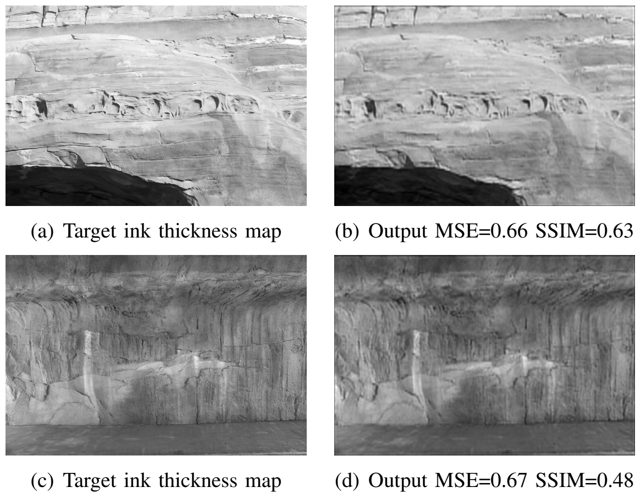
Mobile Robotic Painting of Texture

Majed El Helou[1], Stephan Mandt[2], Andreas Krause[3] and Paul Beardsley[4]

***Abstract*— Robotic painting is well-established in controlled factory environments, but there is now potential for mobile robots to do functional painting tasks around the everyday world. An obvious first target for such robots is painting a uniform single color. A step further is the painting of textured images. Texture involves a varying appearance, and requires that paint is delivered accurately onto the physical surface to produce the desired effect. Robotic painting of texture is relevant for architecture and in themed environments.**

A key challenge for robotic painting of texture is to take a desired image as input, and to generate the paint commands to as closely as possible create the desired appearance, according to the robotic capabilities. This paper describes a deep learning approach to take an input ink map of a desired texture, and infer robotic paint commands to produce that texture.

We analyze the trade-offs between quality of reconstructed appearance and ease of execution. Our method is general for different kinds of robotic paint delivery systems, but the emphasis here is on spray painting. More generally, the framework can be viewed as an approach for solving a specific class of inverse imaging problems.


(a) Target ink thickness map (b) Output MSE=0.66 SSIM=0.63

(c) Target ink thickness map (d) Output MSE=0.67 SSIM=0.48

Fig. 1. Test results of the setup presented in Section V-C for two rock textures. On the left are images that show target intensity / paint thickness, on the right are the reconstructed images realized with feasible spray patterns. The proposed approach allows for a trade-off between the painting quality and efficiency measures like the needed execution time.

I. INTRODUCTION

The motivation for this work is robotic painting and specifically (a) painting of large-scale environments with a mobile robot, (b) painting of texture, not a single uniform color. Example uses would be in architecture to provide a painted textured finish on surfaces, or in themed environments that use painting to give man-made materials a natural appearance. Furthermore, the full goal is to paint on 3D objects, but this work focuses on 2D surfaces.

A prior assumption is the availability of a mobile robotic painting system, and the recent development of systems such as a spray painting quadrotor [1] is inspirational. However, the capabilities of such a system are limited to delivering paint on a surface at an accurate location. This still leaves the significant challenge of how to define the trajectory and corresponding painting commands of the robot, to produce a desired texture.

One approach would be telerobotics - a painter works via a VR/AR interface, in which the target surface is visible, to remotely control a mobile painting robot. An alternative is to create an algorithm which takes an image/graphic of a desired texture, plus a specification of the capabilities of the robotic painting system, and automatically generates a robot trajectory and painting commands to produce that desired texture as accurately as possible.

This paper is concerned with the algorithmic approach, and the approach is based on deep learning. The need for a large corpus of training data of human painter activity is circumvented by basing the approach on an autoencoder, requiring only a training set of images of the desired texture. The autoencoder's decoder learns to reconstruct a desired texture while being constrained by the painting capabilities of the robotic system. The painting commands can then be inferred for an input texture ink map by a pass through the encoder. Desired properties of the robot's traversal path are also favored during the training. In this paper, we assume that the robot will utilise a grid-like trajectory, and we determine painting commands that both reproduce the input ink map and are efficient to execute. Illustrative results are shown for rock-type texture in Fig. 1.

The focus of the paper is specifically spray painting. A typical spray nozzle produces a spray cone, resulting in a circular or elliptical spray pattern on the surface. The distribution of deposited paint within the spray pattern is Gaussian [2], [3]. The proposed method however applies to any painting mechanism that can be approximated through a convolution, including a brush, dabbing, rolling, or stippling.

The algorithm operates on single-intensity images. However, it extends to color images by decomposing into single color layers [4] and applying the algorithm per layer.

[1]Majed El Helou is a PhD candidate in the School of Computer and Communication Sciences, EPFL. The work was done during an internship at Disney Research Zurich. majed.elhelou@epfl.ch

[2] Stephan Mandt is an Assistant Professor of Computer Science at the University of California, Irvine.

[3] Andreas Krause is a Professor of Computer Science at ETH Zurich.

[4] Paul Beardsley was a Principal Research Scientist at Disney Research Zurich at the time of this work.

