This CVPR 2021 workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Deep Graphics Encoder for Real-Time Video Makeup Synthesis from Example

Robin Kips[1,2], Ruowei Jiang[3], Sileye Ba[1], Edmund Phung[3], Parham Aarabi[3], Pietro Gori[2]
Matthieu Perrot[1], Isabelle Bloch[4],

[1] L'Oréal Research and Innovation, France
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, France, [3] Modiface, Canada
[4] Sorbonne Université, CNRS, LIP6, Paris, France

{robin.kips, sileye.ba, matthieu.perrot}@loreal.com, {irene,edmund,parham}@modiface.com
pietro.gori@telecom-paris.fr, isabelle.bloch@sorbonne-universite.fr

Abstract

While makeup virtual-try-on is now widespread, parametrizing a computer graphics rendering engine for synthesizing images of a given cosmetics product remains a challenging task. In this paper, we introduce an inverse computer graphics method for automatic makeup synthesis from a reference image, by learning a model that maps an example portrait image with makeup to the space of rendering parameters. This method can be used by artists to automatically create realistic virtual cosmetics image samples, or by consumers, to virtually try-on a makeup extracted from their favorite reference image.

1. Introduction

Virtual-try-on technologies are now largely spread across online retail platforms and social media. In particular, for makeup, consumers are able to virtually try-on cosmetics in augmented reality before purchase. While creating virtual makeup for entertaining purposes is easy, parametrizing a rendering engine for synthesizing realistic images of a given cosmetics remains a tedious task, and requires expert knowledge in computer graphics. Furthermore, consumers are often prompted to select among a set of predefined makeup shades, but they cannot try makeup look from a reference inspirational images on social media.

In the past few years, the field of computer vision attempted to provide a solution to this problem through advances in the *makeup style transfer* task. This task consists in extracting a makeup style from a reference portrait image, and synthesizing it on the target image of a different person. State-of-the-art methods for this task [9, 10] are based on a similar principle. First, makeup attributes are extracted using a neural network and represented in a latent space. Then, this neural makeup representation is decoded and rendered on the source image using a generative model, such as GAN [4] or VAE [8].

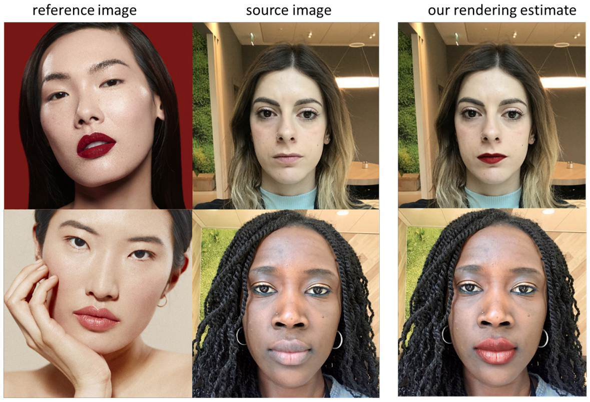

Figure 1. Examples of lipstick transfer from example images using our proposed method.

The use of generative model for addressing rendering-based problem, also denoted as *neural rendering* [14], allows producing realistic results but suffers from various limitations in practice. First, such approaches are currently impossible to use for real-time inference on portable devices. Furthermore, training generative models for video data is an emerging field, and even state-of-the-art models produce sequences of frames with noticeable temporal inconsistencies [2, 15]. Finally, generative methods are highly dependent on the training data distribution and might fail in the case of extreme examples, such as unusual makeup colors. These drawbacks make the use of current makeup transfer methods unusable in practice for consumer virtual try-on applications.

On the other hand, computer graphics methods offer symmetric advantages. Even though the most advanced rendering techniques require intensive computations, many

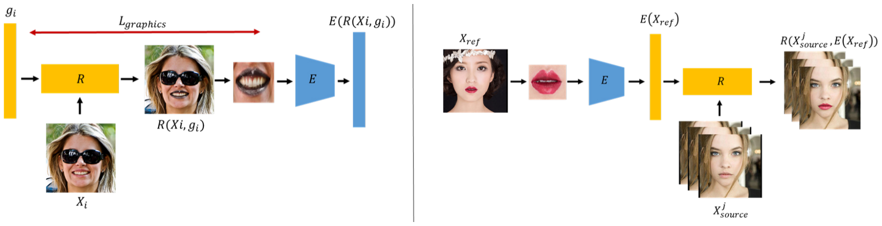

Figure 2. Left: training procedure of our model. We sample a graphics parameters vector g_i and render a corresponding image using a renderer R and a random source image X_i. Then, the inverse graphics encoder E is trained to map the image to the space of graphics parameters with minimum error. Right: inference pipeline. A reference image X_{ref} is passed to the inverse graphics encoder to estimate the corresponding makeup graphics parameters. Then this code can be used as input to the rendering engine, to render the reference makeup on videos in real-time. To facilitate training and increase the proportion of relevant pixels in the image, E is trained on crops of eyes and lips.

