Supplementary Material
Intelli-Paint: Towards Developing More Human-Intelligible Painting Agents

Jaskirat Singh[1,2], Cameron Smith[2], Jose Echevarria[2], and Liang Zheng[1]

[1] Australian National University
[2] Adobe Research

A Analysis: Progressive Layering

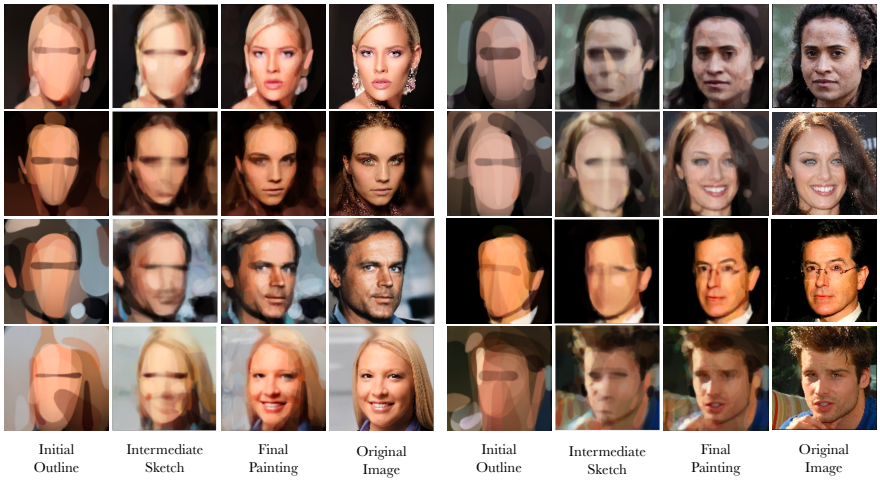
The progressive layering module forms one of the key components of the *intelli-paint* pipeline, wherein much like a human artist, it allows for a multi-layered evolution of the painted canvas. In this section, we provide a discussion on different aspects of the progressive layering module in order to aid a more in-depth understanding of the proposed approach. First, we analyse the use of progressive layering for achieving artistic painting evolution for different domains in Appendix A.1. We then discuss a formulation for extending the proposed layering mechanism for $L > 2$ layers and show its application to achieve a more detailed layering on images with multiple foreground objects (refer Appendix A.2).

A.1 Artistic Painting Evolution

By allowing the final painting agent to draw a given scene in multiple successive layers, the progressive layering strategy facilitates for a more human-relatable evolution of the painted canvas [6]. We next illustrate the use of progressive layering on different domains and analyse some intermediate canvas representations achieved during the painting process.

Facial domain. Results are shown in Fig. 1a. We observe that instead of directly trying to minimize the pixel-wise distance between the painted canvas and the target image (as is done in previous works), our method takes a much more human-like approach to portrait generation. For instance, consider the second example (row-2) for the painting sequence in the right column. Instead of directly painting based on low-level features (*e.g.* white brushstrokes for the mouth region), our method first draws a rough outline for the facial shape. It then refines this outline whilst indicating (but not drawing) the potential locations of important focalpoints (eyes, nose, lips) through appropriate facial shading. Only after the intermediate sketch has been set up, does it then proceed to add the fine-grain details on the exact facial features (eyes, nose, mouth, *etc*).

General images. Illustrations of progressive layering applied to general object images are shown in Fig. 1b. Mimicking the layering approach often observed among human artists [6], the *intelli-paint* agent first begins by drawing a natural background scene, before adding in the foreground objects in the second layer.



(a) Progressive layering for facial domain.

(b) Progressive layering in a more general context.

Fig. 1. Analysing progressive painting evolution for different domains. In contrast with previous works which directly try to minimize the $\mathcal{L}_{pixel}$ distance between the painted canvas and the original image, our method allows for a more human-like evolution of the painted canvas across different domains. For instance, for facial domain (a) it first begins with a rough face outline shown in Col-1. It then refines this outline further to paint the artistic intermediate representation in Col-2, before finally adding all facial details (eyes, nose, lips) in the final painting (Col-3).