II. Related Work

Industrial robotic painting is historically among the first successful applications of robotics. Aesthetic robotic painting has appeared more recently. In both areas, the focus has been on smaller-scale surfaces and fixed robot arms, in contrast to the work here on painting large-scale environments with a mobile robot.

Paint application methods include stippling [5] in which the authors define a model for ink dots and provide a heuristic for approximating an image with these dots. Paint brushing is investigated in [6], using a regular brush and constant feedback from a camera, and using simulation-based predictions to add the best stroke at every time step. Other approaches present algorithms for watercolour [7] or graffiti stroke placement [8], with a visual control refinement stage in [9]. We focus in contrast on spray painting, particularly on machine-learned painting.

Computer-aided human spray painting includes [4], a painter assistant that automatically blocks the spray when inappropriate painting is going to take place for a target appearance, based on a live simulation and the target texture. In [10], the user is guided by a simulator to areas that still need painting, using a particle replacement policy for the simulation. A virtual environment for graffiti painting is described in [11], driven by a dummy nozzle held by a user.

Robotic spray painting of images or texture rather than uniform color paint is a recent development. An automated approach is presented in [3] - a robotic arm moves a spray can in front of a mural. The algorithm relies on a feedback mechanism where the current painting is fed back to the system after every spray to compute the next position. The approach is limited in scalability due to its slow speed as feedback and optimization steps are carried out after every applied spray pattern.

Spray models are essential when working with spray painting, to simulate the process and outcome of specific operations [10], [2], [3], [4]. An airbrush simulation is described in [12], [2]. The spray model in [12] is calibrated experimentally, taking into account factors such as air to paint ratio, viscosity, and distance of the airbrush from the work. A particle simulation model for spraying is presented in [13]. In [14], the authors experimentally create a database to learn the spray model for a nozzle, to create a virtual environment where ship painters can be trained. For such industrial applications, it is the paint thickness that is of most importance. A statistical analysis is carried out in [15] to learn the effects of key parameters - shaping air and paint flow - on the dry film thickness.

Learning applied to painting. Learning to compute a complete paint mission is a more scalable approach than methods that need iterative visual feedback to drive simulation and optimization steps after each spray step. For instance, in the area of style transfer, Johnson *et al.* propose a feed-forward strategy that replaces constant optimization to reach real-time speed [16]. However, as spray paint simulations are not readily differentiable, they cannot be easily integrated into a generic machine learning setup. A generative model, confined to using four strings to move a drawing pen in two dimensions, is presented in [17]. The authors add noise to the motor commands to generate more training data, until the distribution covers the MNIST dataset. DeepMind address the non-differentiability issue with a solution relying on reinforcement learning [18]. It treats the paint simulator as a black box, building on earlier work [19] that they scale up to real-world datasets. A discriminator assesses the realism of the simulated painting, and serves as the reward system. Their results demonstrate the drawing of small binary-colored digits or symbols with a controlled pen. However, generalizing to images is of rather low visual quality, and the training is expensive. It is in the order of 200 million frames, and must be repeated for every new paint delivery system. In contrast, our method has more tractable training times.

Paint mixing can be simulated using the long-established Kubelka-Munk (K-M) theory [20], [21], [22], [13], [23]. The task of painting colored textures reduces to painting a set of single-ink layers of texture.

III. Learning to Paint

A. Problem Setup

Our main objective is to enable painting large outdoor surfaces, with any target texture, no human intervention or feedback mechanisms, such that the automated painting is fast and scalable. The algorithm is suitable for use with a mobile robotic spray painting system equipped with a controllable spray nozzle [1], as spray painting is the most scalable approach to large, potentially 3D, structures.

Given a target digital RGB image to be painted and a set of inks made available to the painting system, an external paint-mixing simulator transforms the RGB colors into a sequence of ink thickness layers [13], [23]. We thus assume the input to our system is a sequence of layers of ink thickness to be achieved by the mobile robotic painting system. A given layer is a matrix of same size as the image, and contains the desired ink thicknesses in space for a specific ink.

The desired output of our system is a sequence of locations, per ink layer, of where the robot needs to be located and the corresponding pattern to be sprayed at every location. A pattern is the shape of the spray model but also its thickness. The former is determined by the nozzle orientation relative to the surface, the latter by the spray duration, pressure and robot speed. We assume the ink deposition follows a Gaussian distribution [2], [3], [24].

B. Painting as a Machine Learning Problem

Learning to paint presents multiple scalability advantages relative to optimization with simulators and constant feedback mechanisms. Recent research efforts have been directed at using neural networks to replace optimizations in certain applications [16], [25], or more general cases [26]. However, learning to paint requires a large dataset of painter-recorded actions (spraying location, distance to the surface, duration and pressure, nozzle orientation relative to the surface) combined with their respective outcomes on the sprayed surface. Such data are expensive and time consuming both to set up and to generate, even if carried out in virtual reality simulators. The quality and generalization of the results are also directly impacted by the quality of the data.

