure for obtaining the motors' commands is designed.

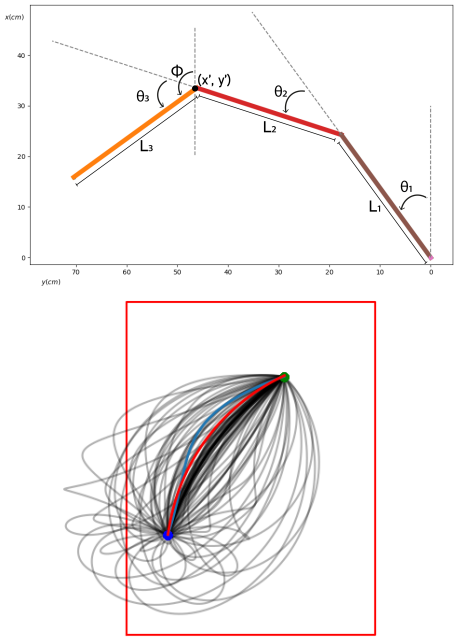

Fig. 4. Up: Three-joints arm with its relevant parameters. Down: Ideal curve (blue), feasible curves (black) and selected curve (red).

1) Finding feasible curves: This step involves finding all the feasible curves that are similar to each ideal curve. This is performed by finding a set of curves that start and end in the same points as the ideal curves, while following the aforementioned restrictions.

The possible configurations of the arm for the start and end points of the curves are derived using inverse kinematics [14] [8]. For every point in the Cartesian coordinate system, assuming a value for Φ (as defined in Figure 4), the corresponding angles for each joint can be computed.

A grid of I possible initial angles Φ_j^{init} and F possible final angles Φ_k^{final} is defined, limiting the amount of feasible curves. For those points, the joints' angles are computed. Then, feasible curves are defined by all combinations of possible arm configurations found for the initial and final points, while considering constant speed for each motor and simultaneous start and end of motion of the three motors. This last step involves computing each curve's points using direct kinematics.

2) Finding the most similar feasible curve: A measure of similarity is needed to choose which feasible curve is more adequate to represent an ideal curve. To this end, a similarity metric, considering discrete curves, is proposed in 1.

$$d(C_T, C_F) = \sum_{i=1}^{N} \min_{j \in [1..M]} \left|\left| C_T[i] - C_F[j] \right|\right|_2, \qquad (1)$$

where $C_T = \{(x_T[i], y_T[i]),\ 0 \leq i \leq N\}$ is the ideal curve and $C_F = \{(x_F[j], y_F[j]),\ 0 \leq i \leq M\}$ is the feasible curve.

Fig. 5. Sketch by Parra, with equidistant shading

A result of choosing the most similar feasible curve according to 1 is illustrated in Figure 4. For an ideal curve in blue, several candidate feasible curves in black are shown. Following the described method, the red curve drawn was found to be the most similar one.

C. Further enhancements

With the aim of providing the drawing process with more human characteristics, while expanding the aesthetic possibilities of the results two specific algorithms are designed: one of them implements a curve partitioning method, while the other one adds hand-tremor noise to the shading process.

As shown in Figure 5, shadings produced by fixed equally spaced strokes make the image look stiff and artificial. On the other hand, considering non-equispaced strokes by adding jitter to the distances causes the illusion of human made drawing more likely, as seen in Figure 8.

1) Curve Partitioning: The motivation behind incorporating to the process an algorithm to split curves into segments of shorter length results from two fundamental ideas. First, the need to characterize the stroke of the robotic arm in such a way that it reproduces the naturalness of a human artist's stroke and composition; second, the opportunity to improve the representation of the ideal curves projected as feasible curves.

Naturalness can be compromised working with the above-stated method, when trying to draw curves that are too long, and by the possibility of encountering high local curvatures

Fig. 6. From left to right, downwards: original image; detected edges; feasible curves without enhancements; feasible curves after splitting the original curves with curvature threshold of 120 degrees; feasible curves after splitting the original curves with curvature threshold of 150 degrees; feasible curves after splitting the original curves with curvature threshold of 120 degrees and a maximum length of 50 pixels per curve.

at some points of the curve. Both factors hinder the chances to achieve sufficient similarity between the ideal curve and the optimal feasible curve, considering that so far, each ideal curve is represented by a single trace.

The curve partitioning algorithm consists of two stages. In the first stage the local curvatures at each point of the curve are estimated, based on the analysis of the curvature at discrete points with the inner product. Once the curvatures are computed, the partitioned curve indices are chosen in such a way that they coincide with the points whose local curvatures exceed a certain threshold. Therefore, partitioning occurs at points where the local angle is too acute for it to be natural to draw the curve as a single stroke. Whenever the curve segments resulting from the first partitioning are still larger than a configurable length (set as a function parameter), a second stage iterates in an equidistant segmentation, until obtaining segments with lengths less than, or equal to, the desired length.

Some results of applying this algorithm can be seen in Figure 6.

2) Noise aided shading: In order to remove the robotic precision of the shading process based on hatching, seeking to achieve a certain degree of human appearance, a shading algorithm including noise is designed.

In a first instance, a random perturbation on the horizontal and vertical axes is added to the ideal curve, limited by a maximum deviation margin parameter defined as a percentage of the canvas size in the horizontal and vertical directions. This way, the ideal curves are perturbed prior to its projection onto the space of feasible curves.