A.2 Extending Progressive Layering for Multiple Layers

The progressive layering formulation discussed in the main paper primarily divides the painting process into two broad layers (background and foreground). In this section, we show that the original two layered formulation can be easily extended to $L > 2$ layers by using ranked saliency maps [8] for target image I.

(a) Original Image *(b) Layer 1* *(c) Layer 2* *(d) Layer 3*

Fig. 2. Progressive layering for $L > 2$ layers. The agent first begins by painting a realistic background scene (layer 1 in column-b). Once the background layer is drawn, the painting agent in each successive layer (layers 2-3) then proceeds to add different foreground objects in decreasing order of saliency.

In particular, consider a ranked saliency map $\mathcal{S}_I$, such that $\{\mathcal{S}_I[k]\}_{k=1}^{L}$ indicate the salient regions for the target image I, ranked in increasing order of saliency. The progressive layering strategy can then be trivially extended to $L > 2$ layers by using the following layered-mask $\mathcal{M}_I(l)$ (refer Eq. 4, 5 in the main paper) for each layer l,

$$\mathcal{M}_I(l) = 1 - \bigcup_{k=1}^{L-l} \mathcal{S}_I[k]. \tag{1}$$

Results for progressive painting sequences while using $L > 2$ layers are shown in Fig. 2. For instance consider the first example (refer row-1, Fig. 2). The painting process is divided in three layers. We observe that in the first layer, the painting agent begins by drawing a realistic background scene (describing road, grass) on the canvas. In the second layer, it then proceeds to add the most salient object (white car) on to the canvas. Finally, in the third layer it shifts its attention to the least salient object (cow) in order to complete the final painting.

B Ablation Studies

The *Intelli-paint* pipeline utilizes three main modules for mimicking the human painting style: 1) progressive layering, 2) sequential brushstroke guidance and 3) brushstroke regularization. In this section, we aim to understand the contribution of each of these modules in the development of a more human-relatable painting process. However, since the main focus of our method is the recreation of the human artistic creation process, we argue that the effect of each module is best understood by visualizing the corresponding painting sequences. We thus refer the readers to the official project webpage for detailed visualizations on the importance of each module in the human-like painting process.

Method	Intelli-Paint Preference
Ours (w/o prog. layering)	89.20 %
Ours (w/o seq. guidance)	77.39 %
Ours (w/o stroke reg.)	84.54 %

Table 1. User-Study Results for Ablation Study: Showing % of painting samples for which human users prefer full intelli-paint system over its different ablations.

Additionally, we also report quantitative results by performing a human-user study wherein participants are asked to compare the full *intelli-paint* pipeline and its different ablations. Results are reported in Table 1. Please note that in order to aid easy comparison, user study {w/o sequential guidance} was performed while indicating a bounding box for the current brushstroke, in order to clearly highlight the area under focus by the painting agent. We clearly observe that all three modules form a key component of the overall *intelli-paint* pipeline, and, the removal of any of these modules leads to reduced performance for the overall painting algorithm. Also, please note that due to the pair-wise nature of comparisons performed during the user-study the final results are not directly comparable to those of other painting methods (Table 2 of main paper). This is because while previous methods possess a significantly different painting style as compared to *intelli-paint*, the removal of a single component (as evident from included videos) makes the inefficiencies of an ablated version (as compared to the full *intelli-paint* pipeline) quite evident to a careful human user.

C Algorithm Details

In this section, we further elaborate on some algorithm details which could not be included (or fully explained) due to space constraints in the main paper.

