Robot Calligraphy using Pseudospectral Optimal Control in Conjunction with a Simulated Brush Model

Sen Wang, Jiaqi Chen, Xuanliang Deng, Seth Hutchinson, and Frank Dellaert[1]

Abstract— Chinese calligraphy is a unique form of art that has great artistic value but is difficult to master. In this paper, we make robots write calligraphy. Learning methods could teach robots to write, but may not be able to generalize to new characters. As such, we formulate the calligraphy writing problem as a trajectory optimization problem, and propose a new virtual brush model for simulating the real dynamic writing process. Our optimization approach is taken from pseudospectral optimal control, where the proposed dynamic virtual brush model plays a key role in formulating the objective function to be optimized. We also propose a stroke-level optimization to achieve better performance compared to the character-level optimization proposed in previous work. Our methodology shows good performance in drawing aesthetically pleasing characters.
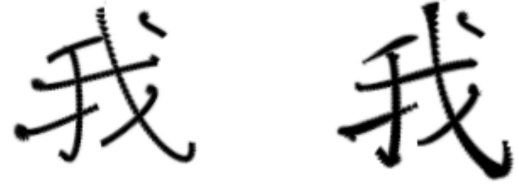
Index Terms— Motion Control of Manipulators, Optimization and Optimal Control, Manipulation Planning

I. INTRODUCTION

Making robots write beautiful calligraphy would be an exceptional feat since learning and mastering this art form takes humans years of practice. Chinese characters are complex and a calligraphic brush is difficult to manipulate properly. In this paper we aim to make a robot write Chinese characters by using a simulated brush model and pseudospectral optimal control methods to optimize for a trajectory.

Most relevant research on making actual robots create art adopts either a learning-based method or a trajectory optimization-based approach. The former often comes down to teaching by demonstration [1], [2], or self-correction [3]. By using learning one can skip the difficulty of modeling the behavior of a real calligraphy brush. However, learning methods also have a large training cost and may not generalize well to unseen characters. On the other hand, trajectory optimization-based methods do not face these problems. Here we *simulate* the writing behavior of an actual brush, and then search for an optimal trajectory for the robot to execute [4], [5]). However, most simulated brush models [5], [6] do not account for the complex ways a brush deforms during the writing process. Being able to capture this complexity has an important influence on the final performance.

We propose a trajectory optimization-based method based on principles from pseudospectral optimal control [7], [8], and also introduce a new dynamic virtual brush model to achieve fully automatic writing of Chinese characters, given a desired character unicode. Pseudospectral methods are generally used for optimizing continuous trajectories and controls,

[1]All authors are with the Georgia Institute of Technology, Atlanta, GA 30332, USA. Corresponding author: dellaert@cc.gatech.edu

(a) Simulated image from initial trajectory estimate

(b) Simulated image after trajectory optimization

(c) Robot executing trajectory

(d) Written image

Fig. 1: Summary of project workflow

but we assume the control is realized by the low-level inverse kinematics solvers on the robot. Different from previous work which optimizes the trajectory for the whole character at once [4], [5], we decompose the character into strokes and perform stroke-based optimization. Full character-based optimization can be computationally expensive and get stuck in local minima. We extract strokes and create initial trajectory estimates by leveraging the properties of vector-based character databases. The proposed virtual brush concentrates on the dynamic mechanisms of an actual calligraphic brush but has a simpler structure compared to previous work [9], [10]. As a baseline, we compare with a virtual brush model similar to Kwok et al. [4].

The primary contributions of this paper are:

1) We use pseudospectral methods to search for optimal writing trajectories to apply to calligraphy robots. Pseudospectral methods have natural modeling abilities for continuous trajectory optimization.
2) We design a new virtual brush model. Such a model is also able to simulate the real brush dynamics with higher accuracy, which leads to better optimized trajectories.
3) We exploit vector-based character libraries for easy stroke extraction and initialization for the stroke-level trajectory optimization.

II. RELATED WORK

This section concentrates on calligraphy robots, even though there are many other art forms that incorporate robotics [11], [12], [13], sculpture [14], graffiti [15], etc. Most algorithms on calligraphy robots using a brush pen can be categorized as learning-based methods or trajectory optimization-based methods.

A. Calligraphy robots using learning-based methods

Sun et al. [1], [16] propose to learn from demonstration. They invite calligraphers to write characters while holding the robot arm and record the robot joint positions to establish a mapping model for robot control. Mueller et al. [3] propose an iterative learning method by trial-and-learn. Some more advanced learning algorithms such as RNN [17], generative adversarial networks [18], deep reinforcement learning [19], local and global learning models [2], are also explored. These methods usually require many iterations of training to achieve good performance, which is inconvenient. Generalizing to new and complicated characters is difficult.

