RoboCoDraw: Robotic Avatar Drawing with GAN-Based Style Transfer and Time-Efficient Path Optimization

Tianying Wang,[1,†] **Wei Qi Toh,**[1,†] **Hao Zhang**[1,†]
Xiuchao Sui,[1] **Shaohua Li,**[1,2] **Yong Liu,**[1,2] **Wei Jing**[1,3,‡]
[1]Artificial Intelligence Initiative, A*STAR
[2]Institute of High Performance Computing, A*STAR
[3]Institute of Information Research, A*STAR
1 Fusionopolis Way, Connexis North Tower, 138632, Singapore
21wjing@gmail.com

Abstract

Robotic drawing has become increasingly popular as an entertainment and interactive tool. In this paper we present RoboCoDraw, a real-time collaborative robot-based drawing system that draws stylized human face sketches interactively in front of human users, by using the Generative Adversarial Network (GAN)-based style transfer and a Random-Key Genetic Algorithm (RKGA)-based path optimization. The proposed RoboCoDraw system takes a real human face image as input, converts it to a stylized avatar, then draws it with a robotic arm. A core component in this system is the AvatarGAN proposed by us, which generates a cartoon avatar face image from a real human face. AvatarGAN is trained with unpaired face and avatar images only and can generate avatar images of much better likeness with human face images in comparison with the vanilla CycleGAN. After the avatar image is generated, it is fed to a line extraction algorithm and converted to sketches. An RKGA-based path optimization algorithm is applied to find a time-efficient robotic drawing path to be executed by the robotic arm. We demonstrate the capability of RoboCoDraw on various face images using a lightweight, safe collaborative robot UR5.

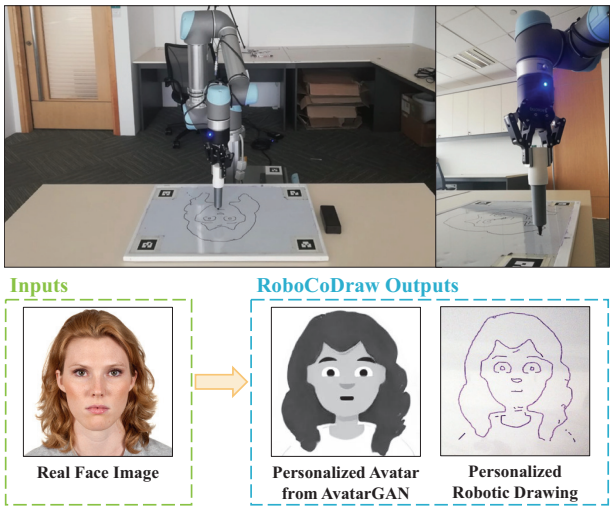

Figure 1: Top: the robot performing the drawing; Bottom: the generated personalized avatar and the drawing result.

Introduction

Robotic drawing is an increasingly popular human-robot interaction task that is fascinating and entertaining to the public. In a typical robotic drawing system, a robotic arm (Song, Lee, and Kim 2018) or a humanoid robot (Calinon, Epiney, and Billard 2005) is usually used in an interactive environment to draw pictures in front of human users. Because of its interactivity and entertainment, robotic drawing systems have found applications in a wide range of scenarios, such as early childhood education (Hood, Lemaignan, and Dillenbourg 2015), psychological therapy (Cooney and Menezes 2018), and social entertainment (Jean-Pierre and Saïd 2012). Among these different systems, drawing human faces is one of the most engaging and entertaining tasks. During the past years, robotic face drawing has been extensively studied (Calinon, Epiney, and Billard 2005), with a focus on generating realistic portraits.

Although somewhat impressive, realistic drawings have a limited level of amusement for human users. In order to increase entertainment and engagement, in this paper, we propose a real-time robotic drawing system that converts input face images to cartoon avatars as a form of robotic art creation. Our system can capture and preserve facial features of the input face images and reproduce them in generated avatars so that the avatars have good likeness with the input faces. Such personalized art creation enriches the interactive experience between the user and the robot, while the optimized execution of the robotic drawing complements and improves the overall experience. Additionally, the whole system is trained without manual annotations.

In this paper, we propose *RoboCoDraw*, a collaborative robot-based, real-time drawing system. The core component of RoboCoDraw is *AvatarGAN*, a Generative Adversarial Network (GAN)-based image style transfer method for transferring real human faces to personalized cartoon avatars

[†] The first three authors contributed equally.
[‡] Corresponding author.