C.1 Sequential Brushstroke Guidance

Algorithm 1 Foreground Object Selection

Input: Foreground selection convex coefficients $\boldsymbol{\alpha}^t$; In-image bounding box detections $\mathcal{B}_i \in \mathbb{R}^4, i \in [1, N]$.
Output: Coarse object attention window $\mathcal{G}_t$.
Defaults: Bounding box over the entire image $\mathcal{B}_0$.

function ObjectSelect($\boldsymbol{\alpha}^t, \{\mathcal{B}_0, \dots \mathcal{B}_N\}$)
  $\{\alpha_0^t, \ \dots \ \alpha_N^t\} = \boldsymbol{\alpha}^t \in \mathbb{R}^{N+1}$;
  $\mathcal{G}_t = \sum_{i=0}^{N} \alpha_i^t \ \mathcal{B}_i$;
  return $\mathcal{G}_t$.
end function

The sequential brushstroke guidance strategy (discussed in Sec. 4.1.3 of main paper), allows the reinforcement learning based sequential planner agent to shift its attention between different image regions through a sequence of coarse-to-fine

Algorithm 2 Markov Updates for Local Attention Window

Input: Current coarse, local attention windows $(\mathcal{G}_t, \mathcal{W}_t)$; Markovian bounding box refinements $\Delta\mathcal{W}_t$.
Output: Updated local attention window $\mathcal{W}_{t+1}$.
Defaults: $w_{min} = h_{min} = 0.2$.

function MARKOVUPDATE($\mathcal{W}_t, \mathcal{G}_t, \Delta\mathcal{W}_t$)

$x_t^{\mathcal{G}}, y_t^{\mathcal{G}}, w_t^{\mathcal{G}}, h_t^{\mathcal{G}} = \mathcal{G}_t$;
$x_t^{\mathcal{L}}, y_t^{\mathcal{L}}, w_t^{\mathcal{L}}, h_t^{\mathcal{L}} = \mathcal{W}_t$;
$\Delta x_t, \Delta y_t, \Delta w_t, \Delta h_t = \Delta\mathcal{W}_t$;

$x_{t+1}^{\mathcal{L}} = x_{t+1}^{\mathcal{G}} + (x_t^{\mathcal{L}} + \Delta x_t)\ w_{t+1}^{\mathcal{G}}$;
$y_{t+1}^{\mathcal{L}} = y_{t+1}^{\mathcal{G}} + (y_t^{\mathcal{L}} + \Delta y_t)\ h_{t+1}^{\mathcal{G}}$;
$w_{t+1}^{\mathcal{L}} = (max(1-\tilde{t}, w_{min}) + \Delta w_t)\ w_{t+1}^{\mathcal{G}}$;
$h_{t+1}^{\mathcal{L}} = (max(1-\tilde{t}, h_{min}) + \Delta h_t)\ h_{t+1}^{\mathcal{G}}$;

$\mathcal{W}_{t+1} = x_{t+1}^{\mathcal{L}}, y_{t+1}^{\mathcal{L}}, w_{t+1}^{\mathcal{L}}, h_{t+1}^{\mathcal{L}}$;
return $\mathcal{W}_{t+1}$.

end function

attention windows $\{\mathcal{W}_0, \mathcal{W}_1 \ldots \mathcal{W}_T\}$. The computation of the localized attention window $\mathcal{W}_t$ at any timestep t is done in the following broad steps.

Foreground object selection. The RL agent first selects the in-focus foreground object by predicting coordinates of a coarse global attention window $\mathcal{G}_t$. Given an input image I with N foreground objects, the computation of the coarse attention window $\mathcal{G}_t$ can be expressed through Algorithm 1.

Local attention window selection. Within each object window $\mathcal{G}_t$, the agent further learns to sequentially shift its focus on different in-object features through a sequence of coarse-to-fine local attention windows $\mathcal{W}_t$. In particular, given the coarse object window coordinates $x_t^{\mathcal{G}}, y_t^{\mathcal{G}}, w_t^{\mathcal{G}}, h_t^{\mathcal{G}}$, the coordinates $\mathcal{W}_t = x_t^{\mathcal{L}}, y_t^{\mathcal{L}}, w_t^{\mathcal{L}}, h_t^{\mathcal{L}}$ for the finer localized attention windows are computed using the Markov update function in Algorithm 2.

