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

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

Abstract— Chinese calligraphy is a unique art form with great artistic value but difficult to master. In this paper, we formulate the calligraphy writing problem as a trajectory optimization problem, and propose an improved virtual brush model for simulating the real writing process. Our approach is inspired by pseudospectral optimal control in that we parameterize the actuator trajectory for each stroke as a Chebyshev polynomial. The proposed dynamic virtual brush model plays a key role in formulating the objective function to be optimized. Our approach shows excellent performance in drawing aesthetically pleasing characters, and does so much more efficiently than previous work, opening up the possibility to achieve real-time closed-loop control.

Index Terms— Motion and Path Planning; Optimization and Optimal Control; Modeling, Control, and Learning for Soft Robots

I. INTRODUCTION

Making robots write beautiful calligraphy is difficult, as learning and mastering this art form takes humans years of practice. Chinese characters are complex and a calligraphy brush is difficult to manipulate properly. Hence, making a robot achieve comparable results is a worthwhile endeavour, both to expand robot's capabilities into art as well as push our understanding on how to manipulate deformable brushes.

Most relevant research in this area adopts either a learning-based method or an approach based on trajectory optimization. The former includes learning from demonstration [1], [2], or learning from visual feedback given a reference image [3]. By using learning, one can eliminate the difficulty of modeling the behavior of a real calligraphy brush. However, learning methods have a large training cost and may not generalize well to previously unseen characters. On the other hand, trajectory optimization methods do not face these problems: they *simulate* the writing behavior of an actual brush, and then search for an optimal trajectory for the robot to execute [4], [5]). The difficulty there is that most simulated brush models [5], [6] do not account for the complex ways a brush deforms during the writing process. Being able to capture the complexity of a deformable brush has an important influence on the final performance.

In this paper, we propose a novel virtual brush model intended to capture the dynamics of an actual calligraphy brush, which is an improved version based on an earlier virtual brush model by Kwok *et al.* [4], which we use as a baseline. We improve on this model by explicitly modeling the brush dynamics, while retaining a much simpler structure compared to the elaborate models in other work [7], [8]. We

[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

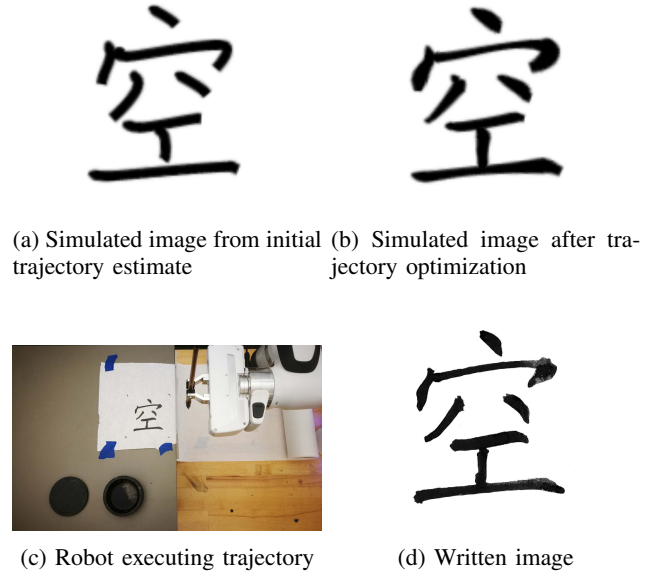(c) Robot executing trajectory

(d) Written image

Fig. 1: Summary of project workflow

then use this model to simulate the writing process for a given open-loop trajectory, which allows us to then optimize for the trajectory parameters.

To do so, our second contribution is an efficient trajectory optimization-based method based on *pseudospectral optimal control* [9], [10] that achieves fully automatic writing of Chinese characters given a character's unicode. Pseudospectral methods are based on Legendre or Chebyshev polynomials, which are excellent at approximating and representing continuous trajectories and controls. Our method converges quickly and efficiently even when using many control points.

Finally, we exploit the existence of vector-based character databases such that, given a character's unicode, we immediately have access to the individual character strokes as well as the stroke order. We optimize for each stroke separately, and to obtain even faster convergence we initialize the trajectory for each stroke by a stroke skeleton which is also available in these databases.

