The Ink Remembers the Brush: Recovering 6-DoF Brush Trajectories from Calligraphy Images with Visual Outcome Supervision

Abstract— **Learning fine-grained embodied skills is bottlenecked by scarce expert demonstrations, while task-outcome images are abundant. Turning outcomes into action supervision could enable scalable skill learning. Chinese brush calligraphy is a difficult testbed because ink traces encode rich motion cues yet the writing process is hard to model. Prior trajectory-recovery methods reduce writing to 2D centerlines and local brush widths, typically discard brush orientation, and yield representations robots cannot directly execute. To address these challenges, we introduce CalliTraj-1k, a dataset of 1,000 real 6-DoF brush trajectories, and CalliMaster, a framework that learns executable 6-DoF actions from large-scale unlabeled calligraphy images. CalliMaster distills a particle-based brush-tuft simulator into a neural contact surrogate and couples it with differentiable rendering for efficient trajectory-to-ink modeling. Progressive vision-supervised training transfers motion priors and physical constraints from limited real actions to two million unlabeled images. CalliMaster consistently outperforms existing methods in ink reconstruction, with the largest gains on out-of-domain Cursive script. Real-robot experiments confirm that predicted trajectories are visually faithful, physically plausible, and directly executable.**

I. Introduction

Learning fine motor skills for embodied intelligence has long been held back by data. For writing, painting, and sculpting, recording expert motions demands accurate sensing and skilled demonstrators, as exemplified by demonstration-based robotic calligraphy [1], [2]. Collecting images or videos of the final result requires no such motion capture. The resulting imbalance poses a central question: with very few real trajectories, how can abundant images without action annotations supervise motion and teach robots fine manipulation skills?

Brush calligraphy is a compelling test case. Its final ink trace carries rich evidence of the underlying motion [3], [4], and the task combines technical difficulty with cultural and practical value. Yet scarce demonstrations are only part of the challenge. Flexible bristle deformation, brush–paper contact, and dependence on contact history make the writing process difficult to model [5]. Capturing the trajectory-to-ink mapping both accurately and efficiently therefore calls for specialized brush models [5], [6].

Most existing trajectory-recovery methods sidestep these difficulties by simplifying the action representation. They approximate writing with 2D stroke centerlines, sometimes augmented with local widths [3], [4], [7]; supervised approaches learn these geometric representations from constructed trajectory labels [7]. This representation can reproduce plausible ink on a virtual canvas, but it does not explicitly encode 3D brush orientation or the evolving physical state of compliant contact. In real writing, brush rotation can move the tip path well away from the stroke centerline, as illustrated in Fig. 1. Nor can a robot execute a local stroke width: deployment still requires an empirical conversion from width to brush height [4]. Reconstructing an image is therefore not equivalent to recovering the physical motion that a robot can execute.

We introduce **CalliMaster** to learn executable 6-DoF brush trajectories from images of the final result. Our contributions are fourfold:

1) **Executable 6-DoF motion from visual outcomes.** CalliMaster predicts robot-executable brush poses rather than 2D centerlines and local widths.
2) **A closed loop from action to visual outcome.** Differentiable trajectory-to-ink modeling. A particle simulator supervises a neural contact surrogate coupled with an analytic renderer, enabling image-only action supervision.
3) **Progressive learning from scarce actions to abundant images.** Progressive action learning. Training scales from $\sim$1K real trajectories to 300K weakly annotated font characters and 2M unlabeled calligraphy images.
4) **CalliTraj-1k.** We introduce 1,000 motion-captured trajectories of real single-character writing, providing scarce, high-quality 6-DoF action data for brush trajectory recovery and robotic calligraphy.

II. Related Work

A. Handwriting Trajectory Recovery

Recovering writing trajectories from images has been studied for over two decades. Early methods relied on geometry: Yao *et al.* [3] thinned calligraphy images to obtain stroke centerlines and fitted B-splines to plan brush motion. With deep learning, DED-Net [8] predicted pen-tip coordinate sequences directly, while Cross-VAE [9] learned a cross-modal image–trajectory mapping. PEN-Net [10] and the Trajectory Transformer [11] further improved recovery for complex characters, long sequences, and global structure. More recent work moves toward richer structural and generative representations: InkSight [12] uses vision–language priors, G-HTR [7] uses contour-aware graphs to model structure, trajectory, and local stroke width, ordered stroke prediction [13] and dual-domain priors [14] introduce stroke hierarchies and writing-motion priors, and diffusion models [15] formulate recovery as conditional generation.

These methods primarily recover 2D pen-tip paths, sometimes augmented with local width and pen-up states, as in G-HTR [7]. Such paths do not specify the 3D brush pose

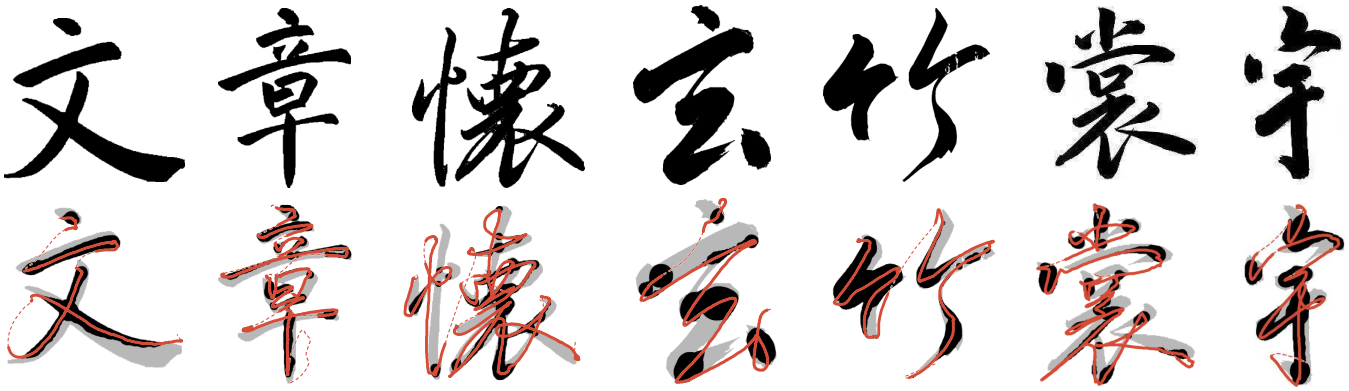

Fig. 1. Ink geometry and physical brush motion. Top: calligraphy images. Bottom: recorded brush trajectories projected onto the image plane (red), overlaid on the corresponding ink traces. The paths can deviate substantially from the ink centerlines, motivating action representations that account for brush orientation and contact.

or evolving contact state. Figure 1 illustrates the discrepancy between ink centerlines and real brush trajectories. G-HTR supports robot execution through CalliRewrite's control-conversion pipeline [4], [7]; CalliRewrite calibrates local radius to brush height [4]. Thus, a geometric reconstruction still requires additional modeling to produce executable brush motion.

