IMPASTO: Integrating Model-Based Planning with Learned Dynamics Models for Robotic Oil Painting Reproduction

Appendix

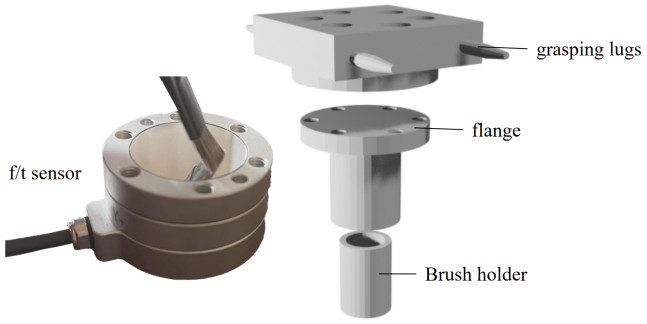

Fig. 1: The f/t sensor and the mounter.

Appendix A. Robotic Hardware System

We use a 7-DoF Franka Emika Panda equipped with a 6-axis force/torque sensor (Kunwei Tech Inc.) mounted at the wrist, as shown in Fig. 1. The mount is designed and 3D-printed, with a grasping lugs for the Franka's parallel grippers, a flange for the f/t sensor, and a brush holder for the oil painting brush. To reduce common-mode noise and mains interference, we ground the acquisition chain by connecting a grounding wire to the data-acquisition module's adapter. Sensor readouts are first bias-corrected (offset removal) at the beginning of each primitive skill and then smoothed with an exponential moving average (EMA):

$$\tilde{x}_t = x_t - \hat{b}, \qquad y_t = \alpha\,\tilde{x}_t + (1-\alpha)\,y_{t-1}, \tag{1}$$

where x_t denotes the raw force/torque reading at time step t; $\hat{b}$ is the per-primitive bias (static offset) estimated before each primitive skill from a short stationary window collected right before the primitive starts; $\alpha \in (0, 1]$ is the EMA coefficient.

The RGB observations of the canvas are captured at 1920×1080 by a single Intel RealSense D435 camera mounted above the canvas.

Dynamics model training uses Distributed Data Parallel (DDP) on NVIDIA GPUs, including RTX A5000, RTX 3090, and A40. Deployment uses NVIDIA RTX 4090 GPUs.

Appendix B. Learning the Dynamics Models

The detailed neural network architecture is shown in Fig. 2, where we use a U-Net–style encoder–decoder that fuses image evidence with action parameters. We train with the Adam optimizer (learning rate 5×10^{-4}, weight decay 10^{-3}) using a batch size of 120 for 1000 epochs. The learning rate follows a multi-step schedule with milestones every 100 epochs, each decayed by a factor of 0.75. We render each candidate stroke as a rasterized, disk-stamped footprint. Along the centerline—either a straight segment or a quadratic arc, depending on the absolute bend—we place N uniformly spaced centers and stamp filled circles at each center. The circle radius follows a force–to–width power law:

$$r(F) = r_{\min} + k\,F^{\gamma}, \tag{2}$$

where F is the applied force, $r_{\min}$ is a baseline radius, k is a scale factor, and $\gamma > 0$ controls the nonlinearity. Colors are treated as grayscale for simplicity: we emit three identical RGB channels (no hue variation) for compatibility with RGB loaders, and fix the alpha channel to fully opaque. After rendering all circles, we resize the canvas to 100×100.

Baseline Implementation Details

The Heuristic-only method recovers a stroke's geometry directly from a binary mask without learning. We skeletonize the mask via iterative, topology-preserving thinning to obtain a 1 -pixel-wide centerline, then identify endpoints as skeleton pixels with exactly one 8-neighborhood neighbor (degree$=$ 1). With endpoints p_0, p_1, we define a local frame along the chord $p_0 \to p_1$, project skeleton points onto the tangent–normal basis, and estimate b (bend) as the signed maximal lateral deviation (thresholded to suppress noise); l (length) is $\|p_1 - p_0\|$ and α (angle) is the chord heading.

FRIDA-CNN maps a 3D geometric parameter vector (l, b, F) to a grayscale stroke mask. A `BatchNorm` layer followed by a lightweight MLP produces a flattened 100×100 field, which is reshaped and refined by a small convolutional layer, yielding $M \in [0,1]^{100\times100}$. Since M encodes geometry without translation and rotation, We transform M using the remaining 3D geometric parameter $(p_0(x_0), p_0(y_0), \alpha)$.