An alternative problem formulation relies on a generative model with a simulator that executes a latent set of commands to generate an image. The machine learning task is then to invert the simulation process: given an image, find the underlying commands that the simulator executed. However, with simulators being non-differentiable, training such a generative model is challenging. Reinforcement learning is highly costly to train, and must be retrained for different black-box simulators. Gradient estimation techniques such as [27] are only approximative and suffer from high variance [28], [29]. This paper proposes a different strategy based on autoencoders [30], [31], [32]. The bottleneck of the autoencoder has the interpretation of the command space, and a carefully designed decoder plays the role of the simulator. The encoder infers a set of painting commands given a target paint texture layer. As detailed in the next section, this architecture is fully differentiable and works well with limited amounts of texture images serving as training data.

C. *Differentiable Simulator*

As follows, we describe how we can design a differentiable simulator that mimics the behavior of the robotic system under consideration. A spray painting simulator essentially creates Gaussian paint patterns with magnitude proportional to the duration of spray, and covariance matrices dictated by the orientation of the nozzle. We consider a finite set of N spray patterns that the paint system can create. Each such pattern corresponds to a Gaussian with a two-dimensional covariance structure corresponding to a certain nozzle orientation and distance relative to the painted surface. We furthermore discretize the space of possible spraying locations to a grid G. The learning problem then amounts to estimating which of the N spraying patterns are to be applied at which locations in order to reconstruct a given image, and with what magnitudes. Simulating the spray painting, under the constraint of only using the N spray patterns, can now be written as a finite sum of convolutions. Each location in G holds an activation (impulse) for each of the N spray patterns; these impulses are constrained to the interval $[0, 1]$. If a spray pattern is not to be applied at the given location, its corresponding impulse is 0, otherwise, the impulse magnitude determines that of the spray. That is, the image becomes a mixture of Gaussians with pre-specified covariance structures.

The fact that the simulator assumes the form of convolutions with pre-specified kernels allows us to propagate gradients through the autoencoder and train it end to end. We use a fully convolutional neural network (CNN) architecture [33]. The implementation details are all grouped in Section IV. The CNN acts as an autoencoder, mapping the input image through a bottleneck layer to an output image. The decoder of the CNN emulates the spray paint simulator (Fig. 2) as explained above; its parameters are fixed and not learnable (the N spray patterns). The last convolution tensor is thus made up of N matrices, each holding a discretized two-variate Gaussian. The previous layer's outputs are the impulses or commands that drive the spray painting.

The discretization we make is two-fold. First, we discretize the possible spraying locations of the painting robot to restrict them to the grid G. This discretization is not costly, as the resolution of G can be as high as the input image. Second, we discretize the space of possible spray patterns. This discretization restricts the capabilities of the spraying system. However, it is readily possible to increase the sampling resolution by increasing N with no other changes needed, to leverage the full capabilities of the system. In practice, we try to restrict N to being as small as possible while obtaining acceptable performance. This speeds up the paint process as the robot does not need to repetitively re-orient its nozzle.

D. *Controlling Commands*

During the painting process, the robot moves over every location in the grid G holding a non-zero impulse and applies the corresponding spray pattern. Therefore, the path traversed and the spray time required by the robot naturally increase with the number of non-zero impulses.

We define unit spray patterns, each being a magnitude-scaled version of the corresponding Gaussian spray pattern, such that they correspond to the maximum duration t we allow for spraying a certain location. A zero impulse indicates the absence of spray, an impulse of 1 indicates the unit spray of duration t, and all values v in between are sprays with duration $v \cdot t$. A linear relation between ink thickness and time is assumed [2]. Time spent at a certain location can be translated to robot speed (Fig. 12 in [3]) or nozzle airflow around that location [10]. To minimize the overall duration of the spray operation, we regularize the tensor of spray commands (Section III-E). A further reduction in the number of commands is carried out by changing the grid resolution. Instead of assigning a command to every entry in G, we effectively downsample the grid, setting to 0 a percentage of the commands. For instance, a downsampling by 2 causes 75% of commands to be set to 0.

E. *Regularization*

Learning a set of commands for spray painting a target image is an ill-posed problem when the model is assumed to be a two-variate Gaussian. Within a certain error margin, multiple different combinations of space-shifted and magnitude-scaled Gaussians can lead to the same image. This makes the inverse problem of learning the spraying command locations and choices of spray patterns ill-posed. To reduce the negative effect of the ill-posed setup to the learning process, we assign different weights to the regularization of the different spray patterns. By adding a pattern-dependent weight to the command regularization, we break the equivalence between the different combinations that lead to a roughly similar output. This strategy can also