B. Virtual brush models

Virtual brush models are mainly used in trajectory optimization-based algorithms. We can divide virtual brush models into two categories: physics-based models and data-driven models.

Physics-based virtual brush models strive to simulate the physical dynamics of a real brush from experimental observation [20], [21], [22], [23] or physical laws [10], [24]. Strassmann [25] proposes an initial design featuring four basic parameters of a hairy brush. Wong et al. [26] propose to use a cone to represent the bundle of the brush and use the cross-section of the cone, an ellipse, to represent the footprint. Xu et al. [9] propose a virtual brush model with much detail and complex mechanisms obtained from approximations and assumptions. However, obtaining and fitting good parameters to complicated virtual brush models mentioned above is difficult, as such, we propose a virtual brush with easy structure to fit and implement, even making real-time trajectory optimization possible.

Data-driven virtual brush models are created from measuring and recording actual brush footprints. Kwok et al. propose a very simple virtual brush which draws droplet shapes with its size proportional to the writing height [4]. In their later work [6], they use a camera placed below the writing plane to collect footprints during the writing process. Lam et al. [5] define their writing mark as a polygon connected by eight points and fit their position parameters with the collected footprints. Considering the big calculation cost, Baxter et al. [27] build a deformation table and makes the calculation process much faster with complicated simulation effect.

C. Stroke extraction

Stroke extraction involves separating a character into its comprising strokes, and is difficult to do with good accuracy when analyzing just pixels of an image. There are three main categories of stroke extraction methods: skeleton-based [28], [29], region-based [30], and contour-based [31], [32], [33]. Most of these methods are complicated and cannot promise good accuracy between the extracted stroke and actual stroke. As such, we propose to use vector-based images as our character dataset, which, in contrast, provides a quick and accurate way to extract strokes.

D. Optimization methods

As mentioned above, Kwok et al. [4], [6] propose to use Bezier curves to represent stroke trajectories and find the optimized trajectory by minimizing differences between simulated images and desired character images. But their algorithms have not yet been applied to actual robots. Furthermore, their algorithms perform character-level optimization, which is computationally expensive and cannot handle complicated characters without needing human intervention. Lam et al. [5] propose to minimize the width difference of the strokes between reference images and a simulated image written by the virtual brush as it moves along the middle axis of each stroke. However, their method is sensitive to small variations in stroke images, and so the results suffer from a loss of smoothness.

III. METHODOLOGY OVERVIEW

An overview of our methodology is introduced by Fig. 2 on the next page. In terms of notation, we use $[x(t), y(t), z(t)]^T$ to denote the open-loop control trajectory given to the real brush. Given this trajectory, the end-effector is assumed to be able to track it with high accuracy.

IV. VIRTUAL BRUSH

There are two virtual brush models developed in this paper for simulating the writing process, a simple virtual brush and a dynamic virtual brush. Both generate good results, but the dynamic virtual brush is more sophisticated and better at closing the "sim2real" gap. The simple virtual brush is similar to Kwok et al. [4]'s work and is mainly used as a baseline.

A. Simple virtual brush model

Given the height z, of the brush, the simple virtual brush simulates drawing a circle with radius proportional to the height z. One important feature that distinguishes our simple virtual brush from others is that our simple brush establishes a continuous mapping from the trajectory $[x(t), y(t), z(t)]^T$ to a continuously-valued image rather than a discrete one, so that gradient information is not lost. The circle drawn by our simple virtual brush has a radial distribution following a sigmoid function:

$$r = \frac{\sqrt{(x_i - m)^2 + (y_i - n)^2}}{z_i} \tag{1}$$

$$p_{m,n} = 255 \frac{1}{1 + e^{-kr}} \tag{2}$$

where $p_{m,n}$ is the pixel value at the coordinates m, n of the generated image, k is a coefficient chosen to restrict the output pixel range.
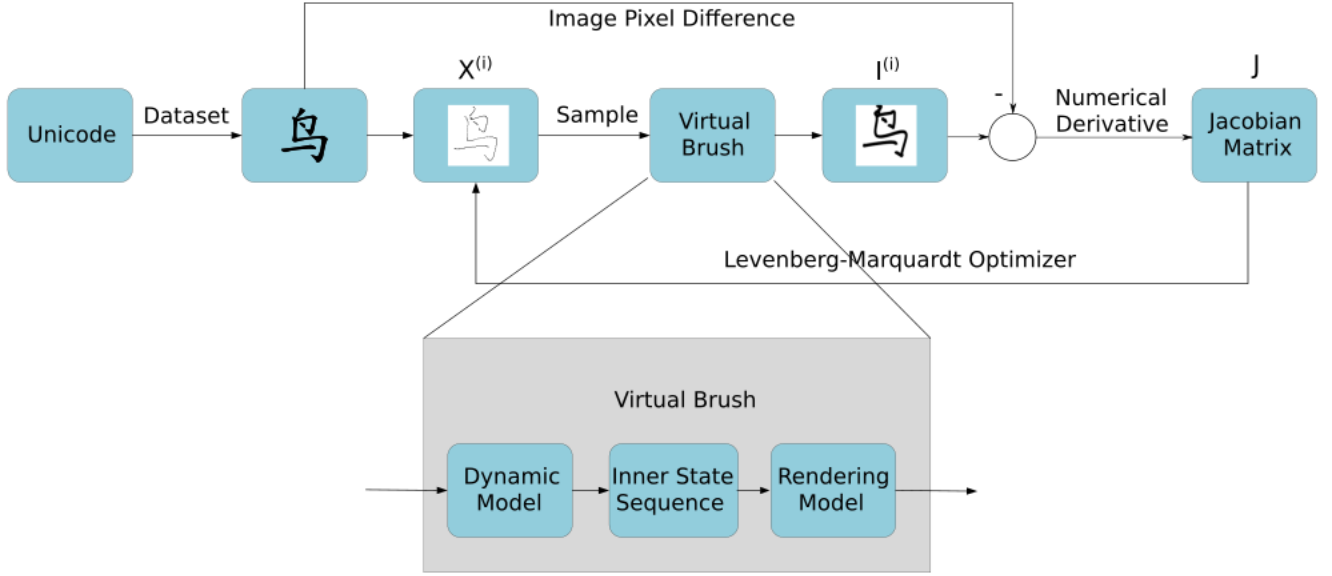
Fig. 2: Flowchart of the Methodology. First, given a unicode, find the corresponding image in the dataset; Second, extract the initial trajectories from the character image; Third, use the virtual brush to draw a simulated image, and then calculate the numerical derivative from the previously drawn image if it exists; Fourth, use the Jacobian matrix to improve the initial trajectory; Fifth, continue the loop until the stopping criteria are satisfied.

B. Dynamic virtual brush model

The dynamic virtual brush has two parts: a part that draws, and a part that updates the parameters of the brush. The drawing part describes the brush leaving a mark on paper depending on its parameters. The updating part is when the brush parameters are updated due to deformations when executing the control commands x_i, y_i, z_i.

1) Drawing Part: The dynamic virtual brush model has four state parameters describing its internal behavior: width w, drag d, offset o, and orientation θ. As shown in Fig. 3, width and drag define the size of a brush mark, offset simulates the deviation of the brush mark from the center of the vertical brush handle (this is due to the bend of the brush hairs when applying pressure), and orientation describes the direction of the writing mark (0 degrees means it points toward the right of the paper). The shape of the brush mark is defined by a quadratic curve. Some special brush characteristics like hair-splitting are generally avoided in real calligraphy, so we do not incorporate this.

2) Updating Brush Parameters Part: The updating process is based on the assumption that given one control command, (x_i, y_i, z_i), we can update the four parameters of the brush. In our experiments, we fit the relationships between the parameters, width w, drag d, and offset d, and z by collecting brush footprint data at varying writing heights. However, orientation o, is dependent on the direction the brush moves in. We observed that moving the brush in one direction will cause the tip of the brush to gradually change towards the opposite direction.
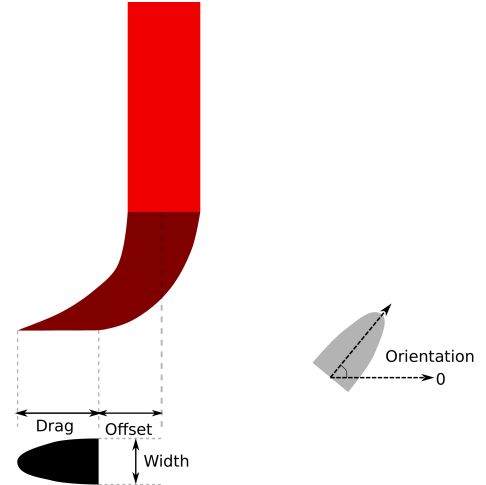

Fig. 3: Dynamic virtual brush model and its main components: width, drag, offset, direction

Fitting the parameters to brush height We measure the width, drag, and offset of writing marks left by a real calligraphy brush, then fit a linear relationship between the parameters, width w, drag d, and offset o, and the change in height of the brush, z.