Our proposed continuous, nonlinear optimization framework differs from previous work [4], [5], [6] which uses heuristic optimization methods known for slow convergence rates and a high computational cost. Hence, our method can potentially be used in a closed-loop control system, which we intend to pursue in future work. The proposed end-to-end system displays excellent writing performance, although there is definitely room for growth.

II. RELATED WORK

There is much work in the robotics community that aims to create art using robots, such as painting and drawing [11], [12], [13], sculpture [14], graffiti [15], etc. Below we focus our discussion on robot calligraphy, most algorithms using a calligraphy brush can be categorized as either learning or trajectory optimization-based.

A. Calligraphy robots using learning-based methods

Examples of simple learning-based methods include Sun *et al.*'s learning from demonstration [16] [1], Mueller *et al.*'s trial-and-learn iterative learning method [3]. Some more advanced learning algorithms such as RNN [17], generative adversarial networks [18], deep reinforcement learning [19], and local and global learning models [2] were also explored. However, these methods require many iterations of training to achieve good performance, and have difficulty generalizing to new and/or complicated characters.

B. Virtual brush models

Trajectory optimization-based algorithms mainly rely on *virtual brush models*, which can be divided 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 [8], [24]. Strassmann [25] proposes an initial design featuring four basic parameters of a hairy brush based on bristles. 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 of the brush. Xu *et al.* [7] propose a detailed virtual brush model for use in synthetic imagery, with good results in creating realistic-looking simulations. However, obtaining and fitting good parameters to the models above is difficult. As such, we propose a virtual brush with an easy structure to fit and implement, geared towards real-time trajectory optimization.

Data-driven virtual brush models are created by measuring and recording actual brush footprints on writing surfaces. Kwok *et al.* propose a simple virtual brush which draws droplet shapes whose size is 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. With an eye for efficiency, Baxter *et al.* [27] build a deformation table to speed up the associated computation while still being able to simulate complex effects.

C. Stroke extraction

Stroke extraction involves separating a character into its comprising strokes and is difficult to do with good accuracy when analyzing only the 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 complex and may not achieve good results as compared to the ground-truth stroke segmentation. As such, we propose using vector-based character database, which provides a quick and accurate way to extract strokes, as well as stroke order.

D. Optimization methods

As mentioned above, Kwok *et al.* [4], [6] propose using genetic algorithm for direct character-level optimization guided by their brush models. However, their reported computation times are long, and the strokes have to be manually separated. Lam *et al.* [5] 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. One of our main contributions below is the use of methods from optimal control to achieve the desired stroke trajectories.

III. METHODOLOGY OVERVIEW

An overview of our methodology is given in Fig. 2 on the next page. Given a character's unicode we find the corresponding SVG image in a vector-based database, which already comes segmented into the different strokes. For each stroke, we compute initial trajectories from the stroke skeletons, and then optimize from there until our simulated image matches the reference images rendered from the SVG file. At the core of the simulation is a virtual brush model, which we discuss at length in the section below, followed by a discussion of our optimization framework.

IV. TWO VIRTUAL BRUSH MODELS

There are two virtual brush models that we have experimented with in order to simulate the calligraphy 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.

A. Simple virtual brush model

The simple virtual brush model is similar to Kwok *et al.*'s work [4] and is used as a baseline comparison in our results. Given the height z of the brush, this model simply fills a circular patch with simulated ink, where the radius of the circle is proportional to z. Note that we align the task coordinate frame for the robot and brush such that the $z = 0$ plane corresponds to the plane of the paper and z increases as the brush is moved towards the paper. Note that Kwok *et al.* also proposed using more complex templates (such as droplets), but we get that behavior for free by modeling the dynamics in our model below.