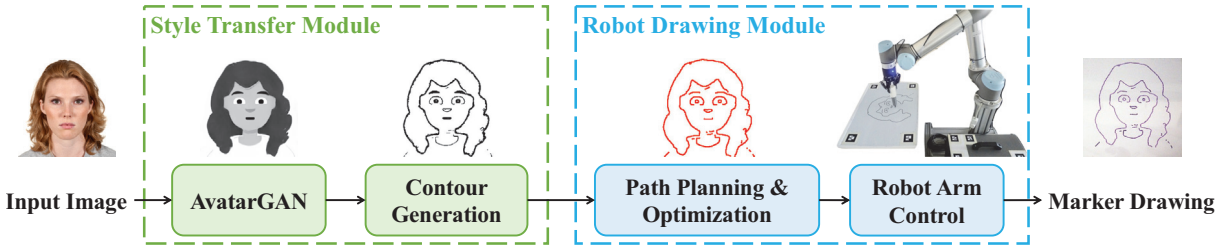

Figure 2: Pipeline of the RoboCoDraw System.

for robotic drawing. After generating the avatar with AvatarGAN, our system utilizes a line extraction algorithm to convert the avatar to a sketch. Subsequently, a Random-Key Genetic Algorithm (RKGA) (Bean 1994) based optimization algorithm is adopted to find a time-efficient robotic path for completing the avatar drawing.

Our main contributions are:

- a two-stream CycleGAN model named AvatarGAN, that maps human faces to cartoon avatars, while preserving facial features (such as haircut, face shape, eye shape, face color);
- a modularized, interactive robotic drawing system that performs faithful style translation and time-efficient face drawing; and
- a path optimization formulation for the robotic drawing problem and an RKGA-based optimization algorithm with two-level local improvement heuristics, so as to find time-efficient robot paths.

Relevant Work

The proposed *RoboCoDraw* system consists of two modules: a style transfer module and a robotic drawing path planning module. The related work to these two modules is discussed separately.

Style Transfer

A lot of research work has been carried out on style transfer with different approaches, such as neural style loss-based methods (Li et al. 2017) and Generative Adversarial Network (Goodfellow et al. 2014) (GAN)-based methods. The GAN-based methods are recent approaches that achieve high-quality style transfer using relatively small datasets. For example, CycleGAN (Zhu et al. 2017) performs image style translation using unpaired image datasets, which yields good texture mapping between objects with similar geometries, such as horse-to-zebra and orange-to-apple. However, it is not good at capturing facial features that are to be used to generate personalized avatars with a good likeness of input faces, as shown in our experiments.

CariGANs (Cao, Liao, and Yuan 2018) are another approach for photo-to-caricature translation, which consists of two networks, CariGeoGAN and CariStyGAN. CariGeoGAN performs geometrical transformation while CariStyGAN focuses on style appearance translation. Although they result in vivid caricatures, the caricatures come with a lot of details and are not well suited to our application scenario of quick drawing with robotic arms.

Moreover, it requires manually labeled facial landmarks, both on real face images and on caricatures, to train CariGeoGAN.

Path Planning for Robotic Drawing

The path planning module aims to reduce the time required for physical execution of the drawing. There are several approaches for planning robot drawing paths from a reference image, one of which is path generation based on replication of pixel shade and intensities in the reference image (Calinon, Epiney, and Billard 2005; Tresset and Fol Leymarie 2013). The Travelling Salesman Problem (TSP) line art combines such an approach with path optimization for efficient continuous line drawings (Kaplan and Bosch 2005). However, the final drawings obtained from replicating pixel shades lack the aesthetics and attractiveness of line art. Another work involving robotic drawing path planning attempted to reduce the number of output points and candidate trajectories (Lin, Chuang, and Mac 2009), but the approach was ad-hoc and lacked quantitative evaluation.

Additional research has been done on robotic path optimization and robotic task sequencing, as summarized in (Alatartsev, Stellmacher, and Ortmeier 2015), which could be applied to the robot drawing problem. Task sequencing, i.e., choosing an efficient sequence of tasks and the way the robot is moving between them, can be formulated as a TSP or its variants. It is often done manually or with offline path planning using a greedy search algorithm (Alatartsev, Stellmacher, and Ortmeier 2015) in industrial robotic applications. Other tour-searching algorithms that can be used for solving sequencing problems include Genetic Algorithms (Zacharia and Aspragathos 2005), Ant Colony optimization (Yang et al. 2008), and variety of local search heuristics (Lin and Kernighan 1973).

For our application, the robotic drawing problem is formulated as a Generalized Travelling Salesman Problem (GTSP), a variant of TSP. The proposed formulation models the path optimization problem more accurately and thus yields better results.

The RoboCoDraw System

In this section, we introduce the proposed *RoboCoDraw* system in detail. We first present the style transfer module, whose core component is *AvatarGAN*, a novel GAN-based

style transfer model for the real-face image to avatar translation; then we describe the robotic drawing module that performs planning and optimization of the drawing paths. Fig. 2 shows the pipeline of the *RoboCoDraw* system.

Style Transfer Module

The style transfer module consists of two components. One is *AvatarGAN*, which performs photo-to-avatar translation; the other is *Contour Generation* for extracting coherent contours from generated avatars. The two components work in sequence to produce stylized avatar sketches based on real face images.

Figure 3: Examples of real face to cartoon-style avatar translation using CycleGAN and AvatarGAN.

AvatarGAN The objective of AvatarGAN is to learn a mapping between the domain of real faces (X) and the domain of cartoon-style avatars (Y) while preserving the consistency in facial features such as haircut, face color, face shape, and eye shapes.