Adding inertia to parameters We introduce inertia to parameter updating because in reality, the brush parameters are not updated fully or instantaneously; brush deformations happen gradually and steadily. Given two adjacent control commands (x_i, y_i, z_i) and $(x_{i+1}, y_{i+1}, z_{i+1})$, the parameters

are updated while taking inertia into account:

$$w_{i+1} = w_i k_{inertia} + Width(z_i)(1 - k_{inertia}) \qquad (3)$$

$$d_{i+1} = d_i k_{inertia} + Drag(z_i)(1 - k_{inertia}) \qquad (4)$$

$$o_{i+1} = o_i k_{inertia} + Offset(z_i)(1 - k_{inertia}) \qquad (5)$$

where $Width(z_i)$, $Drag(z_i)$, and $Offset(z_i)$ respectively calculate a new width, drag, and offset, given control command z_i, and $k_{inertia}$ is the inertia value. The inertia value is inversely proportional to the distance between two adjacent points or it loses effect if we use densely sampled points.

V. Trajectory Optimization

A. Review of Pseudospectral Optimal Control

The calligraphy problem in this paper is formulated as a trajectory optimization problem, and in particular we adopt some of the machinery from pseudospectral optimal control methods. As such, we briefly review pseudospectral optimal control (PSOC) methods in this section, following the exposition from Fahroo et al. [8].

A simplified version of an optimal control problem is formulated as follows. Given a cost function C and a model of the system dynamics f

$$C = g(x, u) \qquad (6)$$
$$\dot{x}(t) = f(x(t), u(t), t)) \qquad (7)$$

the objective is to find the optimal control sequence $u(t)$, that minimizes the cost function C. Above $x(t)$ represents the system's state trajectory.

The basic idea in PSOC is to approximate the control trajectory $u(t)$ and the state trajectory $x(t)$ by a polynomial curve with unknown parameters, thereby transform the original problem into a nonlinear programming problem. To this end, pseudospectral methods choose a specific set of points from the curve for interpolation. For example, in the case of Chebyshev pseudospectral methods, the interpolation points are given as the Chebyshev-Gauss-Lobatto (CGL) points [34]:

$$t_k = cos(\frac{\pi k}{N}), \qquad k = 0, ..., N \qquad (8)$$

To recover the state $x(t)$ at any arbitrary time t one can use barycentric interpolation, , i.e.,

$$x(t) \approx (\sum_k \frac{w_k}{t - t_k} x_k) / \sum_k \frac{w_k}{t - t_k} \qquad (9)$$

where

$$w_k = \begin{cases} (-1)^k/2 & k = 0 \text{ or } k = N \\ (-1)^k & \text{otherwise} \end{cases} \qquad (10)$$

We can now express the original dynamics equation with an approximation, where additionally the objective function Eq. 6 can be discretized if necessary. The original problem

is thus transformed to minimize the cost C with respect to the two coefficient vectors X, U,

$$X = (x_0, ..., x_N), \ U = (u_0, ..., u_N) \qquad (11)$$

representing the values of the state and controls, respectively, at the CGL points.

B. PSOC for Calligraphy

Below we apply these methods to trajectory optimization for open-loop control trajectories of a robot end-effector, with the goal of faithfully reproducing Chinese characters. PSOC methods are generally used for collocated optimal control where the system dynamics are enforced through specialized components of the cost function. However, here we assume that the control is realized by the low-level inverse kinematics solvers on the robot.

The optimization for a character is decomposed into a series of trajectory optimization problems corresponding to the different strokes of the character. This greatly simplifies the process and is also more computationally efficient. In this we are helped by the existence of vector-based character databases in which characters are stored decomposed in their individual strokes.
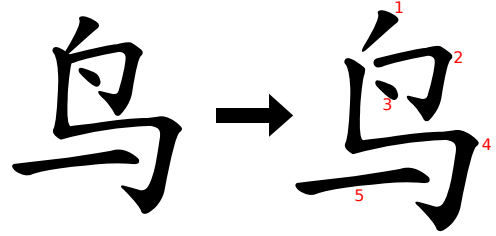
C. Stroke extraction

Fig. 4: The character 'bird', pronounced 'niao' and its extracted strokes

To obtain reference images for each stroke, as well as initialize the nonlinear optimization, we exploit the existence of vector-based character databases. Vector-based images are advantageous because they store individual strokes as Bezier curves, which are themselves continuous polynomial curves defined by a set of control points. Because each stroke is stored separately, it makes the stroke extraction trivial. In our case, the dataset we choose is a Scalable Vector Graphics (SVG) dataset from MakeMeHanzi [35], and an example of extracted strokes can be seen in Fig. 4.