B. Executable Action Generation for Robotic Calligraphy

Existing robotic calligraphy systems can be grouped by the source of their executable actions. Learning from demonstrations: Callibot [1] learns calligraphic strokes from demonstrations, Chao *et al.* [16] build and compose a stroke database from captured arm gestures, and Xie *et al.* [2] learn 3D pen-tip motion and rotation from paired images and poses. Their action supervision relies on recorded demonstrations. Geometry-to-action mappings: Yao *et al.* [3] derive brush paths from image geometry, and Yang *et al.* [17] propose a hybrid image-to-handwriting control framework. Guo *et al.* [18] plan 3D trajectories with rules for pressing height, tilt, and rotation; their formulation uses explicit geometric rules rather than learned image-to-pose inference. Reinforcement learning and tool-aware optimization: Wu *et al.* [19] learn basic-stroke policies from stroke images and rewards. CalliRewrite [4] extracts coarse paths without action labels, refines them through reinforcement learning with tool-contact models, and converts local radii to heights for 3D position control. Our focus is image-conditioned 6-DoF pose prediction with a bristle-contact surrogate.

C. Physical Brush Models and Differentiable Action Learning

Brush modeling dates back to Virtual Brush [6] and later virtual-brush models such as Droplet [20]. These methods synthesize ink by representing brush geometry and deformation, but they mainly target calligraphy synthesis and rendering in computer graphics rather than robot action learning. A closely related robotics approach is Wang *et al.* [5], whose dynamic virtual-brush model maps robot trajectories to rendered ink and uses pseudospectral optimal control to optimize parameterized execution trajectories for individual strokes. More broadly, ChainQueen [21] and DiffPD [22] enable gradient-based optimization through physical simulation, while graph-network simulators [23] learn particle dynamics from simulation data.

These approaches motivate a scalable connection between image-based trajectory recovery and brush-contact modeling. CalliMaster uses an efficient, differentiable trajectory-to-ink mapping to turn large collections of unlabeled calligraphy images into supervision for 6-DoF action learning.

III. Method

A. Problem Formulation and Overview

CalliMaster has three components (Fig. 2). A particle-based bristle-bundle model first maps six-degree-of-freedom (6-DoF) motion to brush–paper contact geometry. We then distill this costly physical process into a differentiable neural contact simulator and combine it with an analytic ink renderer. Finally, progressive training moves from individual strokes to whole characters, and from real-action supervision to image-only supervision, transferring motion priors from a small set of real trajectories to large-scale calligraphy images without action labels.

B. Training Data

CalliMaster uses three data sources with decreasing supervision strength and increasing scale.

1) CalliTraj-1k: real 6-DoF trajectories: This dataset contains approximately 10^3 character trajectories from 12 calligraphers, mainly in Regular and Running scripts, recorded as positions and quaternions in a common paper coordinate frame. Stroke segmentation yields approximately 7×10^3 single-stroke samples. We hold out 100 image–trajectory pairs as *Collected*, with disjoint calligraphers between training and evaluation to prevent style leakage.

2) Font characters with centerline annotations: Approximately 3×10^5 characters rendered from computer calligraphy fonts provide per-stroke ink masks, two-dimensional centerlines, and stroke order, without 6-DoF actions.

3) Unlabeled real calligraphy images: Approximately 2×10^6 real calligraphy images span multiple scripts and contain no action annotations.

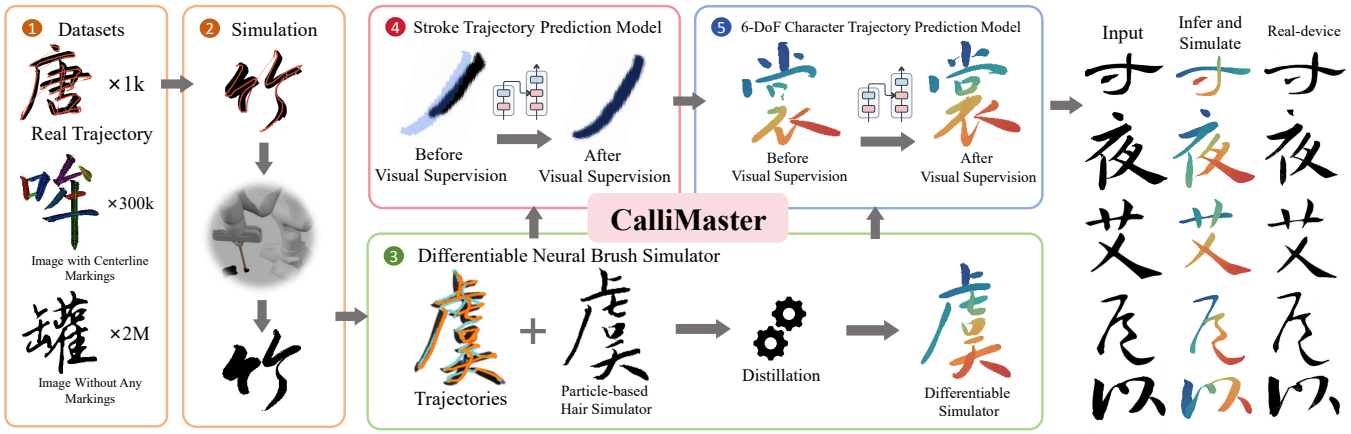

Fig. 2. CalliMaster overview. Real trajectories, font characters with centerline annotations, and unlabeled calligraphy images support progressive action learning. A particle-based bristle simulator supplies targets for a differentiable neural contact simulator, which is coupled with ink rendering to supervise single-stroke and whole-character trajectory predictors through their visual outcomes. The learned 6-DoF trajectories can drive robot writing. Numbered blocks identify components of the pipeline; training stages are detailed in Sec. III-F.