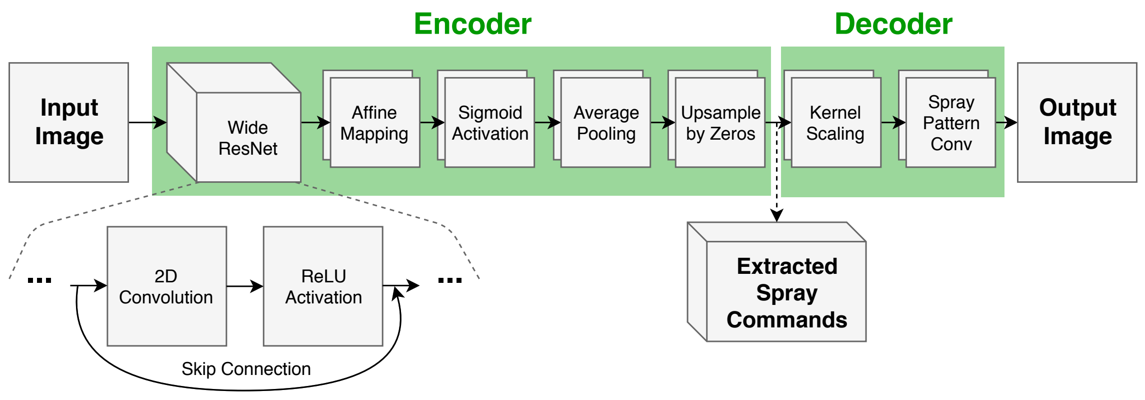


Fig. 2. Architecture of the constrained autoencoder convolutional neural network. The encoder is made up of a wide ResNet architecture containing a sequence of res-blocks [34]. The blocks are constituted of 2D convolutions and ReLU [35] activation layers with skip connections. The output is then passed through an affine mapping to shift the distribution of activation values similar to a batch normalization [36]. This affine mapping allows for a varied output in the range $[0, 1]$ after passing through the sigmoid activation [37]. The average pooling followed by upsampling with zero filling is only used when sparsity is to be imposed. The decoder scales the spray pattern magnitudes and applies our version of a spray simulator. The largest magnitude can be mapped in the physical world to the longest spray duration before droplets are formed.

allow us, for instance, to add more regularization weight to small spray patterns, thus encouraging the use of large ones. Another type of regularization we leverage is total variation (TV) [38], [39], regularization on commands to favor smooth changes [40]. Having smoothly-varying spray intensity is important for energy efficiency and execution practicality.

IV. IMPLEMENTATION

A. Neural Network Architecture

The CNN architecture we implement is fully convolutional. It is an autoencoder network with a constrained decoder. The constraint is that the decoder emulates a spray paint simulator that is reproducible by the given mobile robotic system. The autoencoder CNN follows the residual networks approach [41] but with a shallow and wide architecture [42] portrayed in Fig 2. In total, $51\,\mathrm{k}$ parameters are trainable in the network. The encoder outputs are the spraying commands since the decoder is forced to be a non-trainable layer that emulates spray painting.

B. Implementation Details

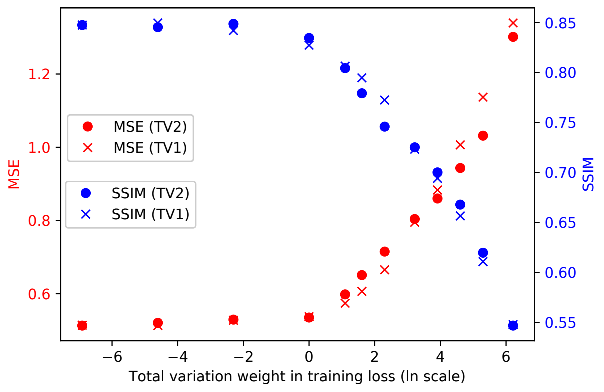
Dataset: We use a dataset of rockface images, with 894 images used for training and 141 used for testing. Training and testing sets are mutually exclusive. The image resolutions are either 1024×683 or 512×768 pixels, and are all normalized to the same mean intensity of 10. We take the luminance of every image as the guiding map for target ink thickness. Color intensity and ink thickness are not fully linearly related due to saturation. When a set of inks is available, a more accurate mapping can be created from color to ink thickness layers, as described in the color mixing literature. This mapping is outside the scope of this paper. We thus aim to solve the problem of painting a single-ink layer of texture which we choose to be the image luminance. The images are divided into 85×85 patches that are created with 50% overlap between each other as an image is traversed, and a batch size of 32 is used in the training process. The full test images can be fed-forward into the network, given large enough GPUs, as the network architecture is fully convolutional by design.

Code: The CNN and all functions used in this project are in Python 3.6.5, with Keras [43] for the CNN architecture, using tensorflow as backend [44]. We use CUDA 9.0.176, cuDNN 7.2 [45] and a GTX 1080 Ti GPU for training our CNN. We use the Adam optimizer [46] with a starting learning rate of 1×10^{-3} (default β, ϵ). Results shown in this paper are obtained with networks trained for only 10 epochs, unless stated otherwise. As over-fitting is not witnessed in our setup, we do not use any dropout or weight decay during the training. The loss function we use is a weighted sum between reconstruction mean-squared error (MSE), a weighted $\ell 2$ regularizer across different spray command maps, and spatial TV regularization on the command maps.

