Artistic Pen Drawing on an Arbitrary Surface using an Impedance-controlled Robot*

Daeun Song†, Taekhee Lee‡ and Young J. Kim†

***Abstract*— We present a semi-autonomous robotic pen-drawing system that is capable of creating pen art on an arbitrary surface with varying thickness of pen strokes but without reconstructing the surface explicitly. Our robotic system relies on an industrial, seven-degree-of-freedom (7DoF) manipulator that can be both position- and impedance-controlled. We use a vector-graphics engine to take an artist's pen drawing as input and generate Bézier spline curves with varying offsets. In order to estimate geometric details of the target, unknown surface, during drawing, we rely on incremental and adaptive sampling on the surface using a combination of position and impedance control. Then, our control algorithm physically replicates this drawing on any arbitrary, continuous surface by impedance-controlling the manipulator. We demonstrate that our system can create visually-pleasing and complicated artistic pen drawings on general surfaces without explicit surface-reconstruction nor visual feedback.**

I. INTRODUCTION

Pen and ink drawings are ancient fine art traced back to Greek art in 300 CE. They are also a convenient, affordable, diverse medium of art, which makes them universal across the time, region and styles, ranging from the Baroque to the Neoclassical art movement, regionally from Islamic to Chinese art, and methodologically from Calligraphy to contemporary digital, graphics art [1].

Due to the recent impressive development of robotic technology, artistic application of robots drew a lot of attention from the robotic and art community. In particular, fully- or semi-autonomous robotic drawing is one of such movements that sparked the recent robot art competition (`http://robotart.org`). However, creating an autonomous robotic drawing system is hard, as it requires robust components of control, sensing, planning, and man-robot interfacing. What makes matters more challenging is that the artistic nature of robotic art requires the interdisciplinary participation of, for instance, art and computer graphics technology such as non-photo realistic rendering (NPR).

Robotic pen drawing requires contact-based manipulation, a challenging problem in robot manipulation and control, with respect to the target drawing surface. In general, contact-based manipulation makes up a large proportion of robotic tasks both in industrial and service robotic settings where the robot grasps or pushes objects while maintaining contact with objects [2]. As a result, many approaches have been developed to enable robotic manipulation based on the contact. In this context, impedance control has become a popular choice to deal with contact tasks. However, existing industrial robots, e.g. spray-painting robot, are often not allowed to touch the workpiece due to the uncertainty present in sensing and modeling the workpiece. Our drawing robot has a similar issue and we attempt to address this uncertainty problem with surface sampling and impedance control techniques.

Main Results: In this paper, we add another complexity to the already challenging problem of robot drawing by proposing artistic pen drawing on an arbitrary, non-planar surface. Our semi-autonomous robotic pen-drawing system is capable of creating pen art on an arbitrary surface with varying thickness of pen strokes, but without reconstructing the surface explicitly nor with any vision support. This can be realized by using a seven-degree-of-freedom (7DoF) manipulator that can be both position- and impedance-controlled, equipped with torque sensors for every joint. The software and algorithmic side of our drawing system are based on two main components: efficient and intuitive vector-graphics engine, and impedance-controlled drawing algorithm. Our novel vector graphics engine takes an artist's pen drawing as input and generates a set of quadratic Bézier spline curves with varying offsets. Our vector graphics is able to render spline curves in a resolution-independent manner so that robotic drawing system can physically realize the drawing on an arbitrary, continuous surface in any scale with no discontinuity. Specifically, our robot drawing algorithm replicates the digital drawing on a continuous surface by using a combination of position- and impedance-control. In order to estimate the geometry of the unknown, target surface using only joint-torque sensors, our robot incrementally samples the surface before and during drawing and builds an adaptive and implicit representation of the surface using a quadtree. We demonstrate that our system can create visually-pleasing and complicated artistic pen drawings on general, non-flat surfaces such as a water tank or a cone.