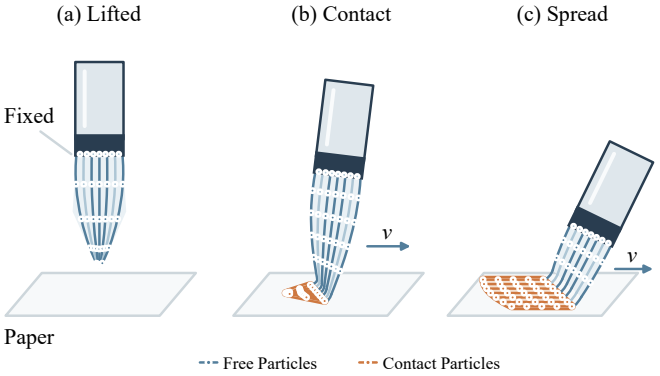

Fig. 3. Macroscopic bristle-bundle behavior. (a) Lifted bristles form a tapered cone. (b) Light pressing preserves a compact tip in contact with the paper. (c) Stronger pressing spreads the bristles and enlarges the contact region. Fixed roots follow the handle pose; blue and orange markers denote free and contacting particles, respectively, and v indicates brush motion.

C. Particle-Based Bristle Simulation

Brush–paper contact depends on coupled bristle deformation, contact, and surface-tension effects (Fig. 3). Full fluid–structure simulation is prohibitively expensive for generating 10^5-10^6 trajectories. We therefore model only the macroscopic contact geometry relevant to ink formation.

1) Bundle representation and constraint reduction: The brush contains $N_h = 500$ bristles, each discretized into $N_p = 50$ particles, for approximately 2.5×10^4 particles in total. Bristle roots are sampled according to a radial distribution over the root cross-section. The five particles nearest each bristle root are kinematic, rigidly attached to the handle, and positioned directly by the handle position p_t and orientation R_t. The remaining particles are solved under three classes of constraints:

- *Root attachment* imposes kinematic boundary conditions, with bundle motion driven entirely by the handle pose.
- *Bundle alignment* determines the overall configuration under the handle orientation, including conical tapering and spreading under compression.
- *Unilateral paper contact and friction* determine the actual contact set and tangential behavior when the bundle interacts with the rigid paper.

2) Orientation-dependent bundle alignment: Let p_s, p_e be axial reference points on the fixed segment of a bristle, and let r_h be its initial radial offset at the root. The target position of free particle k is

$$p_k^{\text{tgt}} = p_s + \kappa_k(p_e - p_s) + \beta_k R_t r_h, \tag{1}$$

Here, κ_k parameterizes the axial position and β_k tapers the radial offset from root to tip. Rotation by R_t couples handle orientation to contact geometry; because targets depend only on the handle pose and rest geometry, all particles can be projected independently.

3) Contact folding and friction: Paper contact combines axial folding, radial spreading, and friction. Let the paper plane be $z = z_0$, its normal $n = (0,0,1)^{\mathsf{T}}$, and the tangential projector $P_{\parallel} = \mathrm{I}_3 - nn^{\mathsf{T}}$. The axial reference position of free particle k is $a_k = p_s + \kappa_k(p_e - p_s)$. Let c_k be the intersection of the bristle axis with the paper. For a penetrating reference point a_k, we preserve the penetration length $\ell_k = \|a_k - c_k\|$ and fold it into the paper plane,

$$a_k^{\text{fold}} = c_k + \ell_k d_k, \quad n^{\top} d_k = 0, \ \|d_k\| = 1.$$

This converts axial penetration into an in-plane extension of equal length. When the reference axis is nearly parallel to the paper, we use orthogonal projection to avoid numerical degeneracy.

Radial extent increases monotonically with the near-paper particle fraction χ_t, allowing the bundle to spread under compression while remaining compact under weak contact.

The folding direction depends on both handle orientation and motion history. Define the axial projection direction as $d_k^{\text{axis}} = P_{\parallel}(a_k - c_k)/\|P_{\parallel}(a_k - c_k)\|$, and let $\hat{v}_t$ denote the unit direction of local brush motion. We measure contact by $\chi_t = N^{-1}\sum_i \mathbf{1}[x_i^z \le z_0 + \varepsilon]$ and set $w_t = \mathrm{clip}(\chi_t/0.7, 0, 1)$. The folding direction is

$$d_k = \frac{w_t d_k^{\text{axis}} - (1 - w_t)\hat{v}_t}{\|w_t d_k^{\text{axis}} - (1 - w_t)\hat{v}_t\|}. \tag{2}$$

Under weak contact, folding primarily opposes brush motion, capturing bristle drag. As contact strengthens, the projected handle axis increasingly dominates the folding direction.

4) Single-pass direct projection: Within each frame, a free particle is updated as

$$x_{k,t} = \Pi_{\mathrm{cf}}\Big(\hat{x}_{k,t} + s_k\big(p_k^{\mathrm{tgt}} - \hat{x}_{k,t}\big)\Big), \tag{3}$$

where $\hat{x}_{k,t}$ is the inertial prediction, $s_k \in (0,1]$ is a compliance-dependent projection gain, and Π_{cf} applies contact folding and friction. Each frame follows a fixed sequence: explicit integration, root placement, alignment projection, and contact handling. These operations are fused into a small number of GPU kernels and require only a single solve pass.

D. Differentiable Forward Model

To preserve physical structure while keeping training efficient, we factor the forward process as trajectory $\rightarrow$ contact geometry $\rightarrow$ ink image, implemented by a neural contact simulator $\mathscr{S}_\psi$ and an analytic differentiable renderer $\mathscr{R}$.

1) Fourier representation of contact cross-sections: We extract a closed contour from the simulated two-dimensional contact particles and resample it along arc length into 128 points. After normalization by centroid μ_t and scale σ_t, we apply a discrete Fourier transform (DFT), remove the DC component, and retain 16 positive and 16 negative frequencies to obtain $F_t \in \mathbb{R}^{64}$. Each contact state is $C_t = (c_t, \mu_t, \sigma_t, F_t)$, encoding contact probability, center, scale, and shape. This compact representation supports differentiable contour reconstruction through an inverse fast Fourier transform (IFFT).