B. Dynamic virtual brush model

The dynamic virtual brush model has two components: a drawing component, and a dynamic update component. The drawing component describes how the brush leaves a mark on paper depending on its parameters. The updating component then describes how the brush parameters are updated due to deformations when executing an open loop trajectory $[x(t), y(t), z(t)]^T$.
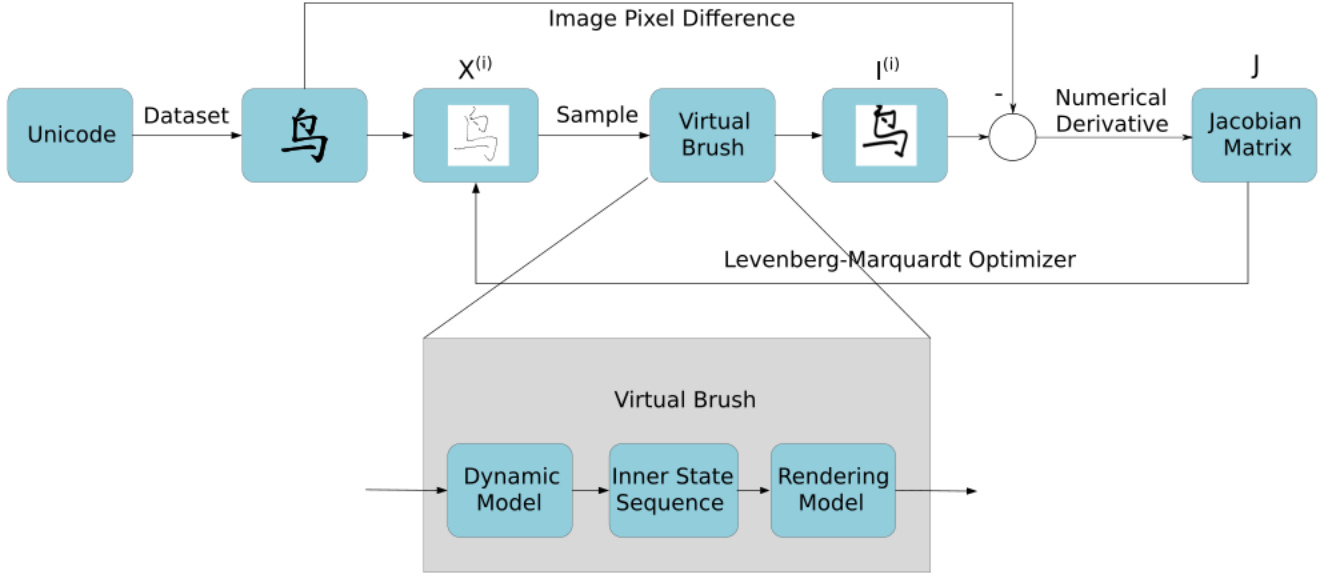
6697

Fig. 2: Flowchart of the Methodology. (1) Given a unicode, find the corresponding image in the database; (2) Extract the initial trajectories from the character image; (3) Use the virtual brush to draw a simulated image, and then compute the numerical derivative from the previously drawn image if it exists; (4) Use the Jacobian matrix to improve the initial trajectory; (5) Continue the loop until the stopping criteria are satisfied.
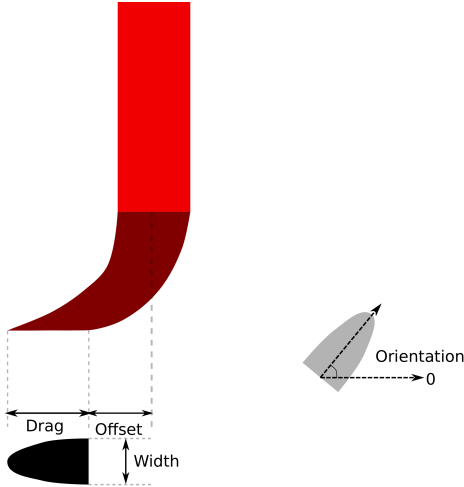

Fig. 3: Dynamic virtual brush model and its parameters. The root location of the brush mark is defined as the middle of the flat end and is determined by the end-effector position X, the offset o, and the orientation θ, whereas the shape of the brush mark is determined by the width w and drag d.

1) Drawing Component: The dynamic virtual brush model has four state parameters governing its drawing footprint: width w, drag d, offset o, and orientation θ. As shown in Fig. 3, width w and drag d define the size of a brush mark, the offset o simulates the deviation of the brush mark from the center of the vertical brush handle due to bending of the brush hairs when applying pressure, and the

orientation θ describes the direction of the writing mark. Together with the end-effector coordinates $X \doteq (x, y, z)$, these four parameters define the 7-dimensional brush state $\mathcal{X} \doteq (x, y, z, w, d, o, \theta)$.