The rest of this paper is organized as follows. We survey works relevant to robotic drawing in Sec.II. In Sec.III, we propose our new vector graphics method and explains robotic incarnation of this technique using a combination of position- and impedance-control techniques in Sec.IV. We show our implementation results and discuss them in Sec.V, and conclude the paper in Sec.VI.

*This project was supported by the NRF in Korea (2017R1A2B3012701). T. Lee was supported by the NRF in Korea (2017R1C1B2007306).

†D. Song, and Y.-J. Kim are with the Department of Computer Science and Engineering at Ewha Womans University in Korea `daeun7250@ewhain.net,kimy@ewha.ac.kr`

‡T. Lee is with the Department of Game and Multimedia Engineering at Korea Polytechnic University in Korea `watersp@kpu.ac.kr`

II. PREVIOUS WORK

A. Robotic Drawing

An early history of creating drawing machines can be attributed to artistic work by Jean Tinguely and Harold Cohens Aaron [3]. In computer graphics and robotics community, an earlier attempt to create drawing robots is largely based on a plotter-type, special-purposed machine. More recently, research efforts have been put into to use a high-DoF, general robot or manipulator for robotic drawing that can span a wide spectrum of artistic expressions.

The Pumapaint project [4] is a telerobotic painting robot that allows online users to draw paintings remotely using a PUMA robot. Calinon et al. [5] used the HOAP2 humanoid robot to draw human portraits, that follow human-characteristic styles. Paul the robot [6] is a robotic installation that creates observational portrait drawing, mimicking artist's stylistic signatures. eDavid [7] is a modified, industrial robot that can create a wide variety of painting styles from an image input. It relies on visual feedback to generate NPR-type painterly results. Recently, Galea et al. used an aerial robot (drone) to create stippling effects from an image input [8]. However, none of the existing works dealt with creating a pen drawing on a non-planar, general surface.

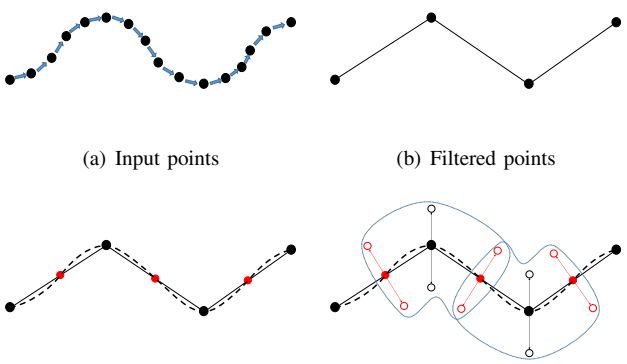
B. Vector Graphics

For robotic drawing system, vector graphics suits better than raster graphics as vector graphics can generate continuous and smooth pen strokes that can be mapped well to smooth robotic motions. Vector graphics typically fill pixels inside implicitly-defined curves with a certain width using CPU-based scanline methods [9], [10], [11]. Since vector graphics techniques need to render implicit curves every frame, the performance of CPU-based rendering methods is slow on dense screen resolution used by modern display devices.

Loop and Blinn suggested a GPU-based fast resolution-independent rendering method which can render paths and bounded regions [12]. Kilgard and Bolz [13] introduced a fast GPU-based, two-step approach, namely stencil-and-cover approach. The stencil step determines the stroked path's filled coverage and the cover step fills the area determined by the stencil step. There exist no vector-graphics method that can adequately reproduce human drawing consisting of free-form lines and curve, even though smooth curve rendering or retrieving methods such as [12], [14], [15], [16] may handle human drawings to some degree.

C. Impedance-controlled Robot

The idea of using impedance control for controlling the interaction between a manipulator and the surrounding environment was first proposed by Hogan et al. [17]. Since then, studies on impedance control-based robot interaction techniques have been done [18], [19]. Such a wide interest is motivated by the need for robotic systems with an ability to interact with unstructured environments beyond industrial environments. More recently, impedance-controlled collaborative robots have been introduced from both industry and academia, which are capable of sensitive object handling; e.g. LBR IIWA from KUKA Robotics, Baxter and Sawyer from Rethink Robotics, and Justin from DLR. On the robotic application side of using impedance-control, the work by [20] used two anthropomorphic dual arms of the Justin robot to unscrew a can. Lee et al. used impedance control for a dual-arm system using the relative Jacobian, which maps the joint velocities of the two arms to the relative motion between their end-effectors [21].