2) Neural contact simulator: The neural simulator $\mathscr{S}_\psi$ takes a complete 6-DoF trajectory and predicts contact states. Given $s_t = [x_t^n, y_t^n, z_t^n, q_t, v_t^n, \|v_t^n\|]$, where q_t is a unit quaternion, v_t is the position difference between adjacent frames, and superscript n denotes normalization. Six Mamba [24] residual blocks (hidden size 192) predict C_t frame by frame.

$$\begin{aligned}\mathscr{L}_{\mathscr{S}} = \ & \mathrm{BCE}(\hat{c}_t, \tilde{c}_t) \\ & + \tilde{c}_t\big[\mathrm{S}(\hat{F}_t, \tilde{F}_t) + \mathrm{S}(\hat{\mu}_t, \tilde{\mu}_t) \\ & \qquad + \mathrm{S}(\hat{\sigma}_t, \tilde{\sigma}_t)\big],\end{aligned} \tag{4}$$

where hats denote predictions, tildes denote physical-simulator targets, and S is the Smooth L1 loss. Geometry losses apply only to contact frames. After training, $\mathscr{S}_\psi$ is frozen and serves as the differentiable physics surrogate for subsequent learning.

3) Differentiable ink rendering: We soft-rasterize each contour as $\alpha_t(\pi) = \sigma(\mathrm{SDF}_t(\pi)/\varepsilon)$ and gate it by $g_t = \sigma((\hat{c}_t - 0.5)/\tau_c)$. , the final ink trace accumulates through saturating coverage:

$$\hat{I}(\pi) = 1 - \prod_{t=1}^{T}\big(1 - g_t\,\alpha_t(\pi)\big). \tag{5}$$

Here, $\hat{I}$ denotes ink coverage, with one indicating ink and zero indicating background. Image supervision is

$$\begin{aligned}\mathscr{L}_{\mathrm{img}} = \ & \mathscr{L}_{\mathrm{BCE}} + \mathscr{L}_{\mathrm{Dice}} + 0.25\,\mathscr{L}_1 \\ & + \lambda_{\mathrm{bd}}\,\mathscr{L}_{\mathrm{bd}},\end{aligned} \tag{6}$$

where $\mathscr{L}_{\mathrm{bd}}$ is a boundary-distance-weighted loss that emphasizes stroke contours and width changes.

E. Trajectory Predictors

1) Single-stroke predictor: Limited real-stroke supervision motivates a low-frequency smoothness bias. After resolving quaternion sign discontinuities, we uniformly sample $M = 48$ position–quaternion control poses in time and normalize each channel to obtain $C^{\mathrm{n}} \in \mathbb{R}^{M\times 7}$. We retain the lowest $K_c = 32$ components of the DCT-II basis Φ:

$$\alpha = (\Phi^{\mathsf{T}} C^{\mathrm{n}})_{0:K_c,:} \in \mathbb{R}^{K_c\times 7}. \tag{7}$$

A residual convolutional encoder and a Transformer [25] decoder with K_c learnable frequency queries jointly predict $\hat{\alpha} = g_\phi(I_{\mathrm{stroke}})$ from the stroke ink image. Inverse DCT, quaternion renormalization, and cumulative cubic $SE(3)$ B-spline interpolation [26] recover a continuous trajectory.

2) Whole-character predictor: With large-scale pseudo-action supervision, the whole-character model predicts trajectory points directly to retain local calligraphic detail without low-frequency truncation.

The input $X = [I, S, D]$ comprises the ink image, skeleton, and interior Euclidean distance field, encoding boundaries, topology, and local stroke width. ConvNeXt V2 [27] with a feature pyramid network (FPN) [28] builds a low-resolution structural memory for global stroke relationships and a high-resolution detail memory for local shape. Following masked autoencoding [29], we pretrain the visual encoder on real calligraphy training images through masked reconstruction of $[I, S, D]$.

A decoder with $T = 384$ temporal queries jointly predicts $\hat{u}_t = [\hat{x}_t, \hat{y}_t, \hat{z}_t, \hat{\rho}_t]$, where $\hat{\rho}_t \in \mathbb{R}^6$ is converted to a valid orientation by Gram–Schmidt orthogonalization [30]. Auxiliary heads predict contact states and writing phases.

F. Progressive Action Learning

Training progresses along two axes: from individual strokes to whole characters, and from real-action supervision through weak geometric supervision to image-only supervision. Limited real 6-DoF actions first establish a reliable prior in motion and physical space. The differentiable forward model then transfers that prior to large-scale data without action labels.

1) Stage I: physical teacher and surrogate: We drive the bristle simulator with real CalliTraj-1k trajectories, obtain contact cross-sections at each frame, and train $\mathscr{S}_\psi$. The trained surrogate is then frozen.

2) Stage II: real-action supervision for individual strokes: We train g_ϕ on approximately 7×10^3 real strokes. The loss supervises DCT coefficients, control poses, and first-order differences:

$$\begin{aligned}\mathscr{L}_{\mathrm{act}} = \ & \|\hat{\alpha} - \alpha\|_{\mathrm{lf}}^2 + 0.2\,\mathrm{S}(\hat{P}, P) \\ & + 0.2\,\mathrm{S}(\hat{P}_{xyz}, P_{xyz}) \\ & + 0.25\,\mathrm{S}(\Delta\hat{P}, \Delta P).\end{aligned} \tag{8}$$

Here, P denotes the control-pose sequence, and P_{xyz} its positional channels. Higher weights on low-frequency coefficients prioritize the overall stroke direction and dominant orientation changes.

3) Stage III: weak physical supervision for individual strokes: We continue training g_ϕ on the font data, using the stroke masks and centerline annotations for supervision of individual strokes. Predicted trajectories pass through $\mathscr{S}_\psi$ and $\mathscr{R}$ for ink reconstruction under $\mathscr{L}_{\text{img}}$. We also match predicted contact centers to centerline points resampled by arc length:

$$\mathscr{L}_{\text{weak}} = \mathscr{L}_{\text{img}} + \lambda_{\text{mid}} \mathscr{L}_{\text{midline}}. \tag{9}$$

A stroke centerline is only the two-dimensional geometric centerline of its ink region; it cannot encode the lift height or orientation changes of real writing. Nevertheless, a correct trajectory should reproduce the target stroke after simulation and rendering, so its ink centers should remain close to the centerline. Centerline matching thus supplies weak but directionally informative geometric supervision for action prediction.