Given the state $\mathcal{X}$, our model then computes the "root" location of the brush mark as

$$X^{root} = \pi(X) - o\mathcal{V}(\theta) \qquad (1)$$

where $\pi(X)$ is the projection of X in the $z = 0$ plane and $\mathcal{V}$ transforms the scalar orientation θ into a three-dimensional unit vector in that same plane. The shape of the brush mark is parameterized by a quadratic curve which is uniquely determined by the width w and the drag d. Some more specialized brush behavior such as intentional hair-splitting is reserved for more artistic calligraphy styles and is not modeled here.

2) Dynamic Update Component: We adopt a simple discrete state dynamical model described at a high level by

$$\mathcal{X}_{i+1} = f(\mathcal{X}_i, \mathcal{U}_i) \qquad (2)$$

where $\mathcal{U}_i = (v_{xi}, v_{yi}, v_{zi})$ is the control at the ith time instance. The x, y, and z components of the state $\mathcal{X}_0$ at $i = 0$ are initialized from the first point in the open loop trajectory. Initializing the other components is more involved and is discussed in more detail in Section V.
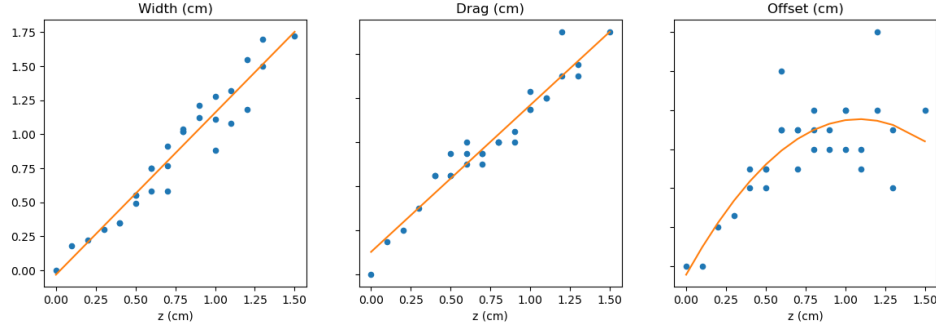
Fig. 4: The data collected for fitting the model parameters for our new virtual brush model (2 outliers were removed).

The dynamics model is given below and was formulated based on careful observation and experimentation:

$$x_{i+1} = x_i + v_{xi}\Delta t \qquad (3)$$
$$y_{i+1} = y_i + v_{yi}\Delta t \qquad (4)$$
$$z_{i+1} = z_i + v_{zi}\Delta t \qquad (5)$$
$$w_{i+1} = w_i K_w + Width(z_{i+1})(1 - K_w) \qquad (6)$$
$$d_{i+1} = d_i K_d + Drag(z_{i+1})(1 - K_d) \qquad (7)$$
$$o_{i+1} = \min(Offset(z_{i+1}),\ Offset'(\boldsymbol{X}_i^{root}, \boldsymbol{X}_{i+1})) \qquad (8)$$
$$\theta_{i+1} = \mathcal{V}^{-1}(\boldsymbol{X}_{i+1}^{root} - \pi(\boldsymbol{X}_{i+1})) \qquad (9)$$

Above $Width(z_{i+1})$ and $Drag(z_{i+1})$ respectively compute a new value for width and drag, given the updated end-effector height z_{i+1}. The constants K_w and K_d model a small amount of inertia in those parameters - we have used 0.02 in our experiments - modeling the fact that the deformations happen gradually and steadily.

The orientation and offset updates are more complicated given that they are governed by friction, modeling the fact that the part of the brush in contact with the paper resists movement. In particular, the function $Offset(z_{i+1})$ computes a value for the offset o when friction is overcome, and $Offset'(\boldsymbol{X}_i^{root}, \boldsymbol{X}_{i+1})$ computes the value of o that makes the brush stay put. The min operator ensures that the brush stays put until it "snaps" to the smaller value $Offset(z_{i+1})$ when the end-effector $\boldsymbol{X}_{i+1}$ moves too far out. The orientation θ_{i+1} is accordingly updated, as $\boldsymbol{X}_{i+1}^{root}$ is a function of o_{i+1}.