graphics-based methods can be used to produce realistic images in real-time, even on portable devices. As opposed to generative methods, graphics-based methods do not rely on training data, and can render videos without time inconsistency issues. However, they need to be carefully parametrized to render a given cosmetic product in a realistic manner. In practice, this is a tedious work that requires expert knowledge in computer graphics.

Recent works on *inverse rendering* introduced methods for estimating graphics attributes using differentiable rendering [7]. Such methods [3] propose to estimate parameters such as shape or BRDF by computing a forward rendering using an engine with differentiable operations, associated with gradient descent to optimize the graphics attributes with respect to one or multiple example images. However, this class of problem is often ill-posed, attempting to compute high-dimensional BRDF from RGB images. Furthermore, most real-time renderers are not differentiable in practice, and would require costly changes in computer graphics methods or implementation choices. To the best of our knowledge, there is no previous work in inverse computer graphics for makeup.

In this paper, we introduce a novel method based on deep inverse computer graphics for automatically extracting the makeup appearance from an example image, and render it in real-time, for a realistic virtual try-on on portable devices. Examples of our results are illustrated in Figure 1. Our contributions can be summarized as follows:

- We introduce a simple but powerful framework for learning an inverse graphics encoder network that learns to map an image into the parameter space of a rendering engine, as described in Figure 2. This is a more efficient and compact approach than inverse rendering, and does not require the use of a differentiable renderer.

- We demonstrate the effectiveness of this framework for the task of makeup transfer, outperforming state-of-the-art results, and achieving high resolution real-time inference on portable devices.

2. Method

2.1. Computer graphics makeup renderer

To achieve a realistic makeup rendering, we use a graphics-based system, that considers rendering parameters of color and texture features of any given cosmetics, such as described in [11]. Figure 3 illustrates the complete rendering pipeline for lipstick simulation.

Table 1. Descriptions of rendering parameters used in our graphics parameters vector representing the makeup material. The complete rendering engine includes a total of 17 parameters.

Description	Range
Makeup opacity	[0, 1]
R,G,B	$[\![0, 255]\!]$
Amount of gloss on the makeup	$[0, +\infty)$
Gloss Roughness	[0, 1]
Reflection intensity	[0, 1]

To obtain real-time inference, we first estimate a 3D lips mesh from estimated 3D facial landmarks. The rendering is then completed in two parts: 1) lips recoloring; 2) a two-step texture rendering based on environment reflection estimation and other controlled parameters. A similar rendering pipeline also applies to other makeup products. Table 1 describes the major rendering parameters used for representing a makeup product.

2.2. Inverse Graphics Encoder

We introduce a simple yet powerful framework to train an inverse graphics encoder that learns to project an example image to the parameter space of a rendering engine. In the case of makeup image synthesis, this allows us to automatically compute the renderer parametrization in order to synthesize a makeup similar to that of a reference example image.

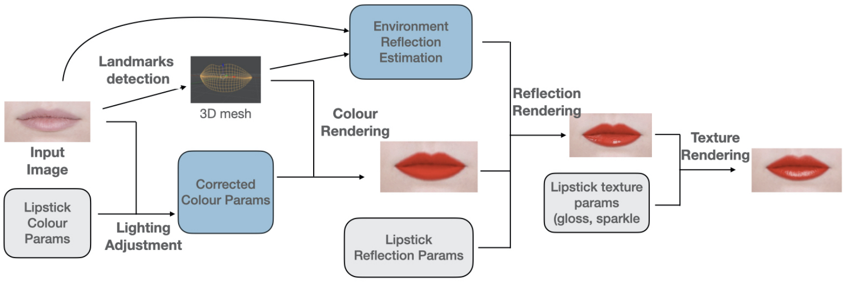

Figure 3. Our computer graphics rendering pipeline. While the makeup parameters are fixed prior to rendering, some parameters such as the lip mesh and the illuminant are estimated on each frame to render.

The training procedure of our framework is described in Figure 2. We denote by R the computer graphics rendering engine, taking as input a portrait image X and parametrized by g, the vector of parameters representing the makeup material that we name the *graphics parameters*. Each component of g is described in Table 1. Our objective is to train an encoder E, so that given an example makeup image X_{ref}, we can estimate the corresponding graphics parameters $\hat{g} = E(X_{ref})$ to render images with the same makeup appearance.