4) Stage IV: whole-character pseudo-action supervision: Using g_ϕ as a pseudo-action generator, we predict each stroke of a font character in writing order, insert airborne transitions, concatenate the strokes into a full trajectory, and resample it to $T = 384$ points to obtain τ^{pseudo}. We initialize the whole-character model f_θ with the masked-autoencoder-pretrained visual encoder and optimize

$$\begin{aligned}\mathscr{L}_{\text{sup}} = {} & \mathrm{S}(\hat{u}, u) + \mathrm{S}(\hat{u}_{xyz}, u_{xyz}) + \mathrm{S}(\hat{u}_\rho, u_\rho) \\ & + 0.5\,\mathrm{S}(\Delta\hat{u}, \Delta u) \\ & + 0.25\,\mathrm{S}(\Delta^2\hat{u}, \Delta^2 u) \\ & + 0.1\,\mathscr{L}_{\text{contact}} + 0.1\,\mathscr{L}_{\text{seg}}.\end{aligned} \tag{10}$$

5) Stage V: physics-consistent post-training on real images: We optimize f_θ on the same real calligraphy training images used for encoder pretraining. Predicted trajectories reconstruct the ink through frozen $\mathscr{S}_\psi$ and renderer $\mathscr{R}$, with objective

$$\begin{aligned}\mathscr{L}_{\text{real}} = {} & \mathscr{L}_{\text{img}} + \lambda_z \Omega_z + \lambda_l \Omega_{\text{len}} \\ & + \lambda_r \Omega_{\text{revisit}} + \lambda_b \Omega_{\text{break}} + \lambda_v \Omega_{\text{vel}}.\end{aligned} \tag{11}$$

The height prior Ω_z keeps the brush tip within a plausible working range. The length prior Ω_{len} matches contact-path length to character geometry. The revisit prior Ω_{revisit} discourages temporally distant contact points from repeatedly visiting the same spatial region. The switching prior Ω_{break} suppresses implausible high-frequency alternation between contact and lift-off. Finally, Ω_{vel} uses the Stage IV model as a soft teacher to constrain motion rhythm. Together, these terms exclude degenerate trajectories that attain low image loss while violating real writing dynamics.

IV. Experiments

We evaluate forward-model fidelity, ink reconstruction, 3D trajectory recovery, and robot execution, then ablate 6-DoF prediction, geometric inputs, and visual post-training.

A. Experimental Setup

Test sets. We evaluate on Cursive, Regular, Clerical, and Running scripts, and the *Collected* split of CalliTraj-1k (Sec. III-B). Collected is excluded from all training. All test images are excluded from the 2M-image corpus used for visual-encoder pretraining and post-training.

Image metrics. We report clDice [31] for stroke-skeleton topology, PSNR for pixel fidelity, and encoder similarity (Enc) in Table II. Forward-model and ablation experiments also report mIoU for ink-region overlap; real-robot evaluation reports clDice, PSNR, and Enc. For simulated ink, Enc uses features from a Vision Transformer (ViT) [32] fine-tuned on approximately 5M raw, unannotated calligraphy images.

B. Simulator Fidelity

Table I compares centerline approximation, the particle-based bristle simulator, and the neural contact simulator. The particle model improves mIoU and clDice over centerline approximation by 6.39 and 13.97 percentage points, respectively, and raises Enc from 0.8360 to 0.9425. Figure 4 shows fewer stroke breaks and missing regions, with better continuity and width variation. Contact modeling thus directly improves ink-structure reconstruction.

The neural simulator closely matches its physical teacher: mIoU and clDice decrease by only 0.0054 and 0.0095, PSNR differs by 0.046 dB, and Enc improves slightly. Their global contours agree, with differences concentrated at local boundaries. This fidelity supports using the neural simulator for differentiable forward computation during visual supervision.

C. Trajectory Recovery Across Scripts

Baselines. We compare with CalliRewrite [4], G-HTR [7], and a Baseline. Baseline uses the CalliMaster architecture and the same font data, but predicts 2D trajectories, local

TABLE I

Forward-model ink reconstruction. The neural surrogate is paired with the differentiable ink renderer. Higher is better; best values are bold.

Model	mIoU	clDice	PSNR (dB)	Enc
Centerline	0.6563	0.6800	8.2736	0.8360
Particle simulator	**0.7203**	**0.8196**	**8.9280**	0.9425
Neural surrogate	0.7148	0.8101	8.8817	**0.9439**

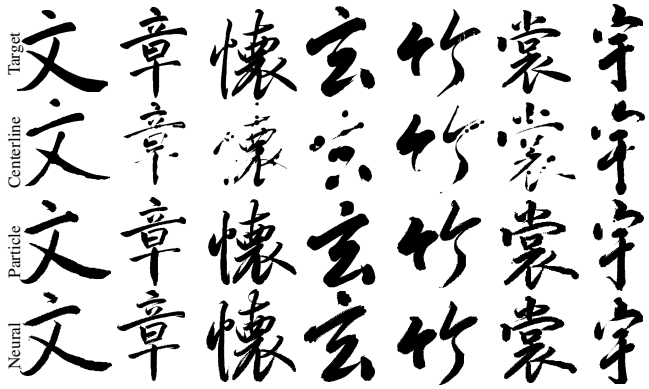

Fig. 4. Forward-model comparison. Rows show the reference ink, centerline approximation, particle simulator, and neural surrogate. Both simulators reduce broken strokes and local shape errors.

TABLE II

INK RECONSTRUCTION ACROSS SCRIPTS, MEASURED BY CLDICE, PSNR (DB), AND ENC. /M: CENTERLINE-AND-WIDTH RENDERING; /S: PARTICLE-BASED BRUSH SIMULATION. CALLIMASTER IS EVALUATED ONLY UNDER /S. HIGHER IS BETTER; BEST VALUES ARE BOLD.