Baselines FRIDA-CNN, IMPASTO-LR, and Heuristics predict only the incremental stroke, rather than the full next state image as IMPASTO-UNet does. To place the stroke on a base image $B \in [0,1]^{100\times100}$ with per-sample grayscale value $c \in [0,1]$ and opacity $a \in [0,1]$, we form a convex combination controlled by an effective opacity $A_{\text{eff}} = \text{clip}(M \cdot a, 0, 1)$:

$$I_{\text{out}} \;=\; B \cdot (1 - A_{\text{eff}}) \;+\; c \cdot A_{\text{eff}}. \tag{3}$$

Color Prediction Module

In the single-stroke trajectory following pipeline, we build a difference mask S by thresholding base-target pixel differences and obtain per-pixel weights w_p from a distance transform. In the multi-step planning pipeline, we obtain S

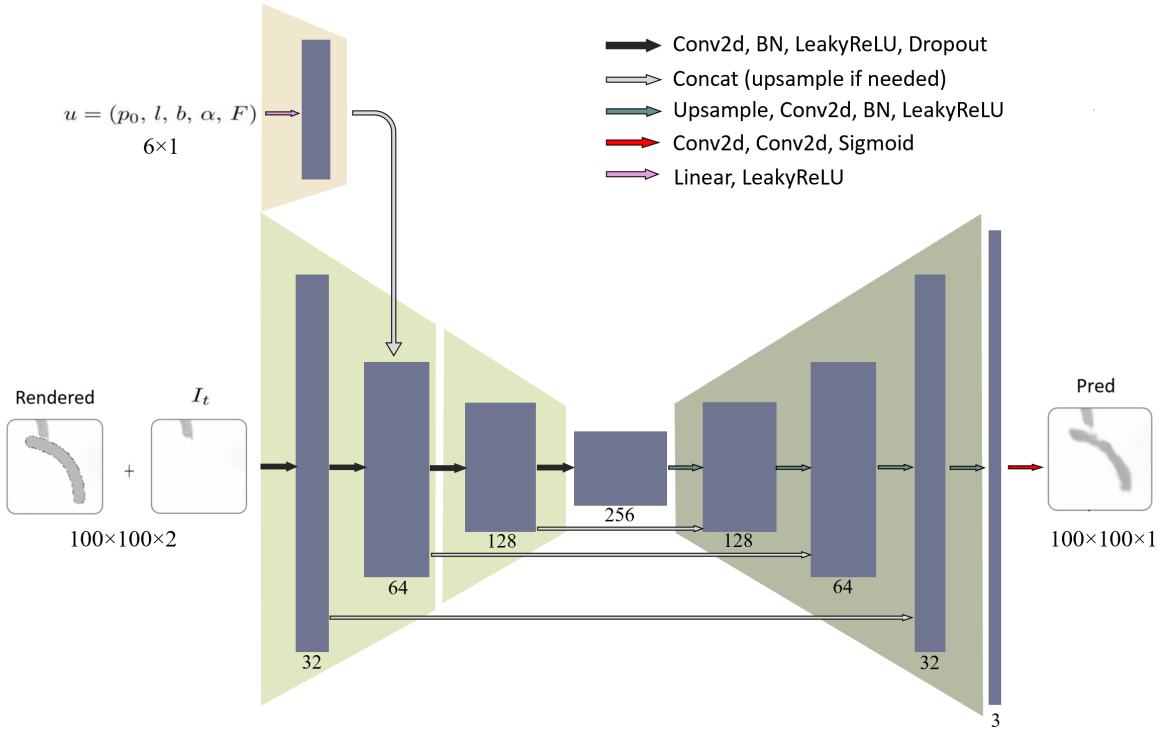

Fig. 2: The neural network architecture for the pixel dynamics model.

Fig. 3: A subset of the color patch database.

as the intersection between the foreground region of the pre-rendered predicted stroke and a disk centered at the stroke's geometric center:

$$S = \{\, p \in \Omega \mid M_{\text{pred}}(p) = 1 \ \wedge \ \|p - c\| \le r \,\}, \quad (4)$$

where M_{pred} is the binary mask of the pre-rendered stroke, c is its geometric center, and r is a chosen radius. The representative color for the stroke is the weighted mean over the mask,

$$\mu = \frac{\sum_{p \in S} w_p \, O_p}{\sum_{p \in S} w_p} \in \mathbb{R}^3, \quad (5)$$

where O_p is the target image in linear RGB at pixel p. We classify μ by nearest prototype in a 25×25 patch database (a subset is shown in Fig. 3) using the Mahalanobis metric. Covariances Σ_i for each label i are estimated from the corresponding patch images:

$$i^\star = \arg\min_i \, (\mu - \mu_i)^\top \, \Sigma_i^{-1} \, (\mu - \mu_i), \qquad C^\star = \mu_{i^\star}. \quad (6)$$