(a) Input points (b) Filtered points

(c) Mid-points generation (red) (d) Offset points generation for pen thickness

Fig. 1. Four steps to generate triangle polygon from input points for GPU-based vector graphics rendering.

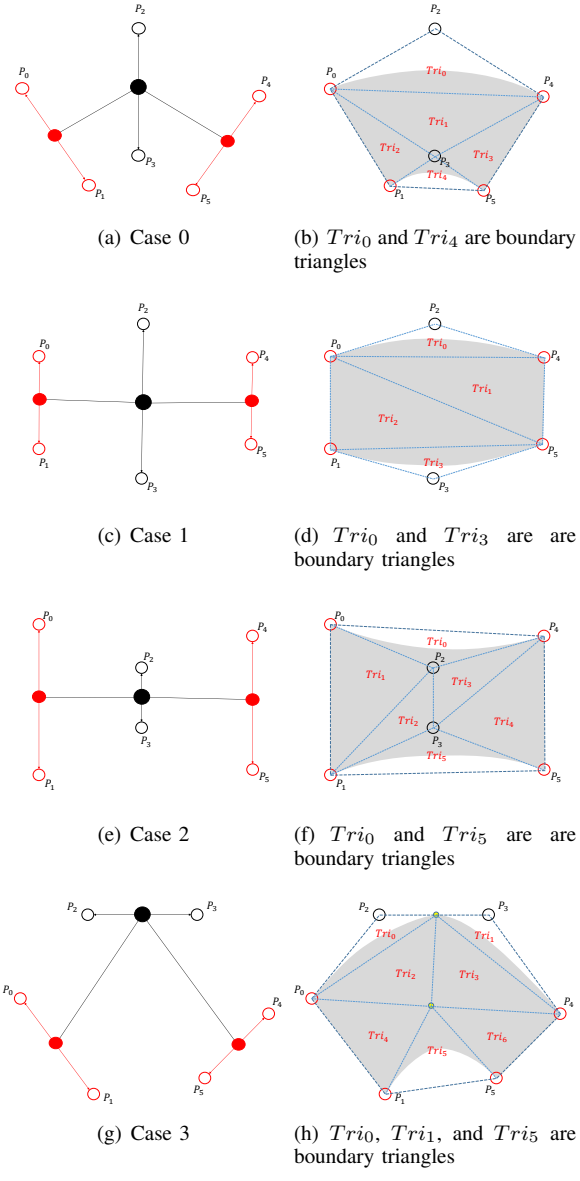
III. VECTOR GRAPHICS ENGINE

Raster graphics may perform poorly with robot manipulators, as it requires a stream of discrete, stop-and-go motions for a robot that would act jerky. On the other hand, vector graphics generates a sequence of continuous vectors that can be mapped to manipulators' continuous motion.

Our system receives input points from pen-ready devices such as tablet devices or mobile phones. The stylus pen generates two-dimensional points along with the corresponding pen-pressure. We convert these input points and pressures into vector graphics output, preview them by rendering them and provide them to a robotic manipulator to physically recreate the drawing on an arbitrary surface.

1) We filter out useless input points to reduce data size, as illustrated in Figure 1(b). Existing tablet-based pen-input devices typically produce sixty points per second, and these points can contain lots of useless and noisy points; e.g. many co-linear points on a straight line. During filtering, we also need to consider input pressure values to keep the variation of thickness. To do this, we apply the median filter with a narrow range, say five points per one filtering step, to filter out redundant points on a straight line. In this step, lots of co-linear points will be eliminated. Then, we apply the bilateral filter to the filtered result. Due to the nature of the bilateral filtering method, points located near the crest of successive input points survive as shown in Figure 1(b).