	Cursive			Regular			Clerical			Collected			Running		
Method	clDice	PSNR	Enc	clDice	PSNR	Enc	clDice	PSNR	Enc	clDice	PSNR	Enc	clDice	PSNR	Enc
Baseline/m	0.5922	6.4931	0.8049	0.7992	7.3280	0.9036	0.9033	9.2951	0.9187	0.8784	10.2213	0.9370	0.7140	7.0262	0.8340
Baseline/s	0.5270	6.1662	0.7733	0.7373	7.4388	0.8532	0.7959	7.5765	0.8330	0.8513	9.3946	0.9235	0.6772	6.9120	0.7912
CalliRewrite/m	0.9407	10.6901	0.9386	0.9233	8.9324	0.9293	0.9159	10.3572	0.9672	0.9074	11.8973	0.9734	0.8683	9.0778	0.9043
CalliRewrite/s	0.8061	8.0741	0.9177	0.8134	7.2816	0.9044	0.8579	8.3547	0.9074	0.8494	10.1831	0.9521	0.7468	7.5874	0.8813
G-HTR/m	0.8128	8.2739	0.8931	0.8155	8.9016	0.9543	0.6086	5.6870	0.7554	0.5422	7.9526	0.9087	0.6444	7.4493	0.8789
G-HTR/s	0.8051	7.8788	0.8651	0.7814	8.2639	0.9413	0.5323	5.2361	0.7529	0.5043	7.2078	0.9064	0.6028	6.6004	0.8602
CalliMaster/s	**0.9685**	**12.2577**	**0.9690**	**0.9643**	**10.9720**	**0.9789**	**0.9308**	**10.9223**	**0.9760**	**0.9230**	**11.9577**	**0.9788**	**0.9507**	**11.1972**	**0.9627**

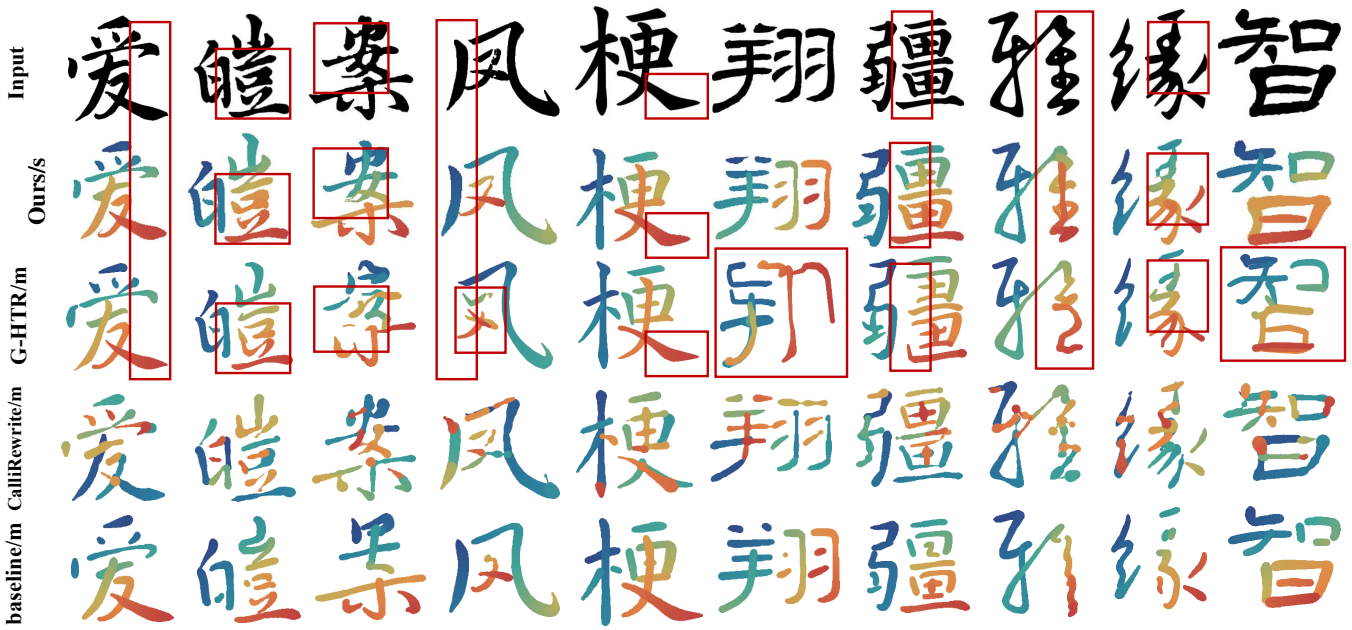

Fig. 5. Comparison with centerline rendering. Rows show the input, CalliMaster/s, G-HTR/m, CalliRewrite/m, and Baseline/m. Color encodes writing progress; red boxes highlight local shape differences. CalliMaster uses physical simulation throughout.

TABLE III

TRAJECTORY AND ORIENTATION ACCURACY ON THE COLLECTED TEST SET. BEST VALUES ARE BOLD; — DENOTES UNAVAILABLE RESULTS.

Method	Cosine ↑	RMSE ↓ (mm)	Fréchet ↓ (mm)	MAE ↓ (°)
Baseline	0.3264	66.86	124.21	—
CalliRewrite	0.4541	55.75	77.45	—
G-HTR	0.2595	59.85	138.79	—
CalliMaster	**0.8741**	**22.68**	**54.38**	**14.95**

widths, and pen-down states. We evaluate centerline methods under two protocols: `/m` directly draws centerlines with local widths; `/s` converts widths to tip heights using CalliRewrite's calibration and renders the resulting trajectories with the same particle-based bristle simulator. CalliMaster directly predicts 6-DoF trajectories and is evaluated under `/s`.

Quantitative results. CalliMaster/s leads clDice, PSNR, and Enc on all five test sets, outperforming even directly rendered centerline baselines (Table II). Under shared physical simulation, its macro-average clDice reaches 0.9475, whereas CalliRewrite drops from 0.9111 with direct drawing to 0.8147. This degradation exposes the limitation of centerline recovery: matching ink geometry does not ensure consistency with brush contact.

Qualitative results. Figure 5 highlights turns, connected strokes, and dense components. Centerline methods produce omissions, spurious connections, or width mismatches. CalliMaster better preserves long oblique strokes, closely spaced horizontal strokes, and component layout. Progress colors expose trajectory organization, although ink similarity alone does not uniquely identify the original motion or stroke order.

D. 3D Trajectory Fidelity

On the Collected test set, we compare predicted (x, y, z) paths and brush orientations with recorded ground truth. For centerline methods, z comes from the same width-to-height calibration used above. Predicted and reference paths are resampled at equal arc-length intervals and translated to align their xy centers. We report XYZ cosine similarity after

TABLE IV

REAL-ROBOT INK RECONSTRUCTION ACROSS SCRIPTS, MEASURED BY CLDICE, PSNR (DB), AND ENC. HIGHER IS BETTER; BEST VALUES ARE BOLD.