Transparency is decided by a simple ratio that compares the median magnitude of the observed change to the palette-to-base contrast for the selected color:

$$\text{ratio} = \frac{\text{median}_{p \in S} \|O_p - B_p\|_1}{\text{median}_{p \in S} \|C^\star - B_p\|_1 + \varepsilon}, \quad (7)$$

where we predict transparent if $\text{ratio} < \tau$, with B_p the base image in linear RGB, $\varepsilon > 0$ for numerical safety, and threshold τ.

Appendix C. Planning

PCA-RANSAC multi-way splitter

The PCA–RANSAC multi-way splitter constructs K disjoint stroke masks from the difference map D_t obtained from the base image I_t and the target image I_{t+n}. We segment D_t with k-means in a spatial feature space by embedding each pixel at $[x/W, \, y/H]$ where (x, y) are image coordinates and W and H are image width and height. Each resulting segment is treated as a candidate group of stroke pixels. For every segment we compute a local PCA frame; letting $\mu \in \mathbb{R}^2$ be the segment mean and $v_0, v_1 \in \mathbb{R}^2$ the principal directions, a pixel (x, y) is projected to (s, t) as

$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} v_0^\top \\ v_1^\top \end{bmatrix} \left([x, y]^\top - \mu \right). \quad (8)$$

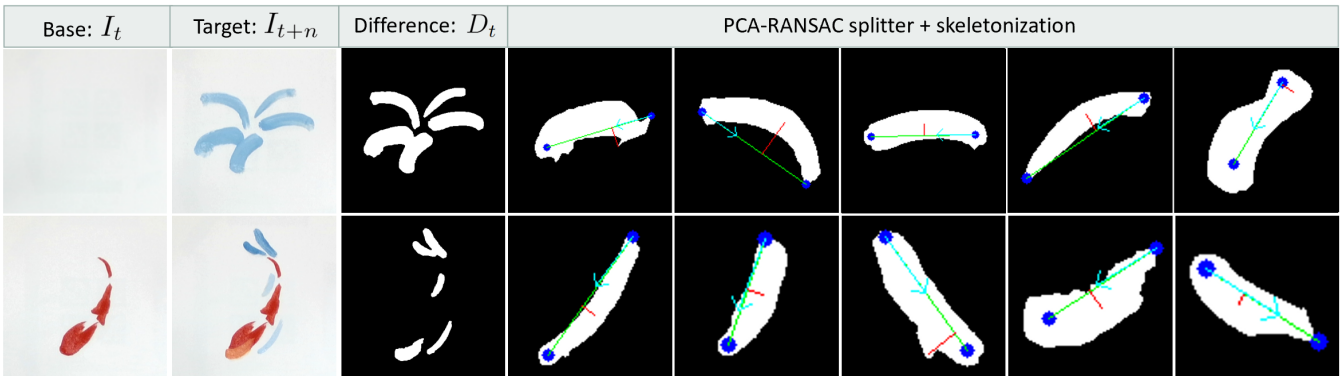

Fig. 4: Example visualizations produced by the PCA-RANSAC Splitter.

We apply Random Sample Consensus (RANSAC) to robustly fit either a line $t \approx p\,s + q$ or a quadratic $t \approx p\,s^2 + q\,s + r$ to the (s, t) samples. The RANSAC inliers define a thin candidate mask. We select K non-overlapping strokes by maximizing coverage of the unexplained region with penalties on curvature and mutual overlap,

$$\text{score} \;=\; \text{cover} \;-\; \lambda_b\,\text{bend} \;-\; \lambda_o\,\text{overlap}, \tag{9}$$

and thicken each selected mask into a single-connected stroke via centerline seeding and constrained region growing, yielding K heuristic initializations for the MPPI optimization. An example of the resulting visualization from the PCA-RANSAC Splitter is shown in Fig. 4.

Stochastic Sampling with CMA-Style Adaptive Covariance

At each MPPI optimization iteration, we assemble a candidate set by inserting the current nominal sequence $\mathbf{U}$ and drawing the remaining $K-1$ samples from a time-indexed Gaussian. Concretely, for each time index t we generate standard normal noise $z_t \sim \mathcal{N}(0, I)$, apply the current covariance structure via a Cholesky factor A_t of Σ_t, where $\Sigma_t = A_t A_t^\top$, and form perturbed controls

$$\tilde{U}_t = U_t + \sigma_t A_t z_t. \tag{10}$$

The full sequences are clipped to action bounds. This yields a batch of candidates centered at $\mathbf{U}$ but shaped by (σ_t, Σ_t) at every timestep.