3) System Identification: The width, drag, and offset functions above were experimentally determined in a separate calibration phase. We measured the width, drag, and offset of writing marks left by a real calligraphy brush, then fit a polynomial relationship between the parameters, width w, drag d, and offset o, and the change in height of the brush, z. We fit the relationships between the parameters width w, drag d, and offset d w.r.t. z by collecting brush footprint data at varying writing heights, shown in Figure 4. The figure shows that width and drag behave almost linearly, but that the offset o has a local maximum. We observe that moving the brush in one direction will cause the tip of the brush to gradually move towards the opposite direction.

V. TRAJECTORY OPTIMIZATION

A. Review of Pseudospectral Optimal Control

Below we formulate the calligraphy writing process as a trajectory optimization problem, and in particular we adopt some of the machinery from pseudospectral optimal control (PSOC) methods. As such, we briefly review pseudospectral optimal control methods in this section, closely following the exposition by Fahroo *et al.* [10].

A simplified version of an optimal control problem can be stated in terms of a cost function C and system dynamics f

$$C = g(x, u) \qquad (10)$$
$$\dot{x}(t) = f(x(t), u(t), t)), \qquad (11)$$

where 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 transforming 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 used are the Chebyshev-Gauss-Lobatto (CGL) points [34]:

$$t_k = \cos\left(\frac{\pi k}{N}\right), \qquad k = 0, ..., N \qquad (12)$$

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

$$x(t) \approx (\sum_k \frac{w_k}{t - t_k} x_k)/\sum_k \frac{w_k}{t - t_k} \qquad (13)$$

where

$$w_k = \begin{cases} (-1)^k/2 & k = 0 \text{ or } k = N \\ (-1)^k & \text{otherwise} \end{cases} \qquad (14)$$

We can now express the original dynamics equation with an approximation, where the objective function (10) can be discretized if necessary. The original problem is thus

6699

transformed to minimizing the cost C with respect to the two coefficient vectors X, U:

$$X = (x_0, ..., x_N), \ U = (u_0, ..., u_N) \qquad (15)$$

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, because our control input is simply the trajectory velocity, which can be calculated exactly from the end-effector trajectory, we have no need for separate control parameters U.

The optimization for a character is decomposed into a series of trajectory optimization problems corresponding to individual 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 as their decomposed individual strokes, as described next.
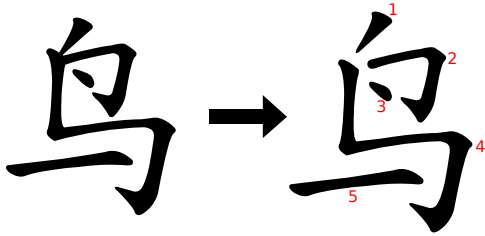
C. Stroke extraction

Fig. 5: 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. This makes the stroke extraction a trivial task. In our case, the database we choose is a Scalable Vector Graphics (SVG) database from MakeMeHanzi [35], and an example of extracted strokes can be seen in Fig. 5.

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 1-dimensional Chebyshev polynomial curves. The trajectories $\boldsymbol{X}$ with three dimensions x, y, z are expanded

by interpolating the values at the Chebyshev-Gauss-Lobatto (CGL) points given by Eq. 13 and Eq. 14:

$$\boldsymbol{X}(t) \approx \left(\sum_k \frac{w_k}{t - t_k} \boldsymbol{X}_k\right) / \sum_k \frac{w_k}{t - t_k} \qquad (16)$$

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

$$X = (x_0, ..., x_N; y_0, ..., y_N; z_0, ..., z_N) \qquad (17)$$

E. Optimization for the stroke trajectories

The objective function C_s for stroke s minimizes the sum-squared pixel difference between the image $V(X_s)$ produced by simulating the drawing process, and a reference image I_s created from the SVG representation. The optimal trajectory parameters X_s^* for stroke s are obtained as

$$X_s^* = \underset{X_s}{\arg\min} \, C_s(X_s) \qquad (18)$$

$$= \underset{X_s}{\arg\min} \{\| V(X_s) - I_s \|^2 + \sum_k \beta_k \|z_k\|^2\} \qquad (19)$$

Above, $V(\cdot)$ is a function that represents the virtual brush simulation, taking a stroke trajectory with pseudospectral parameterization X_s and drawing a stroke image according to the given trajectory. The $\beta_k \|z_k\|^2$ are regularization terms on the brush height to make sure the trajectory has a tendency to lift up the brush near the end of the stroke. The weights β_k were determined by experiment.