2) We calculate the mid-points of all successive input points.
3) We choose an input point as well as its two adjacent mid-points, calculated from the first step, to constitute three control points to define a single Bézier curve. This construction yields C^1 continuity of the entire spline curve.
4) To render a curve with varying thickness (or offset), determined by the pen pressure, we triangulate the bounded areas, as illustrated in Figure 1(d). We classify the bounded area into four cases as shown in Figure 2, and then triangulate them.



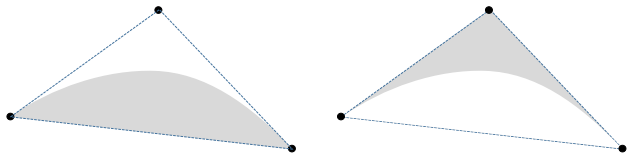
(a) Case 0 (b) Tri_0 and Tri_4 are boundary triangles

(c) Case 1 (d) Tri_0 and Tri_3 are are boundary triangles

(e) Case 2 (f) Tri_0 and Tri_5 are are boundary triangles

(g) Case 3 (h) Tri_0, Tri_1, and Tri_5 are boundary triangles

Fig. 2. Four cases of input-point sequences (a), (c), (e), (g) and their bounding polygons (blue lines) and triangulations (gray-shaded region) (b), (d), (f), (h). The red/black dots and lines on the first column represent the control points and offsetting lines connecting them, respectively. The black and red empty circles are offset from the red and black solid dots depending on the corresponding pen pressure (i.e. offset amount). Boundary triangles are rendered using [12], and the rest are rendered in solid color.



(a) Convex boundary triangle (b) Concave boundary triangle

Fig. 3. Resolution-independent methods such as [12] can fill a curve inside a triangle in both convex and concave manners.

The robotic curve drawing requires the results of step 3, a set of quadratic Bézier curves with C^1 continuity and pressures. Optionally, to preview curve rendering before sending the input to the robot, we use the triangles of step 4 and render them using resolution-independent curve rendering such as [12]. To render the curves, we also categorize the triangles into three types: convex boundary triangle, concave boundary triangle, and inner triangle (see Figures 3). For example, in Figure 2(b), Tri_0 is a convex boundary triangle and Tri_4 is a concave boundary triangle and Tri_1, Tri_2, and Tri_3 are inner triangles. All boundary triangles are rendered using [12] and inner triangles are rendered with solid color.

IV. ROBOTIC CURVE RENDERING

Impedance-controlled robots interact with the environment by employing a mass-spring-damper-like system with active control on the robots [2]. Such a system is well suited for the tasks in which contact forces should be kept small, while their accurate regulation is not mandatory. Thus, impedance control serves nicely for pen drawing tasks.

Robotic curve rendering in our system reproduces drawing on an arbitrary surface without explicitly reconstructing the target surface. However, the system still needs to estimate the geometry of the target surface for better drawing performance, and our impedance-controlled robot incrementally samples the surface before and during the drawing and builds an adaptive and implicit representation of the surface in a quadtree data structure.

Specifically, the robot begins by sampling an extent of drawing canvas space before kicking off actual drawing and keeps adding the control points of every stroke as new samples during drawing. The resulting sampling points constitute a 2.5D height field and are represented as a quadtree data structure $\mathcal{Q}$ as shown in Figure 4. During robotic drawing, for each stroke, the positions of the control points projected to the target surface are estimated using bilinear interpolation using the quadtree. Although the estimated, projected position is rough, we use a combination of position- and impedance-based control to enable the robot to accurately reproduce the drawing on the target surface. Algorithm 1 summarizes an overview of our robotic curve rendering process.

A. SURFACE ESTIMATION

The control points $\mathbf{c}$ for drawing strokes, generated by the vector rendering engine, are defined in $\mathbb{R}^2$, and need to be projected onto the target, unknown surface $\mathcal{D}$ in $\mathbb{R}^3$; i.e.