In order to make an approach broadly applicable to different avatar datasets, it is desired to be fully unsupervised. CycleGAN is an effective approach to learn the domain mapping in a fully unsupervised manner. However, the vanilla CycleGAN fails to preserve the facial features of real faces in the generated avatars (ref. Fig. 3), since it aims to learn the mapping between whole images in the two domains, without focusing on any specific areas within the images. To address this problem, we propose a two-stream CycleGAN model named AvatarGAN, in which an additional stream focuses on translating facial areas only. The two-stream architecture forces AvatarGAN to preserve important facial features when translating the whole image.

The structure of AvatarGAN is shown in Fig. 4. The training real face images are denoted as $\{x_{0,i}\}_{i=1}^{N} \subset X_0$, the training avatar images as $\{y_{0,j}\}_{j=1}^{M} \subset Y_0$, the cropped real facial area images as $\{x_{1,i}\}_{i=1}^{N} \subset X_1$, and the cropped avatar facial area images as $\{y_{1,j}\}_{j=1}^{M} \subset Y_1$.

Let the whole-image data distribution be $x \sim p_0(x)$ and $y \sim p_0(y)$, and the facial-area data distribution be $x \sim p_1(x)$ and $y \sim p_1(y)$. In the whole-image translation stream, the generator $G_{X_0Y_0}$ performs the $X_0 \rightarrow Y_0$ translation (from real face domain X_0 to avatar domain Y_0), and $G_{Y_0X_0}$ performs $Y_0 \rightarrow X_0$ translation. In the facial-area translation stream, the two generators $G_{X_1Y_1}$ and $G_{Y_1X_1}$ are similarly defined.

To make the translations in the two streams produce images in consistent styles, we make the weights shared by

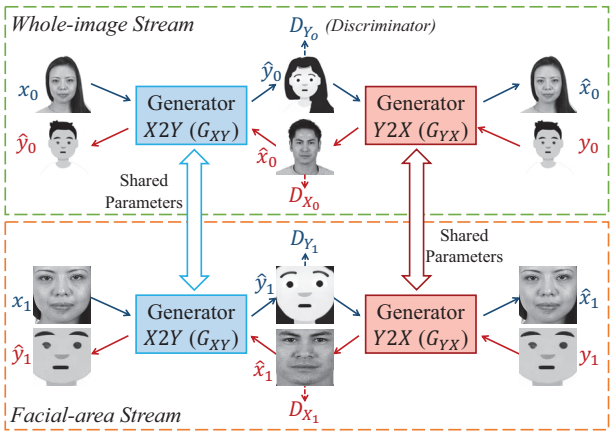

Figure 4: The structure of AvatarGAN, which consists of two translation streams. The whole-image stream performs ordinary style translation between real faces and avatars, while the facial-area stream focuses on learning facial feature mappings between the two domains.

the generators in the two streams, i.e., $G_{X_0Y_0} = G_{X_1Y_1}$, and $G_{Y_0X_0} = G_{Y_1X_1}$. For convenience, we drop the subscript and denote them as G_{XY} and G_{YX}. In addition, we introduce four adversarial discriminators D_{X_0}, D_{X_1}, D_{Y_0}, and D_{Y_1}. They aim to predict the probabilities that an image is from domain X_0, X_1, Y_0 and Y_1, respectively.

The objectives of AvatarGAN are: 1) minimize the adversarial loss ($\mathcal{L}_{adv}$) on both streams to align the distributions of generated images with the target domains; and 2) minimize the cycle consistency loss ($\mathcal{L}_{cycle}$) on both streams to enforces the learned mappings G_{XY} and G_{YX} to form inverse mappings with each other.

For adversarial loss, we only express the definition of the whole-image stream, and that on the facial-area stream is similar. Given the generator G_{XY} and its corresponding discriminator D_{Y_0}, the adversarial loss is defined as:

$$\mathcal{L}_{adv}(G_{XY}, D_{Y_0}, X_0, Y_0) = \mathbb{E}_{y \sim p_0(y)}[\log D_{Y_0}(y)] + \mathbb{E}_{x \sim p_0(x)}[\log(1 - D_{Y_0}(G_{XY}(x))]. \quad (1)$$

Similarly, we have $\mathcal{L}_{adv}(G_{YX}, D_{X_0}, X_0, Y_0)$ for the generator G_{YX} and its discriminator D_{X_0}. The adversarial losses on the facial-area stream $\mathcal{L}_{adv}(G_{XY}, D_{Y_1}, X_1, Y_1)$ and $\mathcal{L}_{adv}(G_{YX}, D_{X_1}, X_1, Y_1)$ for mappings G_{XY} and G_{YX} are defined in the same manner, respectively. The four adversarial losses are summed up as the overall adversarial loss ($\mathcal{L}_{adv}$).

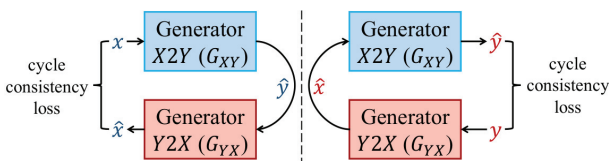

Figure 5: The cycle consistency loss, which enforces the learned mappings G_{XY} and G_{YX} to form inverse mappings.