	Cursive			Regular			Clerical			Collected			Running		
Method	clDice	PSNR	Enc	clDice	PSNR	Enc	clDice	PSNR	Enc	clDice	PSNR	Enc	clDice	PSNR	Enc
Baseline	0.6498	9.7702	0.8889	0.7309	10.4049	0.9239	0.7118	10.2136	0.9184	0.7028	9.7610	0.8660	0.6766	8.9734	0.9147
CalliRewrite	0.7428	10.5282	0.9167	0.7815	10.4565	0.9208	0.8240	11.2291	0.9433	0.6986	9.7793	0.9079	0.7098	8.8826	0.8925
G-HTR	0.6015	9.6514	0.9118	0.7066	9.9788	0.9309	0.7203	9.7423	0.9368	0.5882	9.5142	0.9215	0.6280	8.4182	0.9176
CalliMaster	**0.8070**	**10.7128**	**0.9436**	**0.8592**	**11.4333**	**0.9600**	**0.8486**	**11.6199**	**0.9506**	**0.7807**	**10.1984**	**0.9372**	**0.8191**	**10.0242**	**0.9401**

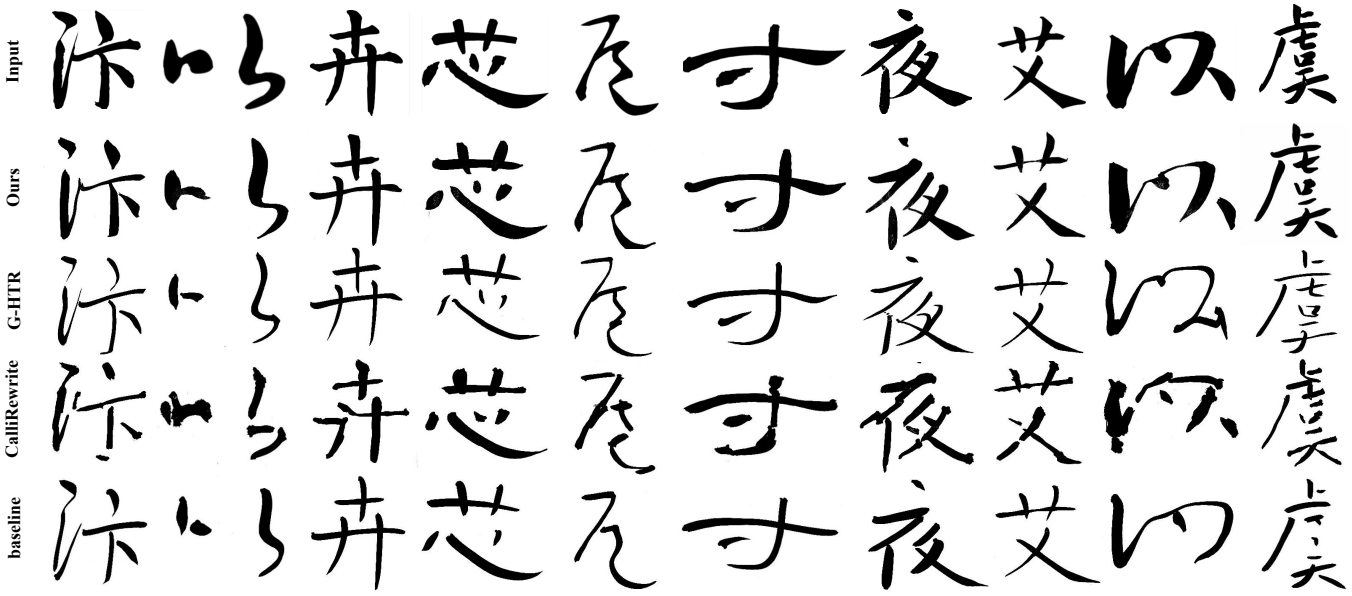

Fig. 6. Real-robot writing. Rows show the target image, CalliMaster, G-HTR, CalliRewrite, and Baseline. CalliMaster better preserves stroke widths, endings, and component layout.

TABLE V

ABLATION OF CALLIMASTER. HIGHER IS BETTER; BEST VALUES ARE BOLD.

Variant	mIoU	clDice	PSNR (dB)	Enc
w/o 6-DoF	0.6008	0.7807	6.7506	0.8388
w/o mi	0.6377	0.8213	7.2444	0.7992
w/o pt	0.6077	0.7113	7.2967	0.8731
full	**0.8167**	**0.9568**	**12.0826**	**0.9728**

Fig. 7. Qualitative ablations. The full model better preserves dense components and lower-stroke geometry.

this preprocessing, and RMSE and Fréchet distance after additional rigid alignment. We also report mean angular error (MAE) for brush orientation. This metric is unavailable for the three baselines, which do not predict orientation.

CalliMaster achieves a cosine similarity of 0.8741, versus 0.4541 for CalliRewrite, the strongest baseline on all three position metrics (Table III). These gains show more faithful recovery of recorded 3D path geometry.

E. Real-Robot Writing

We execute all four methods on five test sets (Table IV). Centerline methods use width-to-height calibration; CalliMaster executes its predicted 6-DoF poses.

CalliMaster leads all three metrics on the five evaluated subsets. Macro-average clDice reaches 0.8229 versus 0.7514 for CalliRewrite (+7.16 percentage points), with the largest gain over the strongest baseline in Running (+10.93 points). Macro-average PSNR also improves by 0.62 dB over CalliRewrite. Together with Fig. 6, these results support faithful ink reproduction during robot execution.

F. Ablation Study

Table V tests 6-DoF prediction, geometric inputs, and visual post-training. `w/o 6-DoF` predicts ink *xy* centerlines and local widths instead of 6-DoF trajectories; `w/o mi` removes the skeleton and interior distance field; `w/o pt` removes Stage V post-training (Sec. III-F); `full` retains all three.

The full model leads all four metrics, reaching 0.8167 mIoU. Replacing 6-DoF trajectories with centerlines and widths reduces mIoU by 21.58 percentage points and PSNR by 5.33 dB. Removing geometric inputs or post-training lowers mIoU to 0.6377 and 0.6077, respectively. Figure 7

shows that the full model better preserves dense components and lower-stroke geometry.