Algorithm 1: Robotic Curve Rendering

Input : $\mathcal{S}$, a set of strokes in Bézier curves in $\mathbb{R}^2$; $\mathcal{D}$, target drawing surface in $\mathbb{R}^3$
Output: Robotic curve rendering on $\mathcal{D}$

Scale $\mathcal{S}$ to fit the 2D workspace of $\mathcal{D}$;
Initialize the quadtree $\mathcal{Q}$ with a 2D extent of $\mathcal{D}$;
foreach *stroke* $s \in \mathcal{S}$ **do**
  foreach *control point* $\mathbf{c} \in s$ **do**
    Search four nearest points $\mathbf{p}_{1..4}$ of $\mathbf{c}$ from $\mathcal{Q}$;
    Estimate the height $\tilde{\mathbf{c}}_z$ of $\mathbf{c}$ on $\mathcal{D}$ using bi-linear interpolation on $\mathbf{p}_{1..4}$;
    Use impedance control to find an actual value of $\mathbf{c}_z$ starting from $\tilde{\mathbf{c}}_z$;
    Add $\mathbf{c}$ to $\mathcal{Q}$ with a height value of $\mathbf{c}_z$;
  end
  Draw s on $\mathcal{D}$ using position control;
end

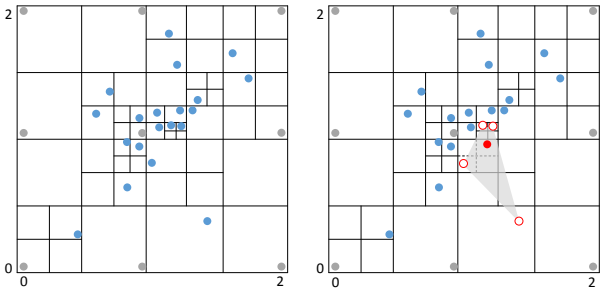
its height value $\mathbf{c}_z$ needs to be determined. To estimate the height $\tilde{\mathbf{c}}_z$ of $\mathbf{c}$ projected on the surface, we first search the four nearest points of $\mathbf{c}$ from the quadtree $\mathcal{Q}$, which forms a quad that includes $\mathbf{c}$ (e.g. the gray quad in Figure 4(b)). From the quad annotated with height values, bi-linear interpolation is performed to yield $\tilde{\mathbf{c}}_z$. Since this quad is not always rectangular, we map the quad to a unit square to facilitate the bi-linear interpolation.

Figure 4 illustrates a quadtree that our system uses for nearest neighbor search (NNS) [22]. Each quadtree node stores a single sampling point (i.e. the control point $\mathbf{c}$) containing its x-y coordinate and the height value ($\mathbf{c}_z$). When a new sampling point is inserted into a node, the node is split into four children if it already contains a sampling point. Note that our quadtree only grows but never shrinks, and thus does not require sophisticated tree re-fitting mechanism.

Once the targeted position of $\tilde{\mathbf{c}}_z$ is calculated, we slightly reduce the value to underestimate the height, in order to apply pressure to pen, used by the computed deviation in the next section (Eq. 1). The more strokes are drawn to the surface, the more sampled points are being collected; i.e. the quadtree is expanded. As a result, by the time of the completion of drawing, an estimation of interpolated values would be more reliable, resulting in intended pen pressure during drawing.

B. IMPEDANCE-CONTROLLED DRAWING

Once the positions of control points on the target surface are determined, the robotic manipulator performs curve rendering in a combined manner of position- and impedance-control. Specifically, the manipulator moves to the exact x-y position of the control points, while giving itself a bit of margin in the normal direction of the contact surface, exerting a spring-mass force. For each drawing stroke, underestimated target drawing positions define a virtual spline curve, which is placed slightly under the physical surface as shown in Figure 5. The deviation $\delta\mathbf{x}$ between the target



(a) Before a new point is inserted (b) After a new point is inserted

Fig. 4. Quadtree data structure for sampled points. The gray points represent the initial points covering the extent and the center of the drawing surface. The blue points represent the points incrementally sampled during the drawing, and the red solid point is a new point being inserted. The four nearest neighbor points are also highlighted in red and form a gray quad.