The optimization is conditioned on the initial brush state $\mathcal{X}_{init}$. The position is initialized from the trajectory, whereas the width w, drag d, offset o, and orientation θ are simply set from the final values of the previous brush stroke.

In practice we try all polynomial orders between 3 and 8 for each stroke and pick the one with the smallest error, in order to adapt to strokes of varying length and complexity. A dense and even sampling of the polynomial is performed in V to execute a continuous stroke in simulation.

We use the Levenberg-Marquardt (LM) algorithm, a second-order trust-region method, to minimize Eq. 18. In short, LM switches between a gradient-based search and a second-order Gauss-Newton update by controlling a *damping factor* λ:

$$[\mathbf{J}^T\mathbf{J} + \lambda \mathbf{diag}(\mathbf{J}^T\mathbf{J})]\delta = \mathbf{J}^T[V(c_i) - I_i] \qquad (20)$$

The Jacobian matrix $\mathbf{J}$ is computed using numerical differentiation at each iteration. We use the the GTSAM library [36] 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 [37], [38].

F. Trajectory Initialization Estimates

We use the skeleton of a stroke to initialize the x and y coordinates, 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 [39], [40], an example result of which is shown in Fig. 6a. In our case, we use the "animation path" provided by the database 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.

Fig. 6: (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)

Given the initial trajectory, we can simulate the image formation process using both the simple virtual brush and the dynamic virtual brush, as illustrated in Fig. 6b and Fig. 6c respectively. The position of the written mark from the dynamic virtual brush is different from its given trajectory, which is ignored by most previous research.

G. Improving the Sim2Real Gap

Part of the Sim2Real gap is due to the unpredictable state of the brush after executing a stroke. After the brush writes a complicated stroke the tip may become messy. At that time, accurately predicting the state of the brush and finding a feasible control trajectory is difficult and unreliable. Writing brushes also need to periodically dip ink to be able to write.

To avoid accumulated prediction error as more strokes are written, we have the robot *dip ink* after a stroke is written, which restores the brush to a predictable state. We handcrafted a control algorithm to accomplish this: given a circular ink stone, the brush is pushed down heavily at first to make the tip flat, and we then slowly move it to the edge of the ink stone in different directions with gradually smaller extent. After that, the tip is generally restored to a predictable state, which we reflect in the initial state $\mathcal{X}_0$ of the dynamic brush model. In particular, width w, drag d, offset o, are set to zero, whereas the orientation θ is initialized to align with initial trajectory direction.

VI. RESULTS

Below we present the results of our approach, including photographs of characters that have been drawn by a Franka robot, using inverse kinematics provided by MoveIt! [41]. The characters are written using relatively slow end-effector velocity to prevent excessive jerk and vibrations.
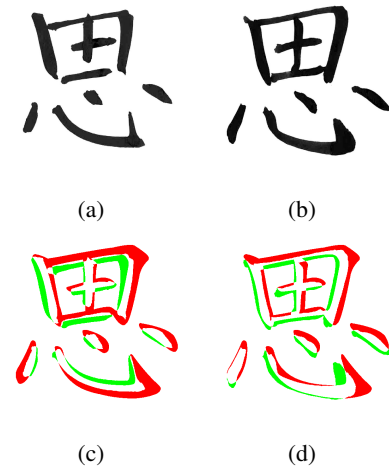

Fig. 7: Character 'si', meaning 'think'. (a) Written result following *simple* brush optimization; (b) Written result following *dynamic* brush optimization; (c) Written result from (a) overlaid with original SVG file; (d) Written result from (b) overlaid with original SVG file. The green and red pixels represent positive and negative differences respectively.

Fig. 7 shows a comparison between the written results generated by the simple virtual brush model and by the dynamic virtual brush model. The simple virtual brush model is able to generate good touching and brush outlines for some simple strokes (like dot) because of the good linear fitting model in the Sim2Real section, but not all (like the long hook). However, it can be seen that the dynamic virtual brush shows obvious advantages in terms of preserving the relative positions of strokes, since it models the deformation of the real brush. Periodically restoring the brush to its initial state helps the simple brush to behave better, but it still worse than the dynamic virtual brush model.