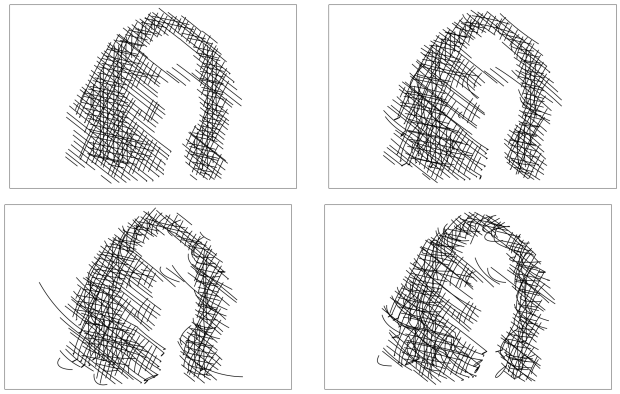

Fig. 7. A single shading layer varying the amount of random noise in the shading algorithm. From left to right, downwards, noise amount increases. In particular no noise is added in the first image.

In a second instance, random noise is added to all the candidate curves in the space of feasible curves, so that when comparing the ideal curve with the new noisy curves, the best one of the candidates (the curve whose distance to the ideal curve is the least) may differ from the one that would have been chosen without adding noise. This randomness increases the naturalness of the drawing, which is clearly depicted in Figure 7.

IV. Drawing order

For an actual drawing performance to take place, the order in which the curves are drawn is of extreme importance. To derive a computational order to the curves that are drawn during the execution, and to allow several variations, the cost function presented in (2) is considered.

$$f_{x,y}(l, x_o, y_o) = d(x, y, x_o, y_o) - \alpha l. \tag{2}$$

$f_{x,y}(l, x_o, y_o)$ represents the cost of drawing each possible next curve, considering the length l of the curve, a configurable point of reference (x_o, y_o) -e.g. center of the canvas, final point of the previous curve, initial point of the previous curve- and the initial point of the next curve (x, y). The parameter α is a regularization constant that can be set by the user. Typical default values for α range between 0.1 and 1.

Ultimately, the curve yielding the minimum cost is drawn next; this calculation takes place each time a new curve is drawn.

V. Results

Two examples of drawings produced by the pipeline are shown in Figure 1. More examples can be found in the project's portfolio: Sketches by PARRA[1].

A drawing process simulator was developed using Python. This simulator outputs a video that shows the arm drawing on a predefined canvas. Simulations of the drawing process can be found in the project's YouTube channel (Proyecto PARRA[2]).

Fig. 8. Sketch by PARRA, including split curves and shading with noise

This system was developed in the context of an undergraduate degree thesis. Further details can be found in the PARRA project documentation [15].

VI. Conclusions

In this work we presented the design and implementation of a drawing robot pipeline that takes a picture as input and produces a video that simulates the motion of a robotic arm that draws from life. In addition, it is possible to obtain instructions for each motor's joint to draw each curve.

Besides the aesthetics of the resulting drawings, the major goal of this project was to endow the drawing execution with human-like features. By imposing the physical restrictions inherent to the motion of a human arm, and considering the processes that usually take place when drawing from life (from techniques such as hatching, to an algorithm for determining which curve should be drawn next), a result perceptibly similar to the drawing process of a human artist was accomplished.

To this end, imperfections or randomness were also included, were also included, removing the resulting aesthetics from any hint of rigidity or robotic precision.

[1]https://tinyurl.com/PARRAportfolio

[2]https://tinyurl.com/PARRAproject

REFERENCES

[1] R. I. García, *Arte y robótica: La tecnología como experimentación estética*. España: Casimiro Libros, 2016.

[2] G. Pask, *An Approach to Cybernetics.* Hutchinson, 1961.

[3] T. Dreher, "History of computer art," [online] 2013. [accessed on 2020-06-15]. http://iasl.uni-muenchen.de/links/GCA-II.3e.html.

[4] Wikipedia, "Métamatics," [online] 2010. [accessed on 2020-06-15]. https://en.Wikipedia.org/wiki/M'/etamatic.

[5] H. Cohen, "AARON," [online] 2011. [accessed on 2019-03-17]. http://aaronshome.com/aaron/index.html.

[6] O. Deussen, T. Lindemeier, S. Pirk, and M. Tautzenberger, "Feedback-guided stroke placement for a painting machine," in *Proceedings of the Eighth Annual Symposium on Computational Aesthethics in Graphics, Visualization and Imaging*, 2012, pp. 25–33.

[7] P. Tresset, "Patrick Tresset," [online] 2017. [accessed on 2020-06-15]. https://patricktresset.com/new/.

[8] V. Kumar, "Introduction to robot geometry and kinematics," *Penn Engineering*, 2002.

[9] P. Tresset and F. Fol Leymarie, "Portrait drawing by Paul the robot," *Computers & Graphics*, vol. 37, no. 5, pp. 348–363, 2013.

[10] F. F. L. Patrick Tresset, "Proyecto AIKON," [online] 2009. [accessed on 2020-06-15]. https://sites.google.com/site/aikonproject/.

[11] F. Devernay, "A Non-Maxima Suppression Method for Edge Detection with Sub-Pixel Accuracy," INRIA, Tech. Rep. RR-2724, Nov. 1995. [Online]. Available: https://hal.inria.fr/inria-00073970

[12] R. G. von Gioi and G. Randall, "A sub-pixel edge detector: an implementation of the Canny/Devernay algorithm." *IPOL Journal*, vol. 7, pp. 347–372, 2017.

[13] B. Grabowski and B. Fick, *Printmaking: A Complete Guide to Materials and Processes*. Prentice Hall, 2009.

[14] J. J. Craig, "Introduction to robotics, 3rd edition," *Prentice Hall, Pearson*, 2002.

[15] J. Arruti, M. Ottavianelli, and A. Solari, "Parra : Artífice de realizaciones robóticas artísticas," *undergradute thesis, Universidad de la República (Uruguay). Facultad de Ingeniería*, 2020.