

Report Digital Image Processing: AutoToon

Automatic Geometric Warping for Face Cartoon Generation

Nguyen Thi Hoang Linh
Hanoi University of Science and Technology (HUST)
linh.nth224324@sis.hust.edu.vn

Le Ngoc Huong
Hanoi University of Science and Technology (HUST)
huong.ln224315@sis.hust.edu.vn

Abstract

Caricatures highlight unique facial features through artistic exaggeration, a task traditionally requiring skilled artists. While deep learning has advanced stylization, accurate face warping remains challenging. AutoToon, a supervised model that generates high-quality geometric warps, decoupled from stylization for flexible combinations. Using an SENet and spatial transformer module, it learns from artist-created warping fields and applies tailored losses to ensure detail and visual appeal. User studies show AutoToon outperforms existing methods, effectively enhancing facial features. We also present the AutoToon dataset, offering paired portraits and artist warps to support further research in caricature and cartoon generation.

1. Introduction/ Motivaiton

Caricature is a unique form of artistic representation that amplifies the most distinctive features of a person's face, making it easier for viewers to recognize them. Traditionally, creating caricatures is a skill reserved for artists who can expertly identify and exaggerate these traits. However, with advancements in computer vision, there's a growing interest in automating this process.

In this context, our goal is to use a system that takes a standard portrait photo as input and produces an exaggerated cartoon version as output. While recent deep learning methods have shown promise in stylizing images, the challenge lies in performing effective geometric warping to enhance key facial features without losing important details.

The task is further complicated by the need to maintain a clear separation between the warping and stylization processes, allowing for greater flexibility in combining different styles. Existing methods often struggle with this, lead-

ing to caricatures that either lack precision in exaggeration or fail to retain the unique characteristics of the original photo. Our work addresses these challenges by introducing AutoToon, a supervised learning framework that automates high-quality face warping for cartoon generation, setting the stage for more effective caricature creation.

Our contributions are as follows

2. Related Work

2.1. Learning Warping

Spatial transformations have been widely studied, with methods like spatial transformer networks (STNs) estimating global transformations and flow-based approaches enabling dense, detailed manipulations. Techniques like DeepWarp [1] refined this for tasks such as gaze adjustment, while Zhao et al. [2] used dense flow to correct portrait distortions. Other works, like Cole et al. [3], applied landmark-based warping to maintain identity, and Zhang et al. [4] incorporated smoothness and alignment for image stitching. AutoToon builds on these advances, integrating dense flow estimation and differentiable warping from STNs to predict warping fields for facial exaggeration.

2.2. Caricature Generation

Caricature generation amplifies unique facial features, initially relying on rule-based methods and later incorporating data-driven approaches using annotated datasets like Web-Caricature. Recent works, such as CariGAN [5] and WarpGAN [6], employed GANs to combine style and warping but often lacked consistency due to unpaired training data and diverse artist styles. AutoToon addresses these challenges with a supervised approach, leveraging paired data and dense warping fields to produce detailed, high-quality exaggerations. By focusing on geometric warping, it ensures flexibility and disentangles style from the exaggera-

tion process.

3. Methodology

3.1. Dataset

101 frontal-facing portrait images will be collected from Flickr, representing diverse demographics in age, gender, race, and face shapes. These images will be styled like realistic artist’s caricatures using Adobe Photoshop. The dataset includes 101 image pairs, with 90 for training and 11 for validation. The test set, collected separately from public platforms, contains images without ground-truth labels.

As part of the dataset, we provide estimated artist warping fields $F_{32} \in \mathbb{R}^{32 \times 32 \times 2}$, which are bilinearly upsampled to a size of $H \times W \times 2$ to match each artist’s caricature. The choice of a 32×32 spatial resolution for the warping fields is discussed in the following section. To generate these fields, we applied gradient descent optimization on the warping field for each Xtoon using L_1 loss through the differentiable Warping Module, aiming to create artist warping fields that closely resemble each Xtoon. The optimization problem we solved is as follows:

$$\arg \min_{F_{32}} \|X_{\text{toon}} - \text{Warp}(X_{\text{in}}, \text{Upsample}(F_{32}))\|_1$$

3.2. Model Architecture

The model consists of two main components: the Perceiver Network and the Warping Module. The Perceiver Network is built on a modified Squeeze-and-Excitation Network (SENet50). The network uses weights pretrained on the VGGFace2 dataset. To adapt SENet50 for AutoToon, the architecture is truncated up to the second bottleneck block, which is followed by an adaptive average pooling layer. The Perceiver Network processes the input image X_{in} and outputs warping field F_{32} , which is then upsampled via bilinear upsampling to obtain F . The Warping Module then applies the warping field F to X_{in} to obtain X_{toon} . During inference, the warping field can be scaled by a factor α to adjust the intensity of the exaggeration effect, allowing for customizable cartoon styles.