The cycle consistency loss (Fig. 5) is defined to constrain the learned mapping functions to be cycle-consistent. In the

whole-image translation stream, for each image x from domain X_0, an image translation cycle is expected to convert x back to the original image, i.e. $x \rightarrow G_{XY}(x) \rightarrow G_{YX}(G_{XY}(x)) \approx x$; while for each image y from domain Y_0, a translation cycle is also expected to satisfies $y \rightarrow G_{YX}(y) \rightarrow G_{XY}(G_{YX}(y)) \approx y$. Thus, the cycle consistency loss in the whole-image stream is defined as:

$$\begin{aligned}\mathcal{L}_{cycle}(G_{XY}, G_{YX}, X_0, Y_0) = \mathbb{E}_{x\sim p_0(x)}\left[\|G_{YX}(G_{XY}(x)) - x\|_1\right] \\ + \mathbb{E}_{y\sim p_0(y)}\left[\|G_{XY}(G_{YX}(y)) - y\|_1\right].\end{aligned} \tag{2}$$

Similarly, the cycle consistency loss on the facial-area translation stream is denoted as $\mathcal{L}_{cycle}(G_{XY}, G_{YX}, X_1, Y_1)$. Then the overall cycle consistency loss is

$$\begin{aligned}\mathcal{L}_{cycle} = \alpha \cdot \mathcal{L}_{cycle}(G_{XY}, G_{YX}, X_0, Y_0) + \\ (1-\alpha) \cdot \mathcal{L}_{cycle}(G_{XY}, G_{YX}, X_1, Y_1),\end{aligned} \tag{3}$$

where α is a hyperparameter to balance the contributions between the whole-image and facial-area streams.

Finally, the full optimization objective is:

$$\mathcal{L} = \mathcal{L}_{adv} + \lambda \cdot \mathcal{L}_{cycle}, \tag{4}$$

where the hyperparameter λ assigns the relative importance of the cycle consistency loss within the whole loss.

Contour Generation In order to make it ready to be drawn by the robotic arm, the generated avatar is processed to generate contours with the Flow-based Difference-of-Gaussian (FDoG) filtering (Kang, Lee, and Chui 2009). During this process, the Edge Tangent Flow (ETF) is first constructed from the input image; then a DoG filter is applied to the local edge flow in the perpendicular directions. More formally, a one-dimension filter is first applied to each pixel along the line that is perpendicular to ETF, and then accumulate the individual filter responses along the ETF direction.

The FDoG filter only enhances coherent lines and reduces isolated tiny edges, thereby results in coherent and clean contours.

Robotic Drawing Module

The robotic drawing module plans an optimized path for the robotic arm to draw the avatars generated from the style transfer module. It first extracts the pixel-coordinates required for the robot drawing path, then optimizes the path by formulating the robotic drawing problem as a Generalized Travelling Salesman Problem (GTSP).

Ordered Pixel-Coordinates Extraction The image from the style transfer module is cleaned to increase clarity before subsequent processes. The lines of the binary image are further thinned to one-pixel in width for easy line tracing. The thinning process involves skeletonization, pruning, and line-ends extension, and is achieved by the hit-or-miss transform.

The thinned image is then split into several line segments from the junction points. The line segments are traced to obtain a sequence of pixel-coordinates to be visited in order when drawing the corresponding line segment. The ordered pixel-coordinates are subsequently stored and passed to the path optimization module to plan a low-cost path. Examples of outputs for each step in the ordered pixel-coordinates extraction process are shown in Figure 6.

Figure 6: The process of obtaining pixel-coordinates from the reference image. (a) Cleaned image; (b) Image skeleton; (c) Trimmed skeleton; (d) Image split at junctions (in red).

Robotic Drawing Path Optimization GTSP, a variant of TSP, is applied to formulate the path optimization problem. GTSP partitions the nodes into sets and aims to find the minimum cost tour that visits exactly one node in each set. Thus, both the visiting sequence of the sets as well as the particular nodes to visit need to be determined for solving GTSP (Noon and Bean 1993).

In the proposed GTSP, each line segment is represented as a set of two nodes, where each node represents a different drawing direction. The Random-Key Genetic Algorithm (RKGA) is then used to solve the GTSP. In RKGA, the random keys stored in the genes encode the drawing path, while a decoding process recovers the path for fitness evaluation. The encoding/decoding processes of RKGA generally help to map the feasible space to a well-shaped space and convert a constrained optimization problem to an unconstrained one (Bean 1994).

Encoding and decoding with random keys The drawing path is encoded as a list of random keys, where the index of each key corresponds to a specific line segment. The keys are real numbers, with the decimal part encoding the visit sequence (line segments corresponding to keys with smaller decimal value are visited earlier in the drawing sequence) and the integer part encoding the direction of drawing the line segment (1 represents that the pixel-coordinates traced from the line segment are visited in the forward sequence, while 0 represents that the pixel-coordinates are visited in reverse). A simplified example of an encoding/decoding path with random keys is shown in Fig. 7.