Brushstroke parameter adjustment. Finally, the coordinates of local attention window are used to modify the predicted brushstroke parameters $\mathbf{a}_t^l$, so as constrain the painting agent to only draw within the localized attention window. This procedure can be expressed as,

$$\mathbf{a}_t^l \leftarrow \text{PARAMADJUST}(\mathbf{a}_t^l, \mathcal{W}_t). \tag{2}$$

Assuming that $\mathbf{a}_t^l$ at each timestep t, depicts a 13 dimensional vector that represents the parameters of a quadratic Bézier curve [2,7] as follows,

$$\mathbf{a}_t^l = (x_0, y_0, x_1, y_1, x_2, y_2, z_0, z_2, w_0, w_2, r, g, b), \tag{3}$$

where the first 10 parameters depict stroke position, shape and transparency, while the last 3 parameters form the *rgb* representation for the stroke color. The parameter adjustment function is then implemented as per Algorithm 3.

Algorithm 3 Parameter Adjustment Function

Input: Initial brushstroke prediction $\mathbf{a}_t^l$; current attention window coordinates $\mathcal{W}_t$.
Output: Modified brushstroke prediction vector $\mathbf{a}_t^l$.

function ParamAdjust($\mathbf{a}_t^l, \mathcal{W}_t$)

$x_0, y_0, x_1, y_1, x_2, y_2, z_0, z_2, w_0, w_2, r, g, b = \mathbf{a}_t^l$;
$x_t^{\mathcal{L}}, y_t^{\mathcal{L}}, w_t^{\mathcal{L}}, h_t^{\mathcal{L}} = \mathcal{W}_t$;

$x_0 = x_t^{\mathcal{L}} + x_0 \cdot w_t^{\mathcal{L}}$;
$y_0 = y_t^{\mathcal{L}} + y_0 \cdot h_t^{\mathcal{L}}$;
$x_2 = x_t^{\mathcal{L}} + x_2 \cdot w_t^{\mathcal{L}}$;
$y_2 = x_t^{\mathcal{L}} + y_2 \cdot h_t^{\mathcal{L}}$;
$w_0 = w_0 \cdot avg(w_t^{\mathcal{L}}, h_t^{\mathcal{L}})$;
$w_2 = w_2 \cdot avg(w_t^{\mathcal{L}}, h_t^{\mathcal{L}})$;

$\mathbf{a}_t^l = x_0, y_0, x_1, y_1, x_2, y_2, z_0, z_2, w_0, w_2, r, g, b$;
return $\mathbf{a}_t^l$.

end function

C.2 Brushstroke Regularization

The current works on autonomous painting systems are often limited to using (an almost) fixed brush stroke budget irrespective of the complexity of the target image. Experiments reveal that this not only reduces the efficiency of the generated painting sequence but also results in redundant (or overlapping) brushstroke patterns (refer main paper) which impart an unnatural painting style to the final agent. To address this, we propose an inference-time brushstroke regularization strategy which refines and removes redundancies from the initial brushstroke sequence predictions $\mathbf{s}_{init}$ to output the most efficient stroke decomposition $\mathbf{s}_{pred}$ for each test image. This process can be summarized through Algorithm 4.

C.3 Overall Inference Algorithm

Given functions defined in Algorithm [1,2,3,4], the overall inference algorithm for the *Intelli-paint* pipeline can now be summarized as per Algorithm 5.

C.4 Hyperparameter Summary

Module	Hyperparameter	Value
Sequential Guidance	w_{min}, h_{min}	0.2
Stroke Regularization	γ	0.01
Seq. Planner Reward	μ	10
	η, λ	$ls(10^{-4}, 0.1)$

Table 2. Hyperparameter summary for Intelli-Paint. $ls(10^{-4}, 0.1)$ implies that coefficients η, λ are raised from 10^{-4} to 0.1 in a linear schedule during the training process.