The decision to upsample a 32×32 warping field was based on two main considerations. First, upsampling naturally smooths out the warping effect, resulting in cartoons with softer, more visually appealing transitions. Second, sticking to resolutions that are powers of 2, a 64×64 warping field was deemed too detailed, while a 16×16 field produced cartoons that lacked sufficient exaggeration. Additional details can be found in the supplementary materials.

3.3. Loss Functions

There are three loss functions used to train AutoToon: the reconstruction loss, artist warping loss, and smoothness regularization loss.

- The reconstruction loss L_{recon} ensures that the generated cartoon X_{toon} closely resembles the artist’s cartoon X_{toon} .
- The artist warping loss L_{warp} supervises the warping field to match the artist’s style.
- The smoothness regularization loss L_{reg} promotes smooth transitions within the warping field to avoid abrupt changes that could make the cartoon appear unnatural.

Combining all three losses, the final loss function used for training the AutoToon model is:

$$L_{\text{autotoon}} = \lambda_1 L_{\text{recon}} + \lambda_2 L_{\text{warp}} + \lambda_3 L_{\text{reg}}$$

($\lambda_1, \lambda_2, \lambda_3$: weighting factors)

4. New Ideas

We want to make something applicable based on the AutoToon method. We added a new script called “Cartoon Webcam Feed”. This is a real-time application that processes live video feed from a webcam and applies a cartoon-style transformation to each frame using a deep learning model.

The features include: Real-Time Processing that captures and processes live frames from the webcam using a neural network model to apply a cartoon effect; Deep Learning-Based Cartoonization and User Interaction is to display the transformed video feed in a separate window and allow the user to quit the application by pressing the ‘q’ key.

On the technical side, there are also some notable details. The system captures video frames from the default webcam using OpenCV’s `cv2.VideoCapture` function. Each frame is resized to 256×256 pixels, normalized to $[0, 1]$, and converted from BGR to RGB format for compatibility with PyTorch. The tensor undergoes a forward pass through the model to generate cartoonized output. Finally, the cartoonized tensor is converted back to a NumPy array, denormalized, and resized to its original dimensions for display. The cartoonized frames are displayed in a window using OpenCV’s `cv2.imshow` function, allowing users to view the live cartoonized feed, which updates in real time. To quit the application, press the ‘q’ key. The application also handles errors such as failure to access the webcam or read frames.

The core functions of the system include `cartoonize_frame_with_autotoon(frame)`, which takes a single frame as input, pre-processes it by resizing, normalizing, and converting it to a tensor, passes the frame through the model to get the cartoonized output, and post-processes the output to convert it back into an image format suitable for display. Another function, `cartoon_webcam_feed()`, initializes the webcam feed, captures frames in a loop, processes them with `cartoonize_frame_with_autotoon`, and displays the cartoonized output while handling quitting when the

user presses the `q` key. The system requirements include a webcam (built-in or external), Python 3.7 or higher, and libraries such as `torch` (PyTorch), `opencv-python`, and `numpy`.

5. Experimental Results Discussion





5.1. Training details

We employ the Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and apply learning rate decay at a rate of 0.95. The batch size is set to 16, with each mini-batch consisting of a randomly chosen and aligned input-cartoon pair, along with the corresponding artist warp. For data augmentation, two techniques are used: random horizontal flipping and color jittering. The color jittering includes variations in brightness, contrast, and saturation (each sampled uniformly from the range $[0.9, 1.1]$), as well as hue jitter (uniformly sampled from $[-0.05, 0.05]$, as specified by PyTorch’s color jitter API). We empirically set $\lambda_1 = 1$, $\lambda_2 = 0.7$, and $\lambda_3 = 1e-6$.

6. Testing model

6.1. Good test results



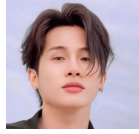


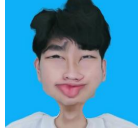


Table 1. Examples of good tests

Original	AutoToon
	
	

6.2. Bad test results

Images with solid, high-contrast backgrounds or those not normalized can show unexpected exaggerations when processed by the AutoToon model. Pre-processing steps like background normalization can help achieve consistent and visually pleasing results.

Table 2. Examples of bad tests

Original	AutoToon	Problem
		Very few exaggeration
		Half exaggeration
		Unusual exaggeration
		can't exaggerate all faces

When a person’s face is partially obscured, tilted, or involves two individuals with one person’s inner eyes not recognized, the AutoToon model selectively exaggerates only the recognized parts of the face. This results in a distinctive half-exaggerated aesthetic. Pre-processing steps to enhance face recognition or handle obscured or tilted faces can help achieve a more comprehensive exaggeration of facial features.

The attempts to exaggerate these images failed during preprocessing as the landmark detectors