V. Conclusion

CalliMaster recovers executable 6-DoF brush trajectories from calligraphy images by combining a particle-based physical teacher, a differentiable neural contact surrogate, and progressive visual supervision. Training transfers motion priors from scarce real trajectories to font characters and unlabeled images. Across four scripts and the Collected set, CalliMaster improves ink reconstruction under shared simulation, better matches recorded 3D trajectories, and yields faithful real-robot writing. Ablations validate 6-DoF prediction, geometric inputs, and physics-consistent post-training.

References

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

[2] F. Xie, P. Le Meur, and C. Fernando, "End-to-end manipulator calligraphy planning via variational imitation learning," *arXiv preprint arXiv:2304.02801*, 2023.

[3] F. Yao, G. Shao, and J. Yi, "Extracting the trajectory of writing brush in Chinese character calligraphy," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 6, pp. 631–644, 2004.

[4] Y. Luo, Z. Wu, and Z. Lian, "CalliRewrite: Recovering handwriting behaviors from calligraphy images without supervision," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2024, pp. 8671–8678.

[5] S. Wang, J. Chen, X. Deng, S. Hutchinson, and F. Dellaert, "Robot calligraphy using pseudospectral optimal control in conjunction with a novel dynamic brush model," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2020, pp. 6696–6703.

[6] H. T. F. Wong and H. H. S. Ip, "Virtual brush: A model-based synthesis of Chinese calligraphy," *Computers & Graphics*, vol. 24, no. 1, pp. 99–113, 2000.

[7] Y. Qin, G. Dai, Y. Zhang, Y. Han, Q. He, and S. Huang, "Towards human-like robot handwriting via contour-aware generation," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2026, pp. 31 597–31 607.

[8] A. K. Bhunia, A. Bhowmick, A. K. Bhunia, A. Konwer, P. Banerjee, P. P. Roy, and U. Pal, "Handwriting trajectory recovery using end-to-end deep encoder-decoder network," in *Proc. Int. Conf. Pattern Recognition (ICPR)*. IEEE, 2018, pp. 3639–3644.

[9] T. Sumi, B. K. Iwana, H. Hayashi, and S. Uchida, "Modality conversion of handwritten patterns by cross variational autoencoders," in *Proc. Int. Conf. Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 407–412.

[10] Z. Chen, D. Yang, J. Liang, X. Liu, Y. Wang, Z. Peng, and S. Huang, "Complex handwriting trajectory recovery: Evaluation metrics and algorithm," in *Proc. Asian Conf. Computer Vision (ACCV)*, Dec. 2022, pp. 1060–1076.

[11] J. Lin, Z. Chen, L. Liang, W. Peng, and S. Huang, "Handwriting trajectory recovery via trajectory transformer with global radical context-aware module," in *Pattern Recognition: ICPR 2024*, ser. Lecture Notes in Computer Science, vol. 15320. Springer, 2025, pp. 182–195.

[12] B. Mitrevski, A. Rak, J. Schnitzler, C. Li, A. Maksai, J. Berent, and C. C. Musat, "InkSight: Offline-to-online handwriting conversion by teaching vision-language models to read and write," *Transactions on Machine Learning Research*, 2025.

[13] E.-G. Wang, Y.-M. Zhang, F. Yin, and C.-L. Liu, "Handwriting trajectory recovery via autoregressive ordered stroke instance prediction," *arXiv preprint arXiv:2609.02251*, 2026.

[14] L. Wang, M. Y. Sharum, R. B. Yaakob, K. A. Kasmiran, Y. Liu, and C. Wang, "Hierarchical offline-to-online Chinese handwriting trajectory reconstruction via dual-domain priors," *Journal of King Saud University Computer and Information Sciences*, vol. 38, no. 7, 2026, art. no. 722.

[15] H. Nagamatsu, S. Toyota, and S. Uchida, "Handwriting trajectory recovery with diffusion models," in *Document Analysis and Recognition: ICDAR 2026*, ser. Lecture Notes in Computer Science, vol. 16972. Springer, 2027, pp. 197–214.

[16] F. Chao, Y. Huang, X. Zhang, C. Shang, L. Yang, C. Zhou, H. Hu, and C.-M. Lin, "A robot calligraphy system: From simple to complex writing by human gestures," *Engineering Applications of Artificial Intelligence*, vol. 59, pp. 1–14, 2017.

[17] Y. Yang, W. Chen, L. Zhou, B. Zheng, W. Xiao, Y. Huang, and Z. Sun, "A hybrid control framework teaching robot to write Chinese characters: From image to handwriting," in *Proc. IEEE Int. Conf. Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 1161–1166.

[18] D. Guo, W. Yang, and W. Fang, "Three-dimensional trajectory planning for robotic Chinese calligraphy: Conforming to brush stroke dynamics," *International Journal of Computational Intelligence Systems*, vol. 19, no. 1, 2026, art. no. 124.

[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," in *Proc. IEEE Int. Conf. Robotics and Biomimetics (ROBIO)*, 2018, pp. 507–512.

[20] X.-F. Mi, M. Tang, and J.-X. Dong, "Droplet: A virtual brush model to simulate Chinese calligraphy and painting," *Journal of Computer Science and Technology*, vol. 19, no. 3, pp. 393–404, 2004.

[21] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "ChainQueen: A real-time differentiable physical simulator for soft robotics," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 6265–6271.

[22] T. Du, K. Wu, P. Ma, S. Wah, A. Spielberg, D. Rus, and W. Matusik, "DiffPD: Differentiable projective dynamics," *ACM Transactions on Graphics*, vol. 41, no. 2, pp. 13:1–13:21, 2022.

[23] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks," in *Proc. 37th Int. Conf. Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 119, 2020, pp. 8459–8468.

[24] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *Proc. Conf. Language Modeling (COLM)*, 2024.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[26] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras," in *Proc. British Machine Vision Conf. (BMVC)*. BMVA Press, 2013, pp. 93.1–93.12.

[27] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie, "ConvNeXt V2: Co-designing and scaling ConvNets with masked autoencoders," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 16 133–16 142.

[28] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2117–2125.

[29] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 16 000–16 009.

[30] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5745–5753.

[31] S. Shit, J. C. Paetzold, A. Sekuboyina, I. Ezhov, A. Unger, A. Zhylka, J. P. W. Pluim, U. Bauer, and B. H. Menze, "clDice – a novel topology-preserving loss function for tubular structure segmentation," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 560–16 569.

[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2021.