D. Stroke trajectory representation

The stroke trajectories will be used as the open-loop control trajectories for the robot to draw strokes, and we represent them as three-dimensional trajectories of the end-effector. Each of the x, y, z components are separately represented as a 1-dimensional Chebyshev polynomial curves. The trajectories X with its three dimensions x, y, z are expanded

by interpolating the values at the Chebyshev-Gauss-Lobatto (CGL) points given by Eq. 8 and Eq. 10:

$$X(t) \approx (\sum_k \frac{w_k}{t - t_k} X_k) / \sum_k \frac{w_k}{t - t_k} \qquad (12)$$

Hence, the decision variables are the combination of the three sets of CGL points in the x, y, z dimensions:

$$X = (x_0, ..., x_N; y_0, ..., y_N; z_0, ..., z_N) \qquad (13)$$

E. Optimization for the stroke trajectories

The objective function $C^{(d)}$ for the dth stroke optimization is to minimize the sum-squared pixel difference between a reference image $I^{(d)}$ for the stroke, and an image produced by simulating the drawing process:

$$\min_{X^{(d)}} C^{(d)} = \{\| V(X^{(d)}) - I^{(d)} \|^2; \theta_{last}^{(d-1)}, S_{init}\} \qquad (14)$$

Above $V(.)$ is a function that represents the virtual brush simulation, taking a trajectory and drawing an image according to the given trajectory, parameterized by the pseudospectral values $X^{(d)}$ for stroke d. Also given is the initial brush state S_{init} when the brush starts writing the dth stroke. The reference image of the dth stroke, $I^{(d)}$, is converted to an image representation from the vector-based stroke. A much more dense even sampling is performed in V to draw a stroke looks continuous. $\theta_{last}^{(d-1)}$ is the last direction parameter after writing the previous stroke, which is assumed to be the initial direction parameter for the current stroke. It is not included for the simple virtual brush. A scale parameter s in $X^{(d)}$ is used to convert the z_i to the same magnitude as x_i and y_i for the convenience of optimization.

Many nonlinear optimization methods can be used to minimize Eq. 6. In our case, we use the Levenberg-Marquardt (LM) algorithm, which is a second-order trust-region method, which switches between a gradient-based search and a second-order Gauss-Newton update:

$$[\mathbf{J}^T\mathbf{J} + \lambda \mathbf{diag}(\mathbf{J}^T\mathbf{J})]\delta = \mathbf{J}^T[V(c_i) - I_i] \qquad (15)$$

The Jacobian matrix $\mathbf{J}$ is calculated using numerical differentiation at each iteration. We utilize the GTSAM library [36], [37] to perform the optimization. GTSAM was originally created to solve simultaneous localization and mapping problems, but has been used in many different contexts since, including motion planning [38], [39].

F. Trajectory Initialization Estimates

We propose to use the skeleton of a stroke as our initialization for the x and y coordinates of our trajectory, while the initial z coordinates are set to a fixed value. From our observations, when people write calligraphy, they generally make the brush approximately follow the skeleton of the stroke while varying the height of the brush. Because we start from a vector-based representation, extracting an initial trajectory for the individual strokes is much simplified. Even when starting from images, there are many good image-based skeleton extraction algorithms, e.g. the Chordal Axis Transform [40], [41], an example result of which is shown
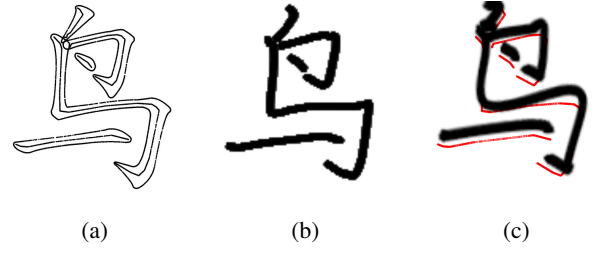

Fig. 5: (a) The character 'bird' with its skeleton; (b) Initial image from simple virtual brush; (c) Initial image from dynamic virtual brush with the trajectory (red)

in Fig. 5a. In our case, we use the "animation path" provided by the dataset as the initial estimation for the 2D x_i and y_i sequence for simplicity. CGL points are sampled on each skeleton path to obtain the pseudospectral representation. Generating an easy estimate for z is not intuitive, and so we just initialize it with a constant sequence.

Given this initial trajectory, we can simulate the image formation process using both the simple virtual brush and the dynamic virtual brush, as illustrated in Fig. 5b and Fig. 5c respectively. The written mark from the dynamic virtual brush is different from its given trajectory, which is ignored by most previous research.
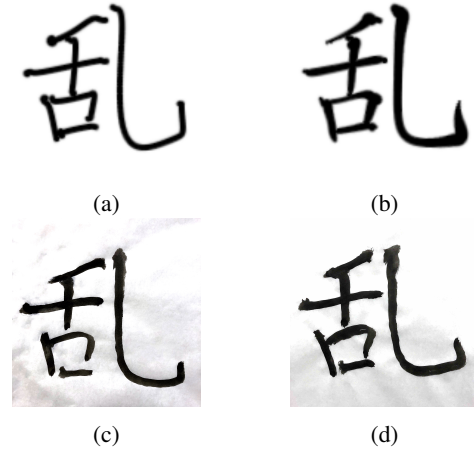
VI. RESULTS