In Fig. 8, we show more extensive results for three different characters. Both the simulated 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.

A detailed error analysis shows that the optimization achieves almost perfect results in simulation, but that there still is a considerable Sim2Real gap. Fig. 9 shows the differences between reference images and both simulated and written characters, respectively before and after optimization. As in Figure 7, green and red pixels represent positive and negative differences. Column (c) shows that the optimization can achieve near-perfect results, where the only errors remaining stem from anti-aliasing, as well as a few small errors near the stroke endings. However, we see larger errors when these optimized results are executed on the robot (column e), even though the characters themselves look visually pleasing.
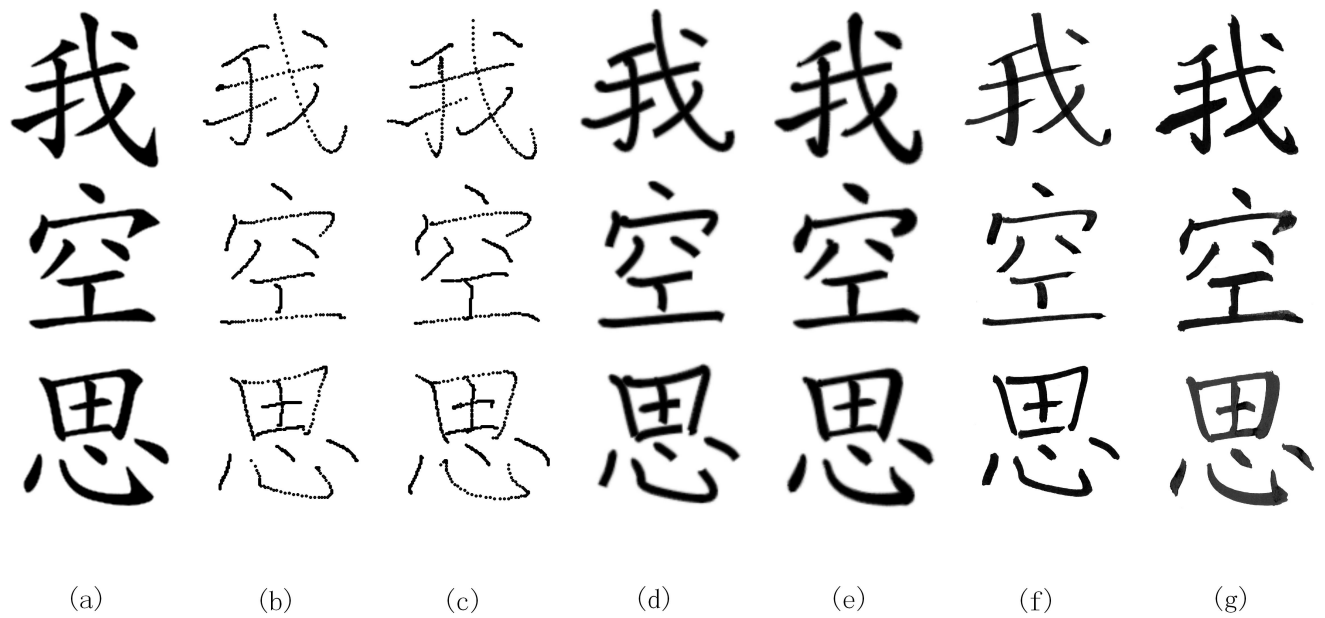
Fig. 8: The optimization of different characters: from top down, 'wo', 'kong', and 'si', meaning 'me', 'empty', and 'think'. (a) The original character pictures from the database; (b) Initial trajectory estimates; (c) The trajectory obtained from optimization; (d) Simulated image drawn by the virtual 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.
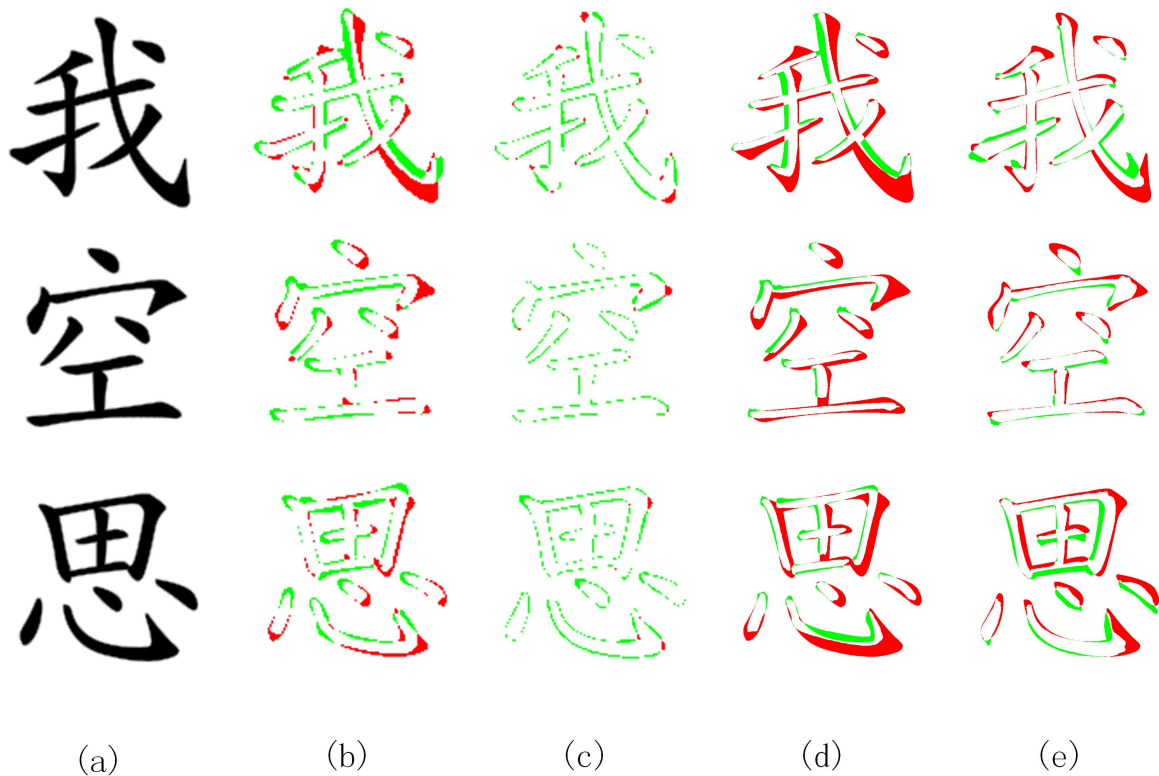