Since the renderer R is not differentiable, we do not use the *inverse rendering* approach and propose to learn E through optimization in the space of graphics parameters. This is a more compact problem than inverse rendering or material appearance extraction, and does not require a time-consuming gradient descent step for inference. Instead, we train a special-purpose machine learning model that learns an accurate solution for a given renderer and graphics parameters choice. Mathematically, we denote by g_i a randomly sampled graphics parameters vector, and X_i a random portrait image. Thus, our model E is trained to minimize the following objective function :

$$L_{graphics} = \frac{1}{n}\sum_{i=1}^{n} \|g_i - E(R(X_i, g_i))\|^2$$

Our approach does not depend on a training dataset of natural images, but only on a sampling of graphics parameters that we control entirely at training time. Therefore, in comparison to existing methods, our model is not sensitive to bias in available training datasets. Instead, we can select a graphics parameters distribution that samples the entire space of rendering parameters, which leads to better performance specially in cases of rare and extreme makeup examples. In our experiments, we used an EfficientNet B4 architecture [12] to represent E, replacing the classification layer by a dense ReLU layer of the same dimension as g. This light architecture is chosen in order to obtain a portable model with real-time inference. We construct two synthetic datasets by sampling $n = 15000$ graphics parameters vectors for eyes and lips makeup, and rendering them on random portrait images from the *ffhq* dataset [6], using the renderer described in Section 2.1. To obtain a realistic data distribution, the graphics parameters are sampled using a multivariate normal distribution fitted on real rendering parameters set by makeup experts to simulate real cosmetics. Furthermore, we also sample graphics parameters using a uniform distribution, in order to reinforce data diversity and improve our model performance on extreme makeup examples. Finally, our model is trained on crops of lips and eyes in order to increase the proportion of relevant pixels in the training image.

At the inference time, an example makeup image is passed to our inverse graphics encoder to estimate a corresponding graphics parameters vector. These parameters can then be used to render the extracted makeup attributes on any source video in real-time using R. Since the graphics parameters are fixed for each makeup, the inverse graphics encoder only needs to be run once per reference makeup image, and can then be used to render later video for any consumer. The inference pipeline is illustrated in Figure 2.

3. Experiments and Results

3.1. Qualitative experiments

In order to obtain a qualitative evaluation of our framework, we compare our approach to two state-of-the-art methods of makeup transfer: BeautyGAN [10] and CA-GAN [9]. We extract makeup from multiple reference images with various colors and glossiness levels, and synthesize makeup on the same source image, as illustrated in Figure 4. Our method produces more realistic results than existing makeup transfer methods, and is capable of accurately rendering the appearance of lipsticks and eye-shadow with various colors and textures, without any loss of image resolution. Furthermore, since our method is not dependent on the distribution of a training dataset, it largely outperforms other methods on extreme makeups such as blue lipstick or yellow eye-shadow, as shown in Figure 4.

In our problem formulation, the eye-shadow application zone and intensity are not part of the estimated graphics pa-

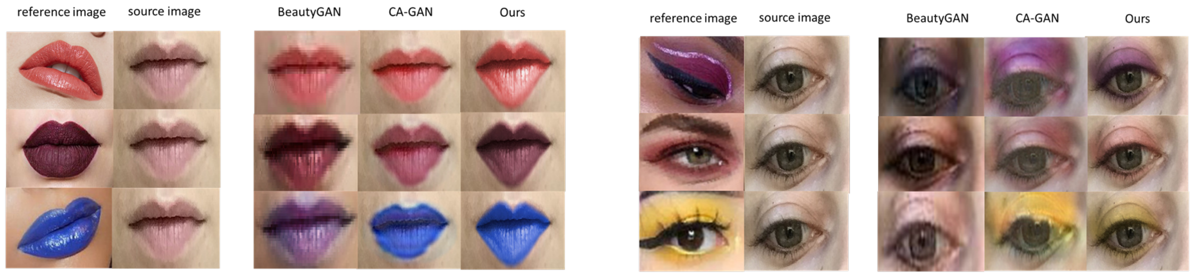

Figure 4. Qualitative comparison on lipstick and eye-shadow synthesis against state of the art makeup transfer methods. Our method is capable of reproducing realistic rendering in high resolution for makeup with various colors and textures. The eye-shadow application zone and intensity are not part of the estimated graphics parameters, but set by the user at rendering time according to their preferences.

Table 2. Quantitative evaluation of the makeup transfer performance using a dataset of groundtruth triplet images.