Algorithm 4 Brushstroke Regularization Function

Input: A target image I; initial brushstroke sequence $\mathbf{s}_{init}$

$$\mathbf{s}_{init} = \{\mathbf{a}_t^l \mid 0 \leqslant l \leqslant L-1, 0 \leqslant t \leqslant T/L\}$$

.

Output: Refined brushstroke sequence $\mathbf{s}_{pred}$.
Defaults: Number of layers L; episode length T;
Number of iterations M, $C_{init} = C_0^{l=0} = \textsc{BlankCanvas}$;

function $\textsc{StrokeReg}(\mathbf{s}_{init}, I)$:
  $\{\mathbf{a}_t^l \mid 0 \leqslant l \leqslant L-1, 0 \leqslant t \leqslant T/L\} = \mathbf{s}_{init}$;
  $x_t^l \sim \mathcal{N}(0, 10^{-3}) \quad \forall l, \forall t$;
  for $0 \leqslant i \leqslant M$ **do**
    $\beta_t^l = \textsc{Sign}(x_t^l) \quad \forall t, l$;
    $C_{out} = \sum_{l=0}^{L-1} \sum_{t=1}^{T/L} C_t^l \odot (1 - \beta_t^l \; S_\alpha(\mathbf{a}_t^l)) + \beta_t^l \; S_{color}(\mathbf{a}_t^l)$;
    $\mathcal{L}_{total} = \mathcal{L}_2(I, C_{out}) + \gamma \sum_{l=0}^{L-1} \sum_{t=1}^{T/L} \|\beta_t^l\|_1$;
    $a_t^l \leftarrow a_t^l - \frac{\partial \mathcal{L}_{total}}{\partial a_t^l} \quad \forall t, l$;
    $x_t^l \leftarrow x_t^l - \frac{\partial \mathcal{L}_{total}}{\partial x_t^l} \quad \forall t, l$;
  end for
  $\mathbf{s}_{pred} = \{\beta_t^l \cdot \mathbf{a}_t^l \mid 0 \leqslant l \leqslant L-1, 0 \leqslant t \leqslant T/L\}$;
  return $\mathbf{s}_{pred}$.
end function

D Human User Study

In this section, we provide details about the human data collection process which is used to quantitatively demonstrate the improved human-like resemblance of our approach as compared to previous state of the art [2, 4, 7, 11].

User study setup. The user-study was conducted across 50 unique Amazon Mechanical Turk [1] subjects wherein each human participant is shown a series of paired painting sequences comparing our method with previous works. For each pair, the human subject is then asked to select the painting sequence which best resembles the human painting style. As mentioned in the main paper, we perform two variations of the user study in order to better understand the human-intelligibility of our method: **1)** User-Study A, where subjects are also provided with a human painting sequence to act as reference in their decision-masking, and **2)** User-Study B, where participants are only shown a pair of artificial painting sequences (ours vs competing method) and are thus asked to make the decision based on their own subjective understanding of the human painting style. Each painting sequence is presented as a *gif* image with a total duration of 10 seconds. Similar to qualitative results in Fig. 4 of the main paper, the number of brushstrokes for each method are chosen so as to ensure simi-

Algorithm 5 Overall Inference Algorithm for Intelli-Paint

Input: A target image I; image-saliency map $\mathcal{S}_I$;
number of layers L; painting episode length T.
Required: RL-based sequential-planner POLICY;