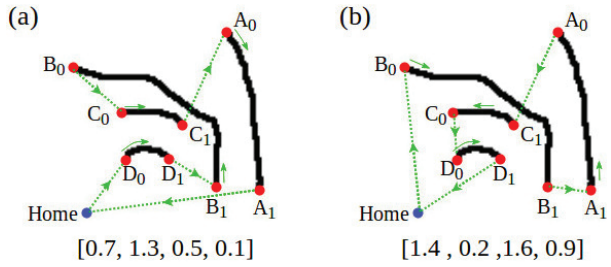

Figure 7: Examples of the path encoding of RKGA for robotic drawing applications. Line segments A, B, C and D corresponds to keys 1, 2, 3 and 4, respectively. There are two possible directions for traversing each segment ($0 \rightarrow 1$ or $1 \rightarrow 0$). The green arrows indicate the drawing path.

Fitness evaluation For robotic drawing, travel time increases with additional distance traversed while the marker is lifted from the whiteboard. Therefore, for a drawing path with K line segments in which $l_1, l_2, ..., l_K$ are the lines ordered in visit sequence, the main factors which contribute to the fitness value of the path are:

- the distances $d_{i,i+1}$ from the end of each line l_i to start of the next line l_{i+1} for $i = 1, ..., K-1$;
- the distances to and from the home position for the first and last lines in the tour ($d_{h,1}$ and $d_{K,h}$ respectively);
- the number of times the marker is lifted from whiteboard, n_{lift}, along with the additional cost $cost_{\text{lift}}$ incurred from each time the marker is lifted and placed.

Assuming constant velocity of the robot end-effector when the marker travels through free space, the increase in travel time is proportional to the increase in distance traveled through free space. Thus, we have

$$v_{\text{fitness}} = n_{\text{lift}} \times cost_{\text{lift}} + d_{h,1} + d_{K,h} + \sum_{i=1}^{K-1} d_{i,i+1}. \quad (5)$$

The goal of the RKGA is to minimize the fitness/cost value v_{fitness} of the robot drawing the path.

Genetic operators We employ three commonly-used genetic operators: reproduction, crossover, and mutation. For the reproduction operation, an elitist strategy is employed to clone the best r individuals directly into the next generation, to ensure the non-degradation of the best solution generated thus far by the algorithm. The remaining $n-r$ individuals for the next generation are produced from a pool of individuals selected via tournament selection from the parent population. Crossover and mutation, which are the second and third genetic operator respectively, are applied to the individuals in the selected pool with probabilities $p_{\text{crossover}}$ and p_{mutation}. Uniform crossover was used as the crossover strategy, while index shuffling and bit flip were used to mutate the of the decimal part and integer part of the key respectively.

Local improvement heuristics In this paper, we use a two-level local improvement method with RKGA to improve the optimization results. Level one improvement involves 2-opt and is applied to all new individuals. Level two involves the Lin-Kernighan (LK) heuristics (Lin and Kernighan 1973), and as the search with LK heuristics is relatively more expensive to perform, the new individuals are thresholded against an individual at v_{thres} percentile of the parent population after level one improvement. If the fitness of the resulting individual is better than the threshold value, the solution is deemed to have more potential and the additional level two improvement step (involving the more expensive the LK heuristic) is then applied to the individuals that passed the threshold.

Experiments and Discussion

Datasets

We conducted experiments using the images from Chicago Face Dataset (CFD) (Ma, Correll, and Wittenbrink 2015).

Figure 8: *Unpaired* examples of (a) CFD face images and (b) cartoon-style avatar images.

For the cartoon-style avatar image dataset, considering the drawing media of robot arm is marker on the whiteboard, the avatars should be suited to artistic composition with clean, bold lines. In order to meet such a requirement, we used the *Avataaars* library to randomly generate diverse cartoon avatar images as our avatar dataset. Examples of CFD image and generated avatar image are shown in Fig. 8. We randomly chose 1145 images from the CFD dataset and 852 images from generated avatar dataset to train AvatarGAN.

The training images from CFD dataset and generated by avatar dataset were *unpaired*. These images were firstly converted into grayscale, because the monochromatic sketches were performed by robot. Subsequently, the images were rescaled to 256×256. After that, the facial-area images are directly obtained by simple cropping from fixed position, and then re-scaled to 256×256.

Experiments on the Style Transfer Module

AvatarGAN Generation For AvatarGAN, we used the same architectures of generator and discriminator proposed by CycleGAN for a fair comparison. The generative networks include two convolution networks, nine residual blocks (He et al. 2016) and two deconvolution networks with instance normalization (Ulyanov, Vedaldi, and Lempitsky 2016). All four discriminators utilized 70×70 PatchGANs (Li and Wand 2016; Isola et al. 2017; Ledig et al. 2017), which aim to classify whether the 70×70 overlapping image patches are real or fake (Zhu et al. 2017). Moreover, we set $\alpha = 0.2$ to encourage the generator to focus more on learning facial features. The weight λ, which controls the relative importance of consistency loss, was set to 10.