From all K candidates, we select the top K_e by terminal cost and compute temperature-weighted softmax weights with temperature β over the elite set. Using those weights, we update the nominal sequence via an exponential moving average (EMA) toward the elite mean and perform CMA-style per-timestep adaptation: the step-size path $p_s(t)$ updates σ_t, and the covariance path $p_c(t)$ together with a rank-μ term updates Σ_t.

Appendix D. Additional Results

For the most complex painting shown in the video (180 strokes), the robot follows the human artist step by step to complete this artwork. Additional visualization results are shown in Fig. 5 and 6, comparing human strokes with robot strokes.

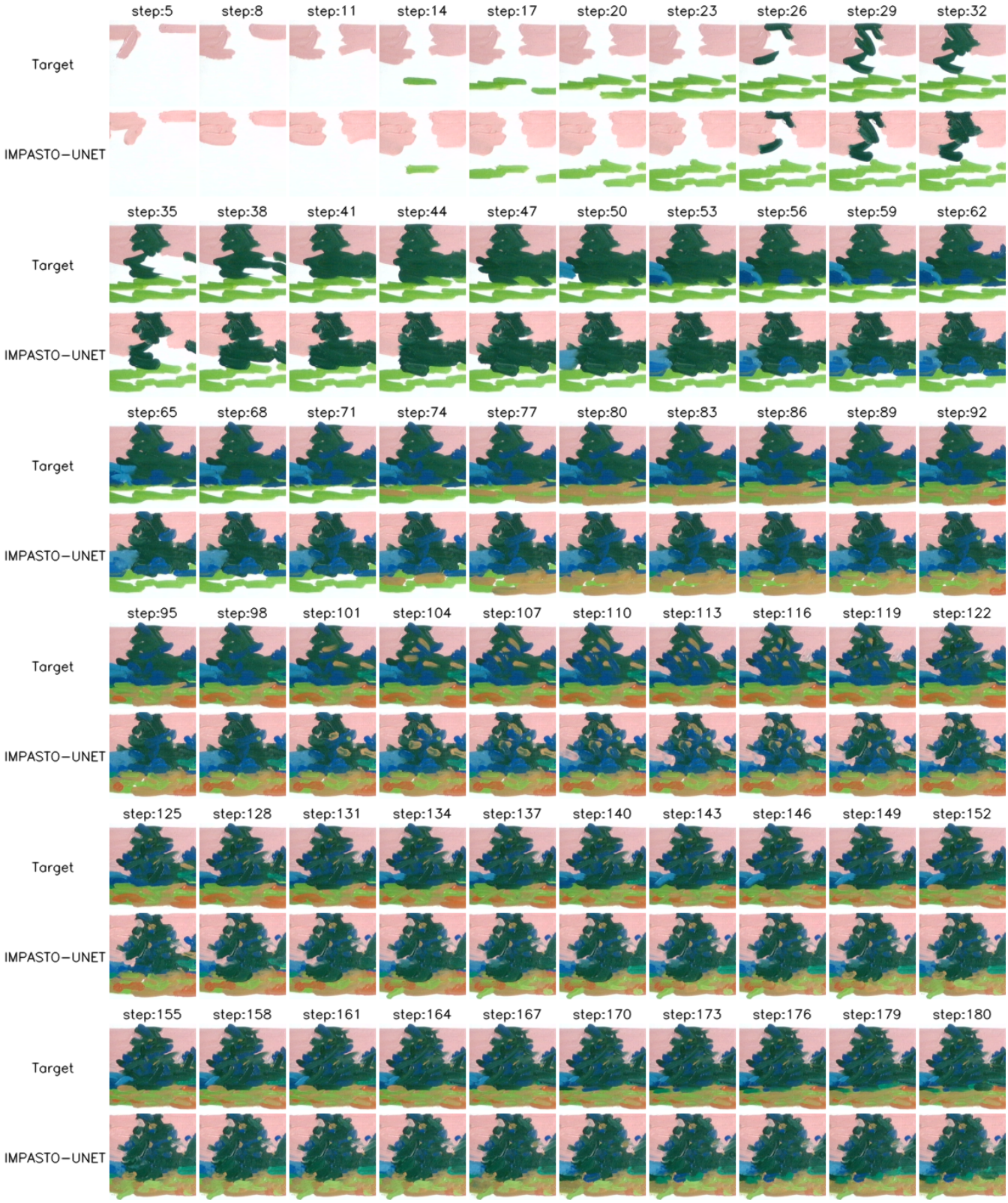

Fig. 5: IMPASTO is able to replicate a complex oil painting with 180 strokes (not all steps are shown).