position, determined by the bi-linear interpolation, and the physical position of the pen tip results in a compliant force in Cartesian space:

$$\mathbf{f} = k\delta\mathbf{x}, \tag{1}$$

where k is the spring stiffness. The Cartesian impedance controller is configured in such a way that the robot is compliant only in the normal direction of the surface. Moreover, we still maintain the tangential motion of the target position using a position-based control. Then, given a set of pen strokes with beginning and end points, we draw each of the strokes independently. Furthermore, in order to avoid self-collision and kinematic limits of the robot, we pre-compute the robot's free configuration space beforehand and use it as a starting configuration for every stroke.

Our impedance control computes joint accelerations using the position and orientation feedback as well as the force and moment measurements. Then, the inverse dynamics computes torques for the actuators. This control scheme, in the absence of interaction, guarantees that the end-effector frame asymptotically follows the desired frame [23]. In the presence of contact with the environment, a compliant dynamic behavior is imposed on the end-effector according to the impedance using Eq. 1, but the torque due to the environmental contact forces is bounded as the position and orientation displacement between the reference and target frames is bounded.

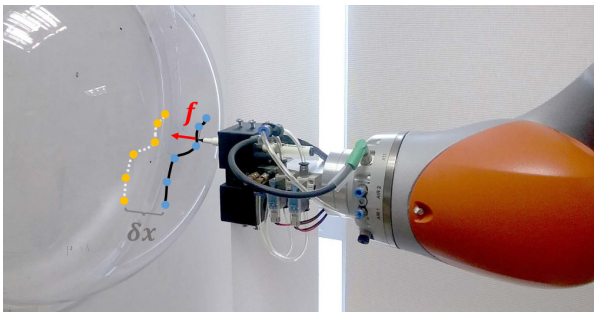
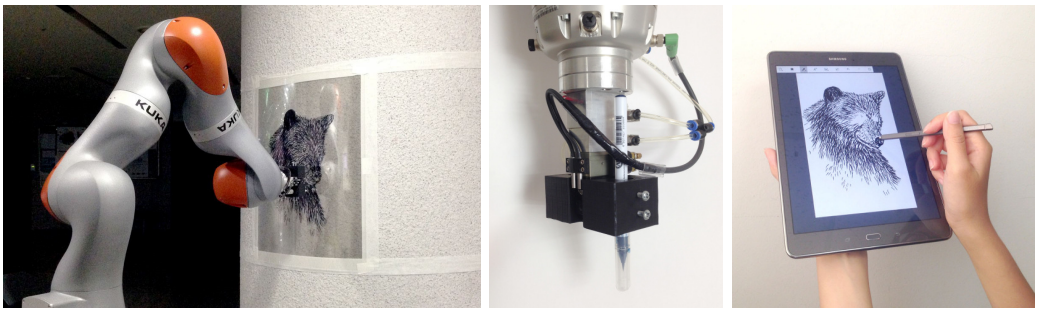


Fig. 5. Force ($\mathbf{f}$) is generated by the impedance-controlled manipulator. The black line represents an expected physical spline curve with its set of control points (blue dots), and the white dotted line represents a virtual spline curve, consists of the set of underestimated target positions (yellow dots).



(a) Robotic setup (b) 3D-printed pen-gripper (c) Vector graphics engine

Fig. 6. Robotic Drawing Setup

TABLE I
DRAWING STATISTICS

Drawings	(c)	(d)	(e), (f)
# of Strokes	1520	1942	523
# of Control points	66,910	159,895	72,845
Drawing Surface Size (mm)	252×491	252×491	126×262
Execution Time (min.)	221	317	216

V. RESULTS AND DISCUSSIONS

In this section, we show our implementation results and discuss robotic drawing results using our system.

A. IMPLEMENTATION DETAILS

In Figure 6, we show our experimental drawing system setup. In our experiments, we use KUKA LBR IIWA 7 R800 as a manipulator, which has seven DoFs and can be position- and force-controlled (Figure 6(a)). A 3D-printed gripper, designed to hold up to two different types and colors of pens, is attached to the end-effector (Figure 6(b)). The drawing tool can be any type of pointed pen that can resist gentle forces in order to perform impedance-controlled drawing.