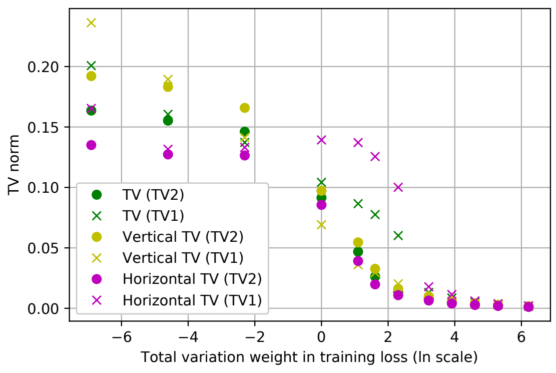
Running times: A single epoch on our entire dataset of $345,216$ patches trains in approximately $850\,\mathrm{s}$. So a single network is trained in less than two and a half hours for the 10 epochs and full training dataset. The feed-forward time is of $7\,\mathrm{ms}$ per patch and $3\,\mathrm{s}$ per test image on average.

V. EXPERIMENTAL RESULTS

The proposed approach is evaluated on the test dataset as described in Section IV. Aside from obtaining acceptable reconstruction quality, physical feasibility is an important criteria. For the commands to be more easily executed by the robotic system, we study both smoothly-varying commands and sparse commands. Commands that vary smoothly are more energy-efficient to execute as the robot's state changes gradually over time. Command sparsity, on the other hand, shortens the path to be traversed by reducing the points that need to be covered. All experiments are conducted using a set of three spray patterns: one symmetric Gaussian, and two ellipses skewed in the vertical and horizontal directions. The Gaussians have a variance of 3 pixels for the symmetric one, and variances of 2 and 4 in each direction for the ellipses. We also test with different diameter scales of spray patterns. A spray scale of value $[1, 3, 5]$ means the spray variances were



(a) MSE and SSIM



(b) Commands total variation

Fig. 3. Average test results with spray scale 1, evaluated with MSE and SSIM. The training is carried out with different losses: TV1 for a training loss with a vertical TV component, and TV2 for a training loss with a 2D TV component. Every data point is a separately-trained network. The trade-off between quality (a) and smoothness (b) is clear. Note that one-directional smoothness can be controlled through the training loss weight.

all multiplied by the corresponding value before the training. The smallest and largest scales correspond to physical spray patterns with roughly $10.4\,\mathrm{cm}$ and $23.2\,\mathrm{cm}$ diameters respectively, measured at three standard deviations.

Different reconstruction results and the corresponding commands are shown in Fig. 5. Reconstruction quality generally decreases with increasing smoothness or increasing sparsity. These trade-offs and quantitative assessments are presented in the following two sections.

A. Command Smoothness Trade-off

This section analyzes the trade-off between image reconstruction quality and command smoothness. The experiments are conducted for a set of different total variation regularization weights. The weights correspond to the weight of the total variation term in the training loss function. The total variation loss term for the network training is computed either vertically (TV1) or in two directions (TV2) and results are shown for each of these two training losses, and for every chosen weight. The Structural SIMilarity (SSIM) index [47] is often used in super-resolution assessment for the conservation of structural content rather than just pixel error [48], [49] and is used in assessing images when photo-realism is desirable [50]. As the network optimizes for MSE, we also show SSIM evaluation as an additional metric unknown to the network.

The trade-off is clear; as the variation in commands decreases with increasing regularization weight (Fig. 3 b), the reconstruction quality decreases (Fig. 3 a). The total variation in commands in Fig. 3 b is computed vertically, horizontally and an average of the two (for both TV1 and TV2 loss networks). When the training loss includes a two-directional TV term, the command results are smoother horizontally than vertically. This is due to the nature of our dataset composed of rock textures with many horizontal lines. We can control the variation in the direction that we want to traverse (Section V-C). As Fig. 3 b shows, we can make the commands smoother vertically by using only the vertical TV in the training loss (TV1). With small weights up to e^{-2}, the variation is largest for both trained networks vertically. But starting from a weight of 1, the variation distribution flips and becomes smaller vertically than horizontally for the network trained to minimize vertical variation (TV1). The horizontal variation is not affected by the vertical variation loss before a weight of e^2, beyond which, the command values converge to the average vertical value, which does not vary a lot from column to column, thus indirectly imposing horizontal smoothness. The one-directional smoothness is useful for the physical execution detailed in Section V-C. For instance, the quadrotor [1] can move more easily in straight lines vertically than horizontally, thus the need for vertically-directed smoothness. This is because it is equipped with the spray nozzle, and moving horizontally requires a movement in the body of the quadrotor that needs more compensation for the attached nozzle. This issue is less severe for vertical displacement. We however obtain similar effects when the smoothness is imposed horizontally.

B. Command Sparsity Trade-off

We also analyze the effect of sparsity in commands on the reconstruction quality. Fig. 4 shows MSE and SSIM results for different spray scales as a function of sparsity. Sparsity is defined by S, such that only one out of every S^2 commands is allowed to be non-zero. The reconstruction quality deteriorates quickly as sparsity increases for the small spray patterns (scale 1). This behavior is expected as it gets hard to spread the ink across the entire surface with a small spray pattern and high sparsity in commands. The problem is less severe with larger sprays. It is thus favorable to use larger patterns when we impose increasingly sparse commands.