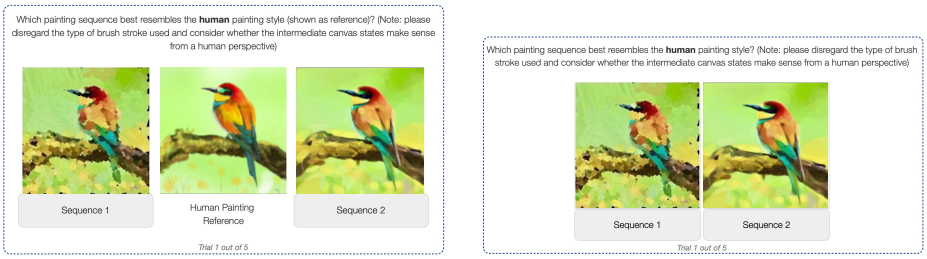
$\mathcal{W}_0 = x_0^{\mathcal{L}}, y_0^{\mathcal{L}}, w_0^{\mathcal{L}}, h_0^{\mathcal{L}} = (0, 0, 1, 1)$;
$C_{init} = C_0^{l=0} = \textsc{BlankCanvas}$;
for $0 \leqslant l \leqslant L-1$ **do**
  for $0 \leqslant t \leqslant T/L$ **do**
    $s_t = (I, C_t^l, \mathcal{G}_t, \mathcal{W}_t, \mathcal{S}_I, l)$;
    $\mathbf{a}_t^l, \boldsymbol{\alpha}^t, \Delta\mathcal{W}_t = \textsc{Policy}(s_t)$;
    $\mathcal{G}_t = \textsc{ObjectSelect}(\boldsymbol{\alpha}^t, \{\mathcal{B}_0, \ldots \mathcal{B}_N\})$;
    $\mathcal{W}_t = \textsc{MarkovUpdate}(\mathcal{G}_t, \mathcal{W}_{t-1}, \Delta\mathcal{W}_t)$;
    $\mathbf{a}_t^l \leftarrow \textsc{ParamAdjust}(\mathbf{a}_t^l, \mathcal{W}_t)$;
    $C_{t+1}^l = C_t^l \odot (1 - S_\alpha(\mathbf{a}_t^l)) + S_{color}(\mathbf{a}_t^l)$;
  end for
end for
$\mathbf{s}_{init} = \{\mathbf{a}_t^l \mid 0 \leqslant l \leqslant L-1, 0 \leqslant t \leqslant T/L\}$;
$\mathbf{s}_{pred} = \textsc{StrokeReg}(\mathbf{s}_{init}, I_{target})$;
return $\mathbf{s}_{pred}$.

lar reconstruction loss between the output canvas and the target image. Fig. 3 illustrates the basic interface setup for both user-studies.

Response filtering. Since the choice on human-likeliness of a painting sequence is subjective, we found that unfiltered collection of data from Amazon M-turk [1] can lead to quite noisy responses, wherein many users simply select the responses at random in order to quickly collect the assignment reward. To address this, we take the following measures for avoiding data collection noise. First, we limit the data collection to human subjects with a HIT rate [1] greater than 90%. In order to further refine the quality of collected data, we also limit the responses to users having a bachelors degree or equivalent. Furthermore, we intentionally use a repeated comparison (control seed) for each human participant. Responses of users who answer differently to this repeated comparison are discarded while reporting the final results in the main paper.

Challenges of an unbiased Turing test. Another way to measure the similarity with the human painting style is to perform a Turing test, wherein the performance of an approach is measured by the frequency with which it is able to fool an actual human user. However, for painting sequences, designing an unbiased Turing test is highly challenging due to several practical reasons,

– Human painting sequences invariably contain leading clues (*e.g.* artist moving or zooming-in to adjust the canvas for digital paintings and artist brushstroke, and, hand movements for paintings in physical medium), which make it almost impossible to conduct an unbiased Turing test.



(a) User Study A (b) User Study B

Fig. 3. Human User Study Interface. Left: User-Study A, where in addition to a pair of artificial painting sequences (*gif* format), the user is shown a human painting sequence to act as reference while making the decision. Right: User-Study B, where participants are only shown pair of artificial painting sequences and asked to select the one which would be most interpretable from a generic human perspective.