Fig. 6: Character 'messy', pronounced 'luan' (a) Initial trajectory; (b) Trajectory after optimization with simple virtual brush; (c) Written result following simple brush optimization; (d) Written result following *dynamic* brush optimization

The dynamic virtual brush model yields better written results than the simple virtual brush model, although the latter can produce very high-quality simulated images. Fig. 6 shows a comparison between the dynamic and the simple virtual brush model. In panel (b), the simulated image produced by the simple model is very good. However, because the deformation of the real brush is not modeled, the results on the real robot are of lower quality.
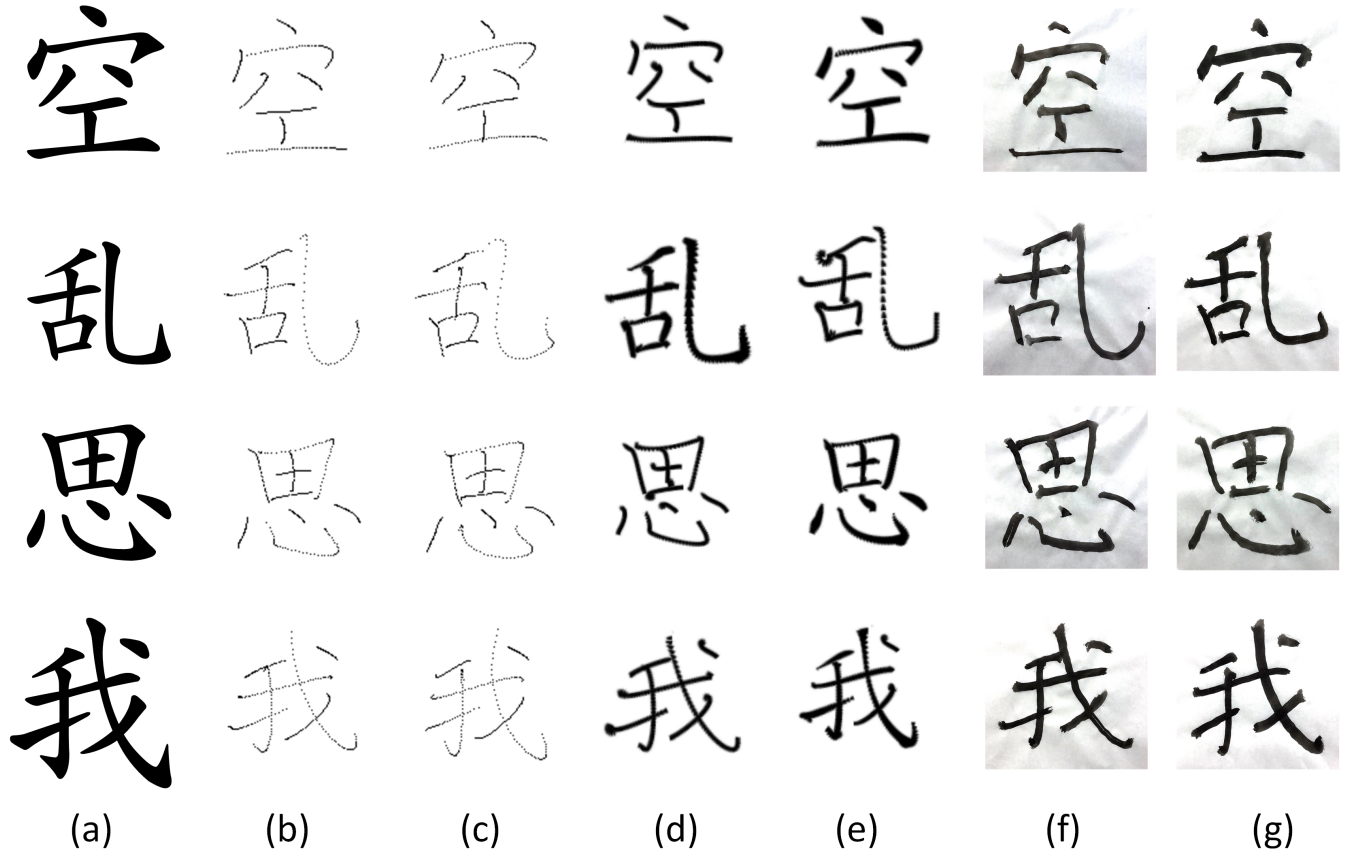
Fig. 7: The optimization of different characters (From top down, 'kong', 'luan', 'si', 'wo' meaning 'empty', 'messy', 'think', 'me'). (a) The original character pictures from the dataset; (b) Initial trajectory estimation; (c) The trajectory obtained from optimization; (d) Simulated image drawn by the virtual brush using the initial trajectory; (e) Simulated image drawn by the virtual brush using an *optimized* trajectory; (f) Written image following initial trajectories; (g) Written image following the *optimized* trajectories

Hence, although minimizing cost between the simulation images and the original character images is used as the objective function, generating too small of a simulation error will lead to over-fitting, and this is true for both models. In other words, the capacity of the virtual brush model to model the actual brush sets the performance limit of the project, and using optimization methods to surpass the limit may lead to bad results. From our experience, an average pixel error between 8% and 16% of all pixels is usually enough to generate good written results.