C. Execution Time

The scale and resolution of the images in our dataset have roughly a $1\,\mathrm{px}$ to $1\,\mathrm{cm}$ correspondence. An image of width $1000\,\mathrm{px}$ corresponds to a $10\,\mathrm{m}$ spray surface width. The quadrotor [1] can reach speeds of $2\,\mathrm{m/s}$ but normally operates at only $30\,\mathrm{cm/s}$ when spray painting. The spray nozzle is mechanically controlled with a time resolution of

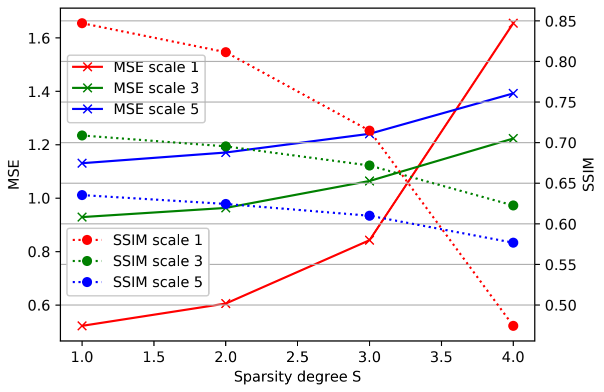
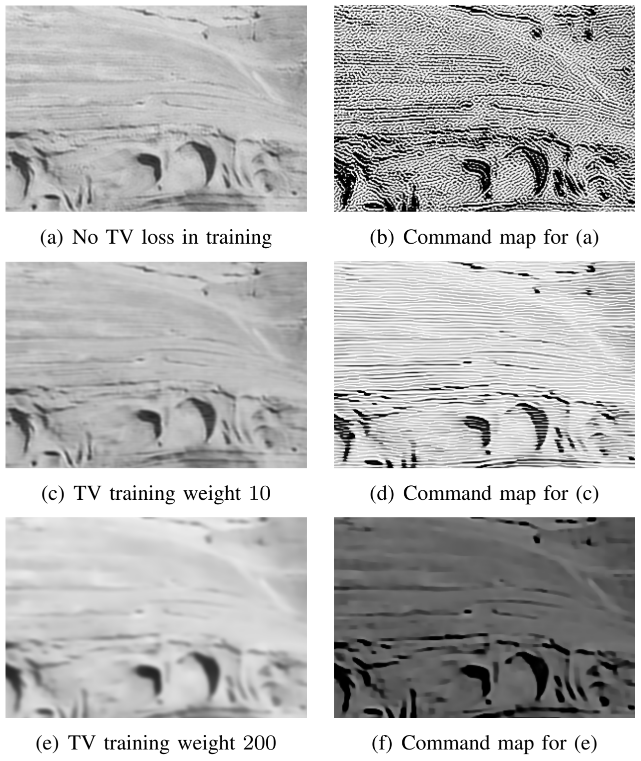


Fig. 4. Average reconstruction results as a function of increasing sparsity S for different spray pattern diameter scales. Only one out of every S^2 commands is non-zero.

$12\,\mathrm{ms}$ and can thus easily change its spray flow at a spacial resolution of $1\,\mathrm{cm}$ when the quadrotor is moving at $30\,\mathrm{cm/s}$.

Giving a command at every pixel in the image corresponds to an updated paint thickness every $1\,\mathrm{cm}$. Traversing the surface vertically takes the quadrotor $20\,\mathrm{s}$, with an added estimate of $2\,\mathrm{s}$ for stopping and positioning at the next column. Therefore, conducting the painting of a 1000×600 pixels (width $\times$ height) image on a $60\,\mathrm{m}^2$ area with a command for every pixel (i.e. every cm) requires $(20 + 2) * 1000 * 3/3600 = 18.33\,\mathrm{h}$ if we traverse every single column. The factor of 3 is because we are using three different spray patterns to create the image. They can be created by rotating the nozzle at every location, or by simply traversing the surface an extra time with an adjusted nozzle orientation. We focus on creating commands that are sparse horizontally to reduce the number of vertical traversals the quadrotor needs to carry out. Execution would be faster with horizontal traversals because the image is wide and there is a repositioning overhead. However, the nozzle-equipped quadrotor carries out vertical traversals in straighter lines than horizontally due to the needed compensation to account for the attached nozzle when tilting the quadrotor's body.

Based on the trade-off analysis, we choose to train a version of our network with a TV weight of 1 in both directions. We use the spray patterns of scale 5 and only allow one out of every 6 rows to be non-zero. Two reconstructed images with this setup are given in Fig. 1. Image (a) maps to $60\,\mathrm{m}^2$ of painted surface and can be sprayed in about $3.05\,\mathrm{h}$ (6x speed-up) by the quadrotor, that is $3.05\,\mathrm{min/m}^2$ compared with $70.1\,\mathrm{min/m}^2$ for the only other automated approach [3]. Average MSE over the test set is 1.32, average SSIM 0.58, and vertical TV 0.06, after 75 training epochs. We also evaluate our solution on the Pandora saliency dataset [51], while training only on the rockface dataset and obtain an MSE of 2.44 and SSIM of 0.62. SSIM is even slightly better than on the rockface dataset, showing the generalization of our solution. MSE is however worse, which is expected due to the details present in the paintings. One of these reconstruction results is illustrated in the submission video.