- The current painting methods [2,4,7,11] invariably use very primitive brush-stroke representations (Bézier curves in ours and [2, 7] or a fixed template based oil brushstroke for [4, 11]), which are easily distinguishable from the fluid drawing movements of actual human users.
- Procurement of large-scale full length painting sequences from real human artists is both challenging in its setup and quite time-consuming.

Thus, as mentioned in the limitations section (Sec. 6.3 of main paper), we do not claim that *intelli-paint* is truly human-like or is able to fool a human. Instead, we ask human users (User-Study A, B) to subjectively compare the human-like resemblance of our painting sequences as opposed to previous works, which provides a more continuous (and practical) evaluation strategy for judging the relatability of final painting sequences to actual human users.

E Time Efficiency Analysis in an Interactive Context

As discussed in the main paper, in this work we focus on the development of a more human-intelligible style in order to facilitate the use of autonomous painting systems in a more interactive context (*e.g.* a robotic painting application). However, in addition to the intelligibility of the painting sequences, the practical usability of a painting approach in an interactive context would also depend on the overall time required for painting (including both stroke-planning and execution) in the real world. In this section, we provide a time efficiency analysis in order to further understand the practical efficacy of the proposed approach for interactive applications (*e.g.* as a teaching tool for human users or robotic painting applications). The total painting time would depend on two factors,

1) Initial stroke planning time refers to the time taken by the painting agent to compose an actionable painting plan for the recreation of a given target image, and depends primarily on the inference time of the used painting algorithm. Results for initial stroke planning time for different painting methods are shown

Method	Initial Planning Time	Estimated Execution Time	Overall Painting Time
RL [2]	2.317 s	1933.5 s	1935.8 s
Semantic-RL [7]	2.631 s	1890 s	1892.6 s
Optim [11]	416.7 s	1170 s	1586.7 s
Transformer [4]	**1.154** s	1783.5[3]s	1784.6 s
Ours	72.21 s	**375** s	**447.2** s

Table 3. Method comparison w.r.t. overall painting time required for interactive / robotic painting applications. While our approach exhibits a slower initial planning time, it produces significant gains in the efficiency of the painting plan which leads to a notable reduction in the overall painting time for interactive painting applications.

in Col-2 of Table 3. We note that the use of gradient descent based stroke regularization causes our method to exhibit a slower inference time as compared to [2, 4, 7]. Nevertheless, we note that while both *intelli-paint* and Optim [11] require gradient descent based optimization, our method is considerably faster since it relies on high-quality initializations from the *sequential-planner* (SP) agent. In contrast, Optim [11] begins with a random or heuristic-based stroke initialization which takes gradient-descent optimization longer to converge.

2) Painting execution time is the time taken by the interactive agent (robotic or human) to implement the provided painting plan in practice. This would in turn depend highly on the *efficiency of the painting plan* produced by the autonomous painting agent. For instance, a painting plan (such as one from *intelli-paint*) which allows a robotic agent to paint the same level of final detail using lesser number of brushstrokes is going to be more faster (and practical) to implement than a plan which uses significantly more brushstrokes.

Consider a robotic agent trying to execute a given painting plan. Manual analysis of painting videos from modern robotic agents [3, 5, 10] reveals that adding a single brushstroke on the canvas takes somewhere from 1.5 to 5 seconds. This includes time taken for retracting the robotic arm from the current location, selecting (or mixing) appropriate colors for the next brushstroke and moving the robotic arm to the next stroke location. Even assuming a conservative estimate of 1.5 seconds per brushstroke, this implies that generating a final canvas with a mere ∼1000 brushstrokes (previous works typically use a much larger stroke count) would require a minimum of 25-30 minutes for the robotic agent.