We use Java programming language under Windows 10 64bit operating system along with Sunrise OS for interfacing the IIWA. We also use a Samsung Galaxy Tablet PC for running vector graphics under the Android operating system (Figure 6(c)). The number of initial sampling points before the drawing task is set to 9. However, this number could be either increased or decreased. A degree elevation technique based on de Casteljauś algorithm is used to match the higher-degree polynomials of spline blocks provided by Sunrise OS. We raise the quadratic Bézier spline curve with three control points to a quadratic spline curve with five control points.

The algorithm 1 presented in Sec. IV can be further optimized by sampling only the first control point of every stroke. The rest of control points are simply bi-linear interpolated using the quadtree.

B. EXPERIMENTAL RESULTS

Figure 7 shows examples of pen drawings on arbitrary surfaces, compared to the digital drawings created from vector graphics engine. We have been able to physically reproduce digital drawings on arbitrary surfaces. The experimented surfaces included not only a simple algebraic surface, such as a transparent half-sphere (Figure 7-(e)) but also unpredictable curved ones (Figure 7-(c), (d), (f)), such as a bumpy circular wall column, a water tank, a cone, a bucket, etc. The statistics of our experimental drawing results including the number of pen strokes, control points, size of drawing surface (i.e. canvas size), robot execution time and digital drawing time are provided in Table I. The execution time is nearly proportional to the number of control points and size rather than to the number of strokes, since the length of strokes may differ by the drawings. Also, note that the drawings can be reproduced on target surfaces of various sizes. Performance optimization has not been considered yet in this work, even though there may exist a few acceleration techniques possible to achieve it - for instance, varying robot execution speed depending on the surface curvature.

We have split the drawing tasks into two different colored sets, capable of reproducing pen drawing with two colors of pen. As shown in Figure 7-(b), (d), the brush color from vector graphics engine is also distinguishable from each other. We fix the approaching direction of the manipulator to be aligned to the initial stroke position, which may make the reproduction of the same digital drawing differ depending on a surface. Even though the original digital drawings take pen pressure into account resulting in varying thickness of pen strokes, our robotic drawing currently does not consider the pen pressure. However, the lack of stroke thickness may be implemented by adaptively changing the pen height from the pen pressure data of the vector graphics engine.

C. DISCUSSIONS

Pen drawing on an arbitrary surface is not an easy task even for humans, who can visually determine the shape of the surface to draw on. Our system, however, reproduces artists' digital drawings on a physical surface without any vision support. Our system can perform the drawing task on any surface and is not limited to specific surfaces as long as the surfaces are monotonic along the height direction (z-direction) (i.e. representable in 2.5D).

There are a few limitations in our current system, which is also our immediate future work. Currently, our system can create only a limited style of pen drawings. Our system uses only up to two colors, which can be held firmly by the gripper. The drawing tool needs to be firm enough in order to perform impedance-controlled drawing; otherwise, the perceived forces would be too small to be sensed. For

(a) (b) (c) (d) (e) (f)

Fig. 7. Robot Drawing results

instance, we can not have a brush-type pen as a drawing tool, which can easily bend.

Another problem is that our robot has a limited workspace, even though our vector graphics engine allows us to scale the drawing arbitrarily. However, we want to address this issue in future by making use of a mobile robot.

We decided to avoid using robot vision in our system, as robot-vision integration is not very robust with respect to noise and lighting condition, and it also has an occlusion problem, which could make our system limited. However, we still consider having a vision support in our system as future work, where vision can help surface-estimation.