(a) No TV loss in training
(b) Command map for (a)
(c) TV training weight 10
(d) Command map for (c)
(e) TV training weight 200
(f) Command map for (e)

Fig. 5. Spray simulated results on the left, and the corresponding command maps (for one of the Gaussians) on the right. Results are obtained with TV2 loss networks trained for only 10 epochs. The image is cropped from Fig. 1 a. Command smoothness increases with little change in reconstruction quality, but with excessively smooth commands, the results begin to degrade. For TV1 loss networks, commands are smoother in the chosen direction and with very large TV1 regularization weights the reconstruction begins to show small line artifacts.

VI. Discussion

The approach proposed in this paper can also be applied to image inverse problems such as deblurring, deconvolution, or dehazing [52], as long as their simulation is integrated in the decoder. The main advantage of the approach is that it does not require an application-specific dataset. Furthermore, our autoencoder can be used for domain adaptation [53], [54]. The autoencoder acts as a style transfer operator [55], with a difference from earlier work that the system is generating not an image in a spray-painted style, but from specific spray commands based on the given patterns and regularization.

VII. Conclusion

This paper presented a method for inferring spray paint commands to paint a desired texture, specified as an input image. Our approach does not require training data beyond an easy-to-obtain unlabelled texture image dataset, and is generalizable to any paint application method. Only a one-time training is required for a new robotic painting system, and painting commands can then be inferred for any image with no further processing. We demonstrated the trade-off between image reconstruction quality and the ease of physical execution such that, for a specified quality of the painted appearance, an energy- or time-efficient painting mission can be executed.

References

[1] Anurag Sai Vempati, Mina Kamel, Nikola Stilinovic, Qixuan Zhang, Dorothea Reusser, Inkyu Sa, Juan Nieto, Roland Siegwart, and Paul Beardsley, "Paintcopter: An autonomous uav for spray painting on three-dimensional surfaces," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, 2018.

[2] Lorenzo Scalera, Enrico Mazzon, Paolo Gallina, and Alessandro Gasparetto, "Airbrush robotic painting system: Experimental validation of a colour spray model," in *International Conference on Robotics in Alpe-Adria Danube Region*, 2017, pp. 549–556.

[3] S Seriani, A Cortellessa, S Belfio, M Sortino, G Totis, and P Gallina, "Automatic path-planning algorithm for realistic decorative robotic painting," *Automation in Construction*, vol. 56, pp. 67–75, 2015.

[4] Roy Shilkrot, Pattie Maes, Joseph A Paradiso, and Amit Zoran, "Augmented airbrush for computer aided painting (cap)," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 2, p. 19, 2015.

[5] Brendan Galea, Ehsan Kia, Nicholas Aird, and Paul G Kry, "Stippling with aerial robots," in *Proc. of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*. Eurographics Association, 2016, pp. 125–134.

[6] Oliver Deussen, Thomas Lindemeier, Sören Pirk, and Mark Tautzenberger, "Feedback-guided stroke placement for a painting machine," in *Proc. of the 8th Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*. Eurographics Association, 2012, pp. 25–33.

[7] Lorenzo Scalera, Stefano Seriani, Alessandro Gasparetto, and Paolo Gallina, "Watercolour robotic painting: a novel automatic system for artistic rendering," *Journal of Intelligent & Robotic Systems*, pp. 1–16, 2018.

[8] Daniel Berio, Sylvain Calinon, and Frederic Fol Leymarie, "Learning dynamic graffiti strokes with a compliant robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3981–3986.

[9] Ren C Luo, Ming-Jyun Hong, and Ping-Chang Chung, "Robot artist for colorful picture painting with visual control system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2998–3003.

[10] Romain Prévost, Alec Jacobson, Wojciech Jarosz, and Olga Sorkine-Hornung, "Large-scale painting of photographs by interactive optimization," *Computers & Graphics*, vol. 55, pp. 108–117, 2016.

[11] Mei Yii Lim and Ruth Aylett, "My virtual graffiti system," in *IEEE International Conference on Multimedia and Expo (ICME)*, vol. 2, 2004, pp. 847–850.

[12] Jonathan Konieczny and Gary Meyer, "Airbrush simulation for artwork and computer modeling," in *Proc. of the 7th International Symposium on Non-Photorealistic Animation and Rendering*. ACM, 2009, pp. 61–69.

[13] Jonathan Konieczny, John Heckman, Gary Meyer, Mark Manyen, Marty Rabens, and Clement Shimizu, "Automotive spray paint simulation," in *International Symposium on Visual Computing*, 2008, pp. 998–1007.