Table 3 (Col-3) reports the estimated painting execution times across different painting methods. The number of brushstrokes for each method is chosen so as to ensure similar quality canvases (as measured by $\mathcal{L}_{pcpt}$ loss) across 100 randomly sampled images from the CUB-Birds [9] birds dataset. All results are

[1] Represents a lower-bound estimate. For each method, we select the min. number of strokes required to achieve similar output performance. While this is easily estimated for [2,7,11], the same is not trivial for [4] which exhibits a huge increase in number of brushstrokes (N) between K=3 ($N \approx 1260$) to K=4 ($N \approx 4.8k$) grid levels. Thus, in order to prevent an unfair comparison, we choose the lower bound on number of strokes required for [4], while estimating the painting time.

reported while assuming a fixed brushtroke budget of 250 brushstrokes for *intelli-paint* and a conservative estimate of requiring 1.5 seconds per brushstroke.

Overall painting time includes both initial painting time and the estimated execution time in the real world. Results are shown in Col-4 of Table 3. We observe that while *intelli-paint* exhibits a slower initial planning time, it produces significant gains in the efficiency of the painting plan which leads to a notable reduction in the overall painting time required in an interactive context.

Final Notes.

- The above analysis only presents a conservative estimate on the benefits of *intelli-paint* in an interactive context. In practice, the actual painting execution time would also depend on use of similar colors and spatial nearness of consecutive brushstrokes (to reduce time required between application of brushstrokes), both of which are explicitly enforced through spatial and color penalties $\{r_t^{spatial}, r_t^{color}\}$ in the *sequential-planner* (SP) agent.
- Please note while the above analysis is mainly performed from the context of robotic painting applications, a similar analysis would also apply to novice human users trying to recreate a painting plan taught by an artificial agent. For instance, a painting teaching tool which teaches a novice user to paint a vivid scene in 200-300 brushstrokes (such as ours) is going to be much more appealing and faster to learn from, than a painting agent which shows how to paint a similar quality output in 5k-10k brushstrokes.

References

1. Crowston, K.: Amazon mechanical turk: A research tool for organizations and information systems scholars. In: Shaping the future of ict research. methods and approaches, pp. 210–221. Springer (2012) 7, 8
2. Huang, Z., Heng, W., Zhou, S.: Learning to paint with model-based deep reinforcement learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 8709–8718 (2019) 5, 7, 9, 10
3. Kite-Powell, J.: This ai robot will paint a canvas at sxsw 2021 (Mar 2021), https://www.forbes.com/sites/jenniferhicks/2021/03/10/this-ai-robot-will-paint-a-canvas-at-sxsw-2021/?sh=5b1f0d1ab449 10
4. Liu, S., Lin, T., He, D., Li, F., Deng, R., Li, X., Ding, E., Wang, H.: Paint transformer: Feed forward neural painting with stroke prediction. arXiv preprint arXiv:2108.03798 (2021) 7, 9, 10
5. Nemire, B.: Ai painting robot (May 2017), https://developer.nvidia.com/blog/ai-painting-robot/ 10
6. Reyner, N.: How to paint with layers - in acrylic & oil (Dec 2017), https://nancyreyner.com/2017/12/25/what-is-layering-for-painting/ 1
7. Singh, J., Zheng, L.: Combining semantic guidance and deep reinforcement learning for generating human level paintings. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 5, 7, 9, 10
8. Siris, A., Jiao, J., Tam, G.K., Xie, X., Lau, R.W.: Inferring attention shift ranks of objects for image saliency. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 2
9. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011) 10
10. Wikipedia contributors: Ai-da (robot) — Wikipedia, the free encyclopedia (2022), https://en.wikipedia.org/w/index.php?title=Ai-Da_(robot)&oldid=1070639724, [Online; accessed 7-March-2022] 10
11. Zou, Z., Shi, T., Qiu, S., Yuan, Y., Shi, Z.: Stylized neural painting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15689–15698 (2021) 7, 9, 10