Fig. 9: Error analysis for the three characters from Figure 8. As before, green and red pixels represent positive and negative differences, respectively. (a) The original character pictures from the database; (b) Simulated image drawn by the dynamic virtual brush using the initial trajectory; (c) Simulated image drawn by the dynamic virtual brush using an *optimized* trajectory; (d) Actual written image following initial trajectories; (e) Actual written image following the *optimized* trajectories.

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 to approximate a reference image. The proposed dynamic virtual brush model 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 ever yield master-level calligraphy. Hence, in future work we plan to investigate both more predictable brush models to reduce the uncertainty, as well as the possibility of adding feedback control around optimized trajectories.

References

[1] Y. Sun, H. Qian, and Y. Xu, "Robot learns Chinese calligraphy from demonstrations," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4408–4413, IEEE, 2014.

[2] A. Kotani and S. Tellex, "Teaching robots to draw," *2019 Intl. Conf. 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," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1734–1739, IEEE, 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," in *2006 IEEE Intl. Conf. on Automation Science and Engineering*, pp. 466–471, IEEE, 2006.

[7] S. Xu, F. C. M. Lau, and Y. Pan, *A computational approach to digital Chinese painting and calligraphy*. Springer Science & Business Media, 2009.

[8] 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.

[9] 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.

[10] 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.

[11] S. Kudoh, K. Ogawara, M. Ruchanurucks, and K. Ikeuchi, "Painter robot: Manipulation of paintbrush by force and visual feedback," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) Workshop" Art and Robots*, pp. 63–68, 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," in *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pp. 185–190, IEEE, 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," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3207–3212, IEEE, 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] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *Intl. J. of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.

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

[39] 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.

[40] 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.

[41] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *ArXiv*, vol. abs/1404.3785, 2014.