The real face to avatar translation performance of AvatarGAN was evaluated visually first. Fig. 9 shows the transferred avatars by CycleGAN (b) and AvatarGAN (c), and the results from coherent contour generation (d). It is observed that the proposed AvatarGAN has two major advantages over CycleGAN in this application. First, AvatarGAN generated avatars with similar facial features as the input images, while CycleGAN seemed to significantly suffer from mode collapse, especially in the facial area. Second, the avatars generated by AvatarGAN are more diverse in the different parts of the cartoon structure.

It is also shown that CycleGAN fails to preserve important facial features and map them to avatars, since cycle consistency loss only enforces faithful mapping between *whole images*, and does not impose special constraints on any particular areas. In contrast, AvatarGAN preserves and maps facial features from the original faces effectively.

Figure 9: Face-to-avatar translation, where (a) are the real face images, (b) are the transferred results by CycleGAN, (c) are the transferred results by AvatarGAN, and (d) are the coherent contours of (c) generated by FDoG.

In addition to the above-mentioned advantages, it is worth noting that AvatarGAN can creatively generate new facial features that are not presented in the original dataset. For instance, there is only one face shape for all the avatars in the dataset (ref. Fig. 8), but the generated avatars have diverse face shapes, such as round, oval, diamond, and square, which are consistent with the original faces. As for another facial feature, the haircut, the resulted haircuts of avatars seem more distinctive and personalized, especially for the long hair. Although the diversity within the training dataset puts constraints on what images could be possibly generated, AvatarGAN still manages to bring in more personalized features and creativity to the avatar generation.

Contour Generation The contour generation sub-module generates avatar contours as shown in Fig. 9 (d). These contours capture important facial features from the generated avatars, such as the haircut, face shape, and facial expressions. In addition, lines in the contours are coherent and smooth, thereby the generated avatar sketches are more suitable to robotic drawing.

Evaluation of the Generalization To evaluate the generalization capability of AvatarGAN, we performed a face-to-avatar translation on additional face images from the CUHK Face Sketch Database (CUFS)

(Wang and Tang 2009). The AvatarGAN and CycleGAN models were trained on the CFD dataset and applied to CUFS dataset directly without fine-tune. The generated avatars by CycleGAN (b) and AvatarGAN (c) are shown in Fig. 10. On the CUFS images, the same observations as on Fig. 9 were made.

Table 1 compares the mean cycle consistency loss ($\mathcal{L}_{cycle}$) of CycleGAN and AvatarGAN. AvatarGAN significantly outperformed CycleGAN on the facial-area translation, as the newly introduced facial-area stream (ref. Fig. 4) enforces the generators of AvatarGAN to preserve facial features in the input image and translate them to the target domain. Surprisingly, CycleGAN slightly outperforms AvatarGAN on the whole-image translation, despite the fact that it generated

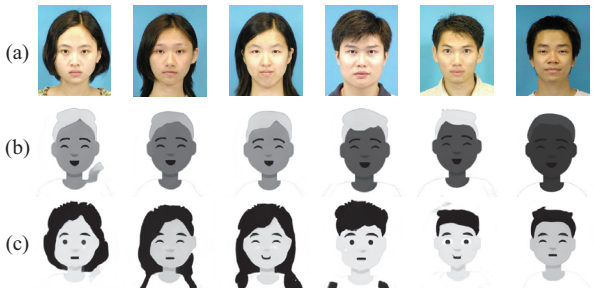

Figure 10: Face-to-avatar translation on the external CUFS dataset, where (a) are the real face images, (b) are the generated cartoon avatars by CycleGAN, and (c) are the translation results by AvatarGAN.

Table 1: Mean cycle consistency loss of CycleGAN and AvatarGAN on the test datasets. The column name indicates the input domain.

Method	Mean cycle consistency loss $\mathcal{L}_{cycle}$			
	Face	Avatar	Real facial area	Avatar facial area
CycleGAN	**0.0355**	**0.0167**	0.2620	0.3298
AvatarGAN	0.0367	0.0252	**0.0329**	**0.0252**

intermediate images of poor quality and low consistency with input images. Because CycleGAN sees the whole intermediate image when mapping it back to the input domain, it might exploit some subtle features in non-facial areas to recover the identity of the original image and use the image identity to facilitate the reconstruction of the input image. In other words, CycleGAN may take shortcuts to satisfy cycle consistency, without learning the true mapping of facial features between the two domains. The facial-area translation stream eliminates such shortcuts.

User Study To further validate the effectiveness of the proposed AvatarGAN, we conducted a user study from 10 candidates. In the survey, candidates were asked to match 10 random faces with the paired avatars generated by our system. The mean accuracy from the user study was 98%, and the standard deviation was 0.04.

Experiments on the Robotic Drawing Module

Pixel-Coordinates Extraction The pixel-coordinate extraction process was tested with a range of avatar image outputs from the style transfer module over 20 test runs. For each test run, the module reliably extracted and obtained the pixel-coordinates of line segments required to construct the robotic line drawing.

Path Optimization For additional testing of the path optimization algorithm, we grouped and placed multiple avatars into a single image to increase the complexity of each trial problem. Examples of the images used are shown in Fig. 12. We conducted the experiments across five different avatar groupings (average number of lines to optimize in each image is 74), with 10 trials conducted for each group of avatars.

Figure 11: Example of pixel-coordinates extraction. (a) The avatar sketch; (b) Simulated drawing using extracted pixel-coordinates; (c) Overlay of (a) and (b).

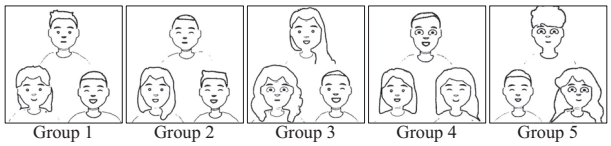

Figure 12: Examples of the avatar groupings used as inputs in the path optimization experiments.

The parameters used for RKGA are $N = 100$, $r = 3$, $p_{\text{crossover}} = 0.8$, $p_{\text{mutation}} = 0.5$, and $cost_{\text{lift}} = 30$. For the local search heuristic, the threshold percentile v_{thres} for the two-level improvement is set by $v_{\text{thres}} = \min(0.05 + 0.01c, 0.10)$, where c is number of consecutive generations with no improvement in the best solution. For the uniform crossover strategy employed, each key in the individual will originate from the first parent with a probability of 0.7, with an additional 0.5 chance for the tour encoded by the first parent to be reversed before the crossover operation. For mutation, the mutated individual will have indexes of the decimal part of the key shuffled with probability 0.05, followed by a bit-flip for integer part with probability 0.05.

The proposed optimization algorithm (RKGA w/ 2-opt, LK) was benchmarked against other commonly used methods. We used the greedy search algorithm as a baseline for our comparison, and calculated the percentage improvement of each algorithm's path fitness over the fitness of the greedy path. In particular, we achieved good path optimization results using the RKGA with 2-opt and LK heuristics. The resultant paths had significant improvement against the results from the greedy search method, with an average improvement in path fitness of 17.34%, as shown in Table 2.

Table 2: Improvements in path fitness value, v_{fitness}, of various optimization methods over the greedy benchmark (%).

	G1	G2	G3	G4	G5	Avg.
Greedy w/ 2-opt	16.4	20.2	5.5	8.8	9.8	12.1
Greedy w/ 2-opt, LK	16.9	22.5	14.7	11.0	10.3	15.1
RKGA w/ 2-opt	18.0	22.5	19.8	12.2	11.3	16.8
RKGA w/ 2-opt, LK	**19.1**	**22.9**	**20.4**	**12.4**	**11.9**	**17.3**

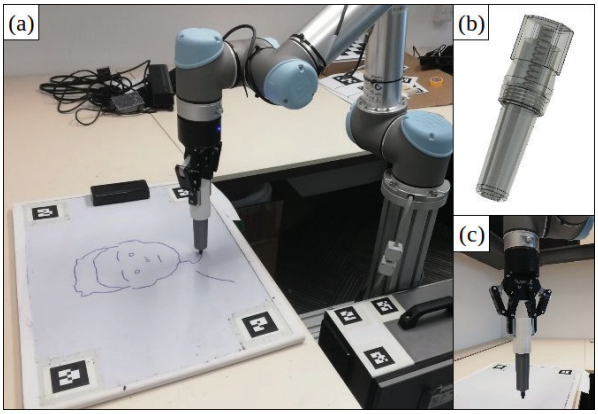

Figure 13: Integrated tests setup. (a) UR5 robot executing a drawing; (b) Tool holder design; (c) Robot end-effector.

Integrated Tests of RoboCoDraw System

We conducted 20 integrated trials with our RoboCoDraw system. Our drawing system was implemented with the UR5 robotic arm, with a Robotiq two-finger gripper attached as the end-effector. In addition, for more effective gripping and drawing with the marker, a customized spring-loaded marker holder was designed to compensate for small errors in the z-axis, ensuring that a constant amount of pressure is applied when drawing on slightly uneven surfaces. Fig. 13 shows our experiment setup for robotic drawing. Fig. 14 shows examples of the original photographs and the corresponding whiteboard drawings produced in our experiments.

In the integrated tests, the average time used for physical drawing was 43.2 seconds, while the other computational processes (image prepossessing, AvatarGAN translation, contour generation, etc.) took 9.9 seconds to complete.

CONCLUSIONS

In this paper, we proposed the RoboCoDraw system that facilitates the efficient creation and drawing of personalized avatar sketches on the robotic arm, given real human face images[1]. The proposed system consists of 1) a GAN-based style transfer module for personalized, cartoon avatar generation, and 2) an RKGA-based path planning and optimization module, for time-efficient robotic path generation. We compared the two proposed modules with existing methods in the style transfer and path optimization tasks, respectively. For the style transfer, our AvatarGAN generates more diversified cartoon avatars with much better likeness; For the path optimization, our method reduced 17.34% of the drawing time on average compared with the baseline. The RoboCoDraw system has great potential in public amusement and human-robot interactive entertainment applications.

Acknowledgements

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046).

[1]Code available at https://github.com/Psyche-mia/Avatar-GAN

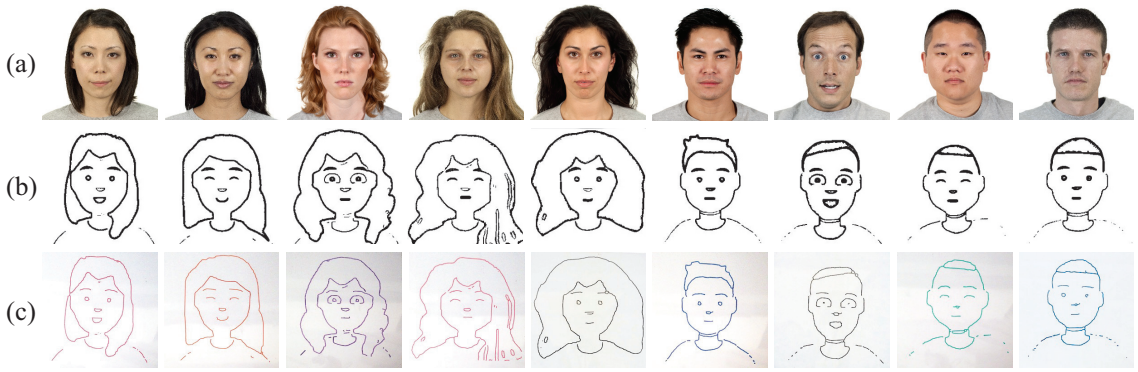

Figure 14: Results of the integrated tests, where (a) are the real face images, (b) are the corresponding cartoon avatars generated by AvatarGAN after coherent contour generation, and (c) are the marker-on-whiteboard drawings finished by the robotic arm.

References

Alatartsev, S.; Stellmacher, S.; and Ortmeier, F. 2015. Robotic task sequencing problem: A survey. *Journal of Intelligent and Robotic Systems* 80:279–298.

Bean, J. C. 1994. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal of Computing 6*.

Calinon, S.; Epiney, J.; and Billard, A. 2005. A humanoid robot drawing human portraits. In *IEEE-RAS International Conference on Humanoid Robots, 2005.*, 161–166. IEEE.

Cao, K.; Liao, J.; and Yuan, L. 2018. Carigans: Unpaired photo-to-caricature translation.

Cooney, M., and Menezes, M. 2018. Design for an art therapy robot: An explorative review of the theoretical foundations for engaging in emotional and creative painting with a robot. *Multimodal Technologies and Interaction* 2(3):52.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 770–778.

Hood, D.; Lemaignan, S.; and Dillenbourg, P. 2015. When children teach a robot to write: An autonomous teachable humanoid which uses simulated handwriting. In *ACM/IEEE International Conference on Human-Robot Interaction*, 83–90. ACM.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 5967–5976.

Jean-Pierre, G., and Saïd, Z. 2012. The artist robot: A robot drawing like a human artist. In *IEEE International Conference on Industrial Technology*, 486–491. IEEE.

Kang, H.; Lee, S.; and Chui, C. K. 2009. Flow-based image abstraction. *IEEE Transactions on Visualization and Computer Graphics* 15(1):62–76.

Kaplan, C. S., and Bosch, R. 2005. Tsp art. In Sarhangi, R., and Moody, R. V., eds., *Renaissance Banff: Mathematics, Music, Art, Culture*, 301–308. Southwestern College, Winfield, Kansas: Bridges Conference.

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; and Shi, W. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 105–114.

Li, C., and Wand, M. 2016. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*.

Li, S.; Xu, X.; Nie, L.; and Chua, T.-S. 2017. Laplacian-steered neural style transfer. In *ACM international conference on Multimedia*, 1716–1724. ACM.

Lin, S., and Kernighan, B. W. 1973. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.* 21(2):498–516.

Lin, C.-Y.; Chuang, L.-W.; and Mac, T. T. 2009. Human portrait generation system for robot arm drawing. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 1757–1762. IEEE.

Ma, D. S.; Correll, J.; and Wittenbrink, B. 2015. The chicago face database: A free stimulus set of faces and norming data. *Behavior research methods* 47 4:1122–1135.

Noon, C. E., and Bean, J. C. 1993. An efficient transformation of the generalized traveling salesman problem. *Information Systems and Operational Research* 31(1):39–44.

Song, D.; Lee, T.; and Kim, Y. J. 2018. Artistic pen drawing on an arbitrary surface using an impedance-controlled robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, 4085–4090. IEEE.

Tresset, P., and Fol Leymarie, F. 2013. Portrait drawing by paul the robot. *Computers and Graphics* 37:348–363.

Ulyanov, D.; Vedaldi, A.; and Lempitsky, V. S. 2016. Instance normalization: The missing ingredient for fast stylization. *CoRR* abs/1607.08022.

Wang, X., and Tang, X. 2009. Face photo-sketch synthesis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(11):1955–1967.

Yang, J.; Shi, X.; Marchese, M.; and Liang, Y. 2008. An ant colony optimization method for generalized tsp problem. *Progress in Natural Science* 18(11):1417 – 1422.

Zacharia, P., and Aspragathos, N. 2005. Optimal robot task scheduling based on genetic algorithms. *Robotics and Computer-Integrated Manufacturing* 21:67–79.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*.