VI. CONCLUSION

We presented a robotic pen-drawing system that can create pen art on an arbitrary surface using an impedance-controlled manipulator with vector graphics engine. Our vector-graphics engine takes an artist's pen drawing as input and generates Bézier spline curves. Our impedance-controlled drawing mechanism is used without any vision support to physically replicate digital drawing on an arbitrary, unknown surface. To do so, we implicitly and adaptively reconstruct the surface by incrementally sampling points during the drawing sequence. The proposed robotic drawing system still relies on human creativity while robot realizes a creation process and produces physical artworks on an arbitrary surface, which is quite challenging to achieve by human efforts. It goes beyond simulating human drawing and proceeds to explore a novel style of robotic drawing. As future work, we want to develop artist-machine collaborative art setup, for which our IIWA manipulator is designed originally.

REFERENCES

[1] A. L. Guptill, *Drawing with Pen and Ink*. Van Nostrand Reinhold Company, 1980.

[2] N. Hogan, "Stable execution of contact tasks using impedance control," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4. IEEE, 1987, pp. 1047–1054.

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

[4] M. Stein and C. Madden, "The pumapaint project: Long term usage trends and the move to three dimensions," in *Proceedings of the IEEE Intl Conference on Robotics and Automation (ICRA)*, 2005.

[5] S. Calinon, J. Epiney, and A. Billard, "A humanoid robot drawing human portraits," in *IEEE-RAS International Conference on Humanoid Robots*, 2005.

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

[7] T. Lindemeier, S. Pirk, and O. Deussen, "Image stylization with a painting machine using semantic hints," *Computers and Graphics*, vol. 37, 2013.

[8] B. Galea, E. Kia, N. Aird, and P. G. Kry, "Stippling with aerial robots," in *Computational Aesthetics in Graphics, Visualization and Imaging*, 2016.

[9] D. S. Arnon, "Topologically reliable display of algebraic curves," *SIGGRAPH Comput. Graph.*, vol. 17, no. 3, pp. 219–227, July 1983. [Online]. Available: http://doi.acm.org/10.1145/964967.801152

[10] G. Taubin, "Distance approximations for rasterizing implicit curves," *ACM Trans. Graph.*, vol. 13, no. 1, pp. 3–42, Jan. 1994. [Online]. Available: http://doi.acm.org/10.1145/174462.174531

[11] A. Quint, "Scalable vector graphics," *IEEE MultiMedia*, vol. 10, no. 3, pp. 99–102, July 2003.

[12] C. Loop and J. Blinn, "Resolution independent curve rendering using programmable graphics hardware," in *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 1000–1009.

[13] M. J. Kilgard and J. Bolz, "Gpu-accelerated path rendering," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 172:1–172:10, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366191

[14] A. Orzan, A. Bousseau, P. Barla, H. Winnemöller, J. Thollot, and D. Salesin, "Diffusion curves: A vector representation for smooth-shaded images," *Commun. ACM*, vol. 56, no. 7, pp. 101–108, July 2013. [Online]. Available: http://doi.acm.org/10.1145/2483852.2483873

[15] D. Sỳkora, J. Buriánek, and J. Zára, "Sketching cartoons by example." in *SBM*, 2005, pp. 27–33.

[16] J.-D. Favreau, F. Lafarge, and A. Bousseau, "Fidelity vs. simplicity: A global approach to line drawing vectorization," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 120:1–120:10, July 2016. [Online]. Available: http://doi.acm.org/10.1145/2897824.2925946

[17] N. Hogan, "Impedance control: An approach to manipulation," in *American Control Conference, 1984*. IEEE, 1984, pp. 304–313.

[18] S. A. Schneider and R. H. Cannon, "Object impedance control for cooperative manipulation: Theory and experimental results," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 383–394, 1992.

[19] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Six-dof impedance control based on angle/axis representations," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 289–300, 1999.

[20] T. Wimbock, C. Ott, and G. Hirzinger, "Impedance behaviors for two-handed manipulation: Design and experiments," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 4182–4189.

[21] J. Lee, P. H. Chang, and R. S. Jamisola, "Relative impedance control for dual-arm robots performing asymmetric bimanual tasks," *IEEE transactions on industrial electronics*, vol. 61, no. 7, pp. 3786–3796, 2014.

[22] H. Samet, *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[23] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.