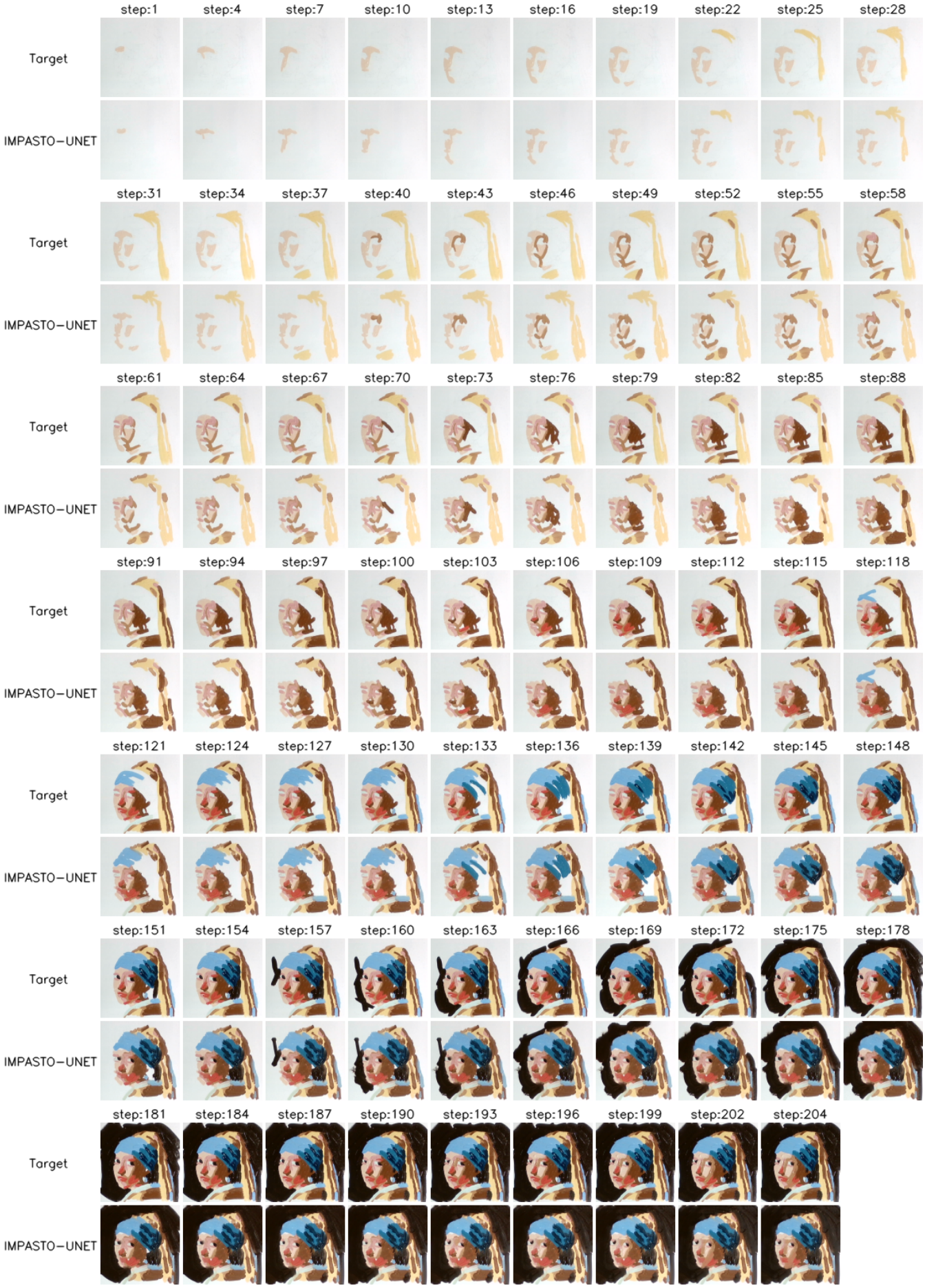

Fig. 6: IMPASTO is able to replicate a complex oil painting with 204 strokes (not all steps are shown).