Another possible cause of over-fitting is the degree of the polynomial parameterization used to represent the character strokes. Currently we choose a degree for every character once the simulation error of Eq. 14 falls into the range $[8\% - 16\%]$. But such a method does not promise to find the global optimum, but could provide a more practical and robust solution for the robot to execute. In the future, we can tune the degree to better fit each stroke.

In Fig. 7, we present the results of our approach, including photographs of characters that have been drawn by a Fetch robot in our lab. We show results for four different Chinese characters. Both the simulation and written images before and after optimization are shown for easy comparison. From the figure we can see that the optimization achieves good performance for simulated images. However, because of a "sim2real" gap in the virtual brush model, the proposed method still has a ways to go in terms of approaching the smoothness and definition of detail displayed in the reference images, rendered from the vector-based character database.

VII. DISCUSSION AND FUTURE WORK

In this paper, we presented a trajectory optimization method to make robots write calligraphy, searching for open-loop control trajectories by minimizing a simulation against a reference image. The proposed dynamic virtual brush simulation yields good results in simulation and in turn produces reasonable open-loop control trajectories for a real robot. However, from the results it is clear that the sim2real gap has not been fully closed, and it is an open question whether a closed loop strategy will every yield master-level calligraphy. Hence, in future work we plan to push on both better brush models and the possibility of feedback control around optimized trajectories.

REFERENCES

[1] Y. Sun, H. Qian, and Y. Xu, "Robot learns chinese calligraphy from demonstrations," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4408–4413, 2014.

[2] A. Kotani and S. Tellex, "Teaching robots to draw," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4797–4803, 2019.

[3] S. Mueller, N. Huebel, M. Waibel, and R. D'Andrea, "Robotic calligraphy — learning how to write single strokes of chinese and japanese characters," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1734–1739, 2013.

[4] K.-W. Kwok, S. M. Wong, K. W. Lo, and Y. Yam, "Genetic algorithm-based brush stroke generation for replication of chinese calligraphic character," *2006 IEEE International Conference on Evolutionary Computation*, pp. 1057–1064, 2006.

[5] J. H. M. Lam and Y. Yam, "Stroke trajectory generation experiment for a robotic chinese calligrapher using a geometric brush footprint model," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2315–2320, 2009.

[6] K.-W. Kwok, K. W. Lo, S. M. Wong, and Y. Yam, "Evolutionary replication of calligraphic characters by a robot drawing platform using experimentally acquired brush footprint," *2006 IEEE International Conference on Automation Science and Engineering*, pp. 466–471, 2006.

[7] G. N. Elnagar and M. A. Kazemi, "Pseudospectral chebyshev optimal control of constrained nonlinear dynamical systems," *Computational Optimization and Applications*, vol. 11, pp. 195–217, 1998.

[8] F. Fahroo and I. Ross, "Direct trajectory optimization by a chebyshev pseudospectral method," *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, vol. 6, pp. 3860–3864 vol.6, 2000.

[9] S. Xu, F. C. M. Lau, and Y. Pan, "A computational approach to digital chinese painting and calligraphy," in *A Computational Approach to Digital Chinese Painting and Calligraphy*, 2009.

[10] N. S.-H. Chu and C.-L. Tai, "Real-time painting with an expressive virtual chinese brush," *IEEE Computer Graphics and Applications*, vol. 24, pp. 76–85, 2004.

[11] S. Kudoh and K. Ogawara, "Painter robot : Manipulation of paintbrush by force and visual feedback," in *Painter Robot : Manipulation of Paintbrush by Force and Visual Feedback*, 2007.

[12] Y. Lu, J. H. M. Lam, and Y. Yam, "Preliminary study on vision-based pen-and-ink drawing by a robotic manipulator," *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 578–583, 2009.

[13] L. Scalera, S. Seriani, A. Gasparetto, and P. Gallina, "Watercolour robotic painting: a novel automatic system for artistic rendering," *Journal of Intelligent and Robotic Systems*, pp. 1–16, 2018.