[14] Ungyeon Yang, Gun A Lee, Seonhyung Shin, Sunyu Hwang, and Wookho Son, "Virtual reality based paint spray training system," in *Proc. of the IEEE Virtual Reality Conference (VR)*, 2007, pp. 289–290.

[15] KV Chidhambara, B Latha Shankar, *et al.*, "Optimization of robotic spray painting process parameters using taguchi method," in *IOP Conference Series: Materials Science and Engineering*, vol. 310, 2018, p. 012108.

[16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. of the IEEE European Conference on Computer Vision (ECCV)*, 2016, pp. 694–711.

[17] Vinod Nair and Geoffrey E Hinton, "Inferring motor programs from images of handwritten digits," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 515–522.

[18] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals, "Synthesizing programs for images using reinforced adversarial learning," *arXiv preprint arXiv:1804.01118*, 2018.

[19] Ning Xie, Hirotaka Hachiya, and Masashi Sugiyama, "Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting," *IEICE Transactions on information and systems*, vol. 96, no. 5, pp. 1134–1144, 2013.

[20] Paul Kubelka, "New contributions to the optics of intensely light-scattering materials. part i," *JOSA*, vol. 38, no. 5, pp. 448–457, 1948.

[21] ——, "New contributions to the optics of intensely light-scattering materials. part ii: Nonhomogeneous layers," *JOSA*, vol. 44, no. 4, pp. 330–335, 1954.

[22] Chet S Haase and Gary W Meyer, "Modeling pigmented materials for realistic image synthesis," *ACM Transactions on Graphics (TOG)*, vol. 11, no. 4, pp. 305–335, 1992.

[23] William Baxter, Jeremy Wendt, and Ming C Lin, "Impasto: a realistic, interactive model for paint," in *Proc. of the 3rd International Symposium on Non-Photorealistic Animation and Rendering*. ACM, 2004, pp. 45–148.

[24] Nijanthan Berinpanathan, Mina Kamel, Roland Siegwart, and Paul Beardsley, "Characterizing the spray coverage of nozzles," *ETHZ Semester Thesis: unpublished but available on request*, 2018.

[25] Jun Han, Salvator Lombardo, Christopher Schroers, and Stephan Mandt, "Deep probabilistic video compression," *arXiv preprint arXiv:1810.02845*, 2018.

[26] Joseph Marino, Yisong Yue, and Stephan Mandt, "Iterative amortized inference," *Proc. of the International Conference on Machine Learning (ICML)*, 2018.

[27] Ronald J Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[28] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber, "Policy gradients with parameter-based exploration for control," in *International Conference on Artificial Neural Networks*. Springer, 2008, pp. 387–396.

[29] Alexander Buchholz, Florian Wenzel, and Stephan Mandt, "Quasi-monte carlo variational inference," in *Proc. of the International Conference on Machine Learning (ICML)*, 2018, pp. 667–676.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*. MIT press, 2016.

[31] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[32] Zhiwei Deng, Rajitha Navarathna, Peter Carr, Stephan Mandt, Yisong Yue, Iain Matthews, and Greg Mori, "Factorized variational autoencoders for modeling audience reactions to movies," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2577–2586.

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[35] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. of the International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[36] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[37] Naiyan Wang and Dit-Yan Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 809–817.

[38] Curtis R Vogel and Mary E Oman, "Fast, robust total variation-based reconstruction of noisy, blurred images," *IEEE transactions on image processing*, vol. 7, no. 6, pp. 813–824, 1998.

[39] Tony Chan, Antonio Marquina, and Pep Mulet, "High-order total variation-based image restoration," *SIAM Journal on Scientific Computing*, vol. 22, no. 2, pp. 503–516, 2000.

[40] Antoni Buades, Bartomeu Coll, and J-M Morel, "A non-local algorithm for image denoising," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2005, pp. 60–65.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Identity mappings in deep residual networks," in *Proc. of the IEEE European Conference on Computer Vision (ECCV)*, 2016, pp. 630–645.

[42] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[43] François Chollet *et al.*, "Keras," 2015, software available from https://keras.io.

[44] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew

Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[45] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[46] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[47] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[48] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[49] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1646–1654.

[50] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network." in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017, p. 4.

[51] Corneliu Florea, Răzvan Condorovici, Constantin Vertan, Raluca Butnaru, Laura Florea, and Ruxandra Vrânceanu, "Pandora: Description of a painting database for art movement recognition with baselines and perspectives," in *Proc. of the IEEE European Signal Processing Conference (EUSIPCO)*, 2016, pp. 918–922.

[52] Majed El Helou, Frederike Dümbgen, Radhakrishna Achanta, and Sabine Süsstrunk, "Fourier-domain optimization for image processing," *arXiv preprint arXiv:1809.04187*, 2018.

[53] Amir Atapour-Abarghouei and Toby P Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 18, 2018, p. 1.

[54] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[55] Jing Yongcheng, Yang Yezhou, Feng Zunlei, Ye Jingwen, Yu Yizhou, and Mingli Song, "Neural style transfer: A review," *arXiv preprint arXiv:1705.04058v6*, 2018.