Model	L1	1-MSSIM [16]	LPIPS [17]
BeautyGAN [10]	0.123	0.371	0.093
CA-GAN [9]	0.085	0.304	0.077
Ours	**0.083**	**0.283**	**0.060**

rameters, but set by the user at rendering time according to their preferences. This choice allows for an increased user control on the makeup style, at the cost of not reproducing automatically the entire eye makeup style of the reference image. Finally, to give the reader more insight about our model, we provide example videos as supplementary materials, as well as an interactive demo application for lipstick transfer.

3.2. Quantitative experiments

In order to compare our results with existing methods, we reproduce the quantitative evaluation of makeup transfer performance on lipstick, introduced in [9]. More precisely, we use the dataset provided by the authors with 300 triplets of reference portraits with lipstick, source portraits without makeup, and associated ground-truth images of the same person with the reference lipstick. We compute the accuracy of our model over various perceptual metrics and report the results in Table 2. These results confirm that our framework outperforms the existing makeup transfer methods.

3.3. Inference Speed

An important limitation of generative-based methods for makeup transfer is their inference speed with limited resources, especially on mobile platforms. For instance, the StarGAN [1] architecture used in CA-GAN takes 18 seconds to synthesize a 256x256 image on an Ipad Pro with a A10X CPU. Even though some optimization is possible using GPU or neural inference special-purpose chips, this makes the use of generative models currently prohibitive for real-time consumer applications.

In comparison, our method uses a neural network not on every frame of the source video, but only once to compute the graphics parameters vector sent to the renderer. Furthermore, our graphics encoder is based on EfficientNet-lite-4 [12], an architecture adapted to mobile inference, reportedly reaching an inference time of 30ms per image on a Pixel 4 CPU [5]. Thus, the additional computational time introduced by our graphics encoder can be considered negligible when generating a video. To illustrate the inference time of our video solution, we profile our computer graphics pipeline on different mobile devices. We use the landmarks detection model described in [11] and convert it to NCNN [13] to make it runnable on mobile platforms. To get accurate profiling results, we skip the first 100 frames and average the results of the next 500 frames for each device. As shown in Table 3, our system is able to achieve excellent performance even on old devices such as Galaxy S7.

Table 3. Profiling of our graphics rendering pipeline on 3 different devices. Since our graphics encoder is only used once before the rendering and not at each frame, we consider its time is negligible in the video synthesis.

Device	Detection	Rendering	Display Time
Galaxy S21	11.97ms	14.95ms	2.12ms
Pixel 4	18.32ms	19.54ms	2.23ms
Galaxy S7	27.89ms	58.55ms	10.68ms

4. Conclusion

We introduced a method for learning a special-purpose inverse graphics encoder that maps an example image to the space of a renderer parameters, even in the absence of a differentiable renderer. We showed that our framework can be applied to the task of makeup transfer, allowing non-expert users to automatically parametrize a renderer to reproduce an example makeup. In the future, we intend to improve our framework with a larger definition of the graphics parameters, such as including the estimation of the eye makeup application region.

References

[1] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018. 4

[2] Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *TOG*, 2020. 1

[3] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *TOG*, 2019. 2

[4] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 1

[5] Google. Tensorflow blog : higher accuracy on vision models with efficientnet-lite. `https://blog.tensorflow.org/2020/03/higher-accuracy-on-vision-models-with-efficientnet-lite.html`, Last accessed on 2021-04-15. 4

[6] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 3

[7] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020. 2

[8] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *ICLR*, 2014. 1

[9] R. Kips, M. Perrot, P. Gori, and I. Bloch. CA-GAN: Weakly supervised color aware GAN for controllable makeup transfer. In *ECCV Workshop AIM*, 2020. 1, 3, 4

[10] T. Li, R. Qian, C. Dong, S. Liu, Q. Yan, W. Zhu, and L. Lin. BeautyGAN: Instance-level facial makeup transfer with deep generative adversarial network. In *ACM Multimedia*, 2018. 1, 3, 4

[11] T. Li, Z. Yu, E. Phung, B. Duke, I. Kezele, and P. Aarabi. Lightweight real-time makeup try-on in mobile browsers with tiny CNN models for facial tracking. *CVPR Workshop on CV for AR/VR*, 2019. 2, 4

[12] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 3, 4

[13] Tencent. NCNN, high-performance neural network inference framework optimized for the mobile platform. `https://github.com/Tencent/ncnn`, 2018. 4

[14] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, 2020. 1

[15] H. Thimonier, J. Despois, R. Kips, and Perrot. Learning long term style preserving blind video temporal consistency. In *ICME*, 2021. 1

[16] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, 2003. 4

[17] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 4