[14] X. Niu, J. Liu, L. Sun, Z. Liu, and X. Chen, "Robot 3d sculpturing based on extracted nurbs," *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1936–1941, 2007.

[15] Y. Jun, G. Jang, B.-K. Cho, J. Trubatch, I. Kim, S.-D. Seo, and P. Y. Oh, "A humanoid doing an artistic work - graffiti on the wall," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1538–1543, 2016.

[16] Y. Sun and Y. Xu, "A calligraphy robot — callibot: Design, analysis and applications," *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 185–190, 2013.

[17] K. Sasaki, K. Noda, and T. Ogata, "Visual motor integration of robot's drawing behavior using recurrent neural network," *Robotics and Autonomous Systems*, vol. 86, pp. 184–195, 2016.

[18] F. Chao, J. Lv, D. Zhou, L. Yang, C.-M. Lin, C. Shang, and C. Zhou, "Generative adversarial nets in robotic chinese calligraphy," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1104–1110, 2018.

[19] R. Wu, W. Fang, F. Chao, X. Gao, C. Zhou, L. Yang, C.-M. Lin, and C. Shang, "Towards deep reinforcement learning based chinese calligraphy robot," *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 507–512, 2018.

[20] S. Xu, M. Tang, F. C.-M. Lau, and Y. Pan, "A solid model based virtual hairy brush," *Comput. Graph. Forum*, vol. 21, pp. 299–308, 2002.

[21] S. Xu, F. C. M. Lau, F. Tang, and Y. Pan, "Advanced design for a realistic virtual brush," *Comput. Graph. Forum*, vol. 22, pp. 533–542, 2003.

[22] S. Saito and M. Nakajima, "3d physics-based brush model for painting," in *SIGGRAPH Abstracts and Applications*, 1999.

[23] J. Lee, "B simulating oriental black-ink painting," in *B Simulating Oriental Black-ink Painting*, 1999.

[24] N. S.-H. Chu and C.-L. Tai, "An efficient brush model for physically-based 3d painting," *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.*, pp. 413–421, 2002.

[25] S. Strassmann, "Hairy brushes," in *SIGGRAPH*, 1986.

[26] H. T. F. Wong and H. H.-S. Ip, "Virtual brush: a model-based synthesis of chinese calligraphy," *Computers and Graphics*, vol. 24, pp. 99–113, 2000.

[27] W. V. Baxter and N. K. Govindaraju, "Simple data-driven modeling of brushes," in *SI3D*, 2010.

[28] C.-L. Liu, I.-J. Kim, and J. H. Kim, "Model-based stroke extraction and matching for handwritten chinese character recognition," *Pattern Recognition*, vol. 34, pp. 2339–2352, 2001.

[29] J. Zeng, W. Feng, L. Xie, and Z.-Q. Liu, "Cascade markov random fields for stroke extraction of chinese characters," *Inf. Sci.*, vol. 180, pp. 301–311, 2010.

[30] R. Cao and C. L. Tan, "A model of stroke extraction from chinese character images," *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 4, pp. 368–371 vol.4, 2000.

[31] C. Lee and B. Wu, "A chinese-character-stroke-extraction algorithm based on contour information," *Pattern Recognition*, vol. 31, pp. 651–663, 1998.

[32] Y. Sun, H. Qian, and Y. Xu, "A geometric approach to stroke extraction for the chinese calligraphy robot," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3207–3212, 2014.

[33] F. Chao, Y. Huang, C.-M. Lin, L. Yang, H. Hu, and C. Zhou, "Use of automatic chinese character decomposition and human gestures for chinese calligraphy robots," *IEEE Transactions on Human-Machine Systems*, vol. 49, pp. 47–58, 2019.

[34] P. T. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," *2012 IEEE International Conference on Robotics and Automation*, pp. 2088–2095, 2012.

[35] "Makemeahanzi." https://github.com/skishore/ makemeahanzi. Accessed: 2019-09-10.

[36] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," in *Factor Graphs and GTSAM: A Hands-on Introduction*, 2012.

[37] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, pp. 1–139, 2017.

[38] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," vol. 37, no. 11, pp. 1319–1340, 2018.

[39] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "Steap: simultaneous trajectory estimation and planning," *Autonomous Robots*, pp. 1–20, 2018.

[40] E. Y. Y. Zhang and P. S.-P. Wang, "A parallel thinning algorithm with two-subiteration that generates one-pixel-wide skeletons," *Proceedings of 13th International Conference on Pattern Recognition*, vol. 4, pp. 457–461 vol.4, 1996.

[41] K. W. Lo, K.-W. Kwok, S. M. Wong, and Y. Yam, "Brush footprint acquisition and preliminary analysis for chinese calligraphy using a robot drawing platform," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5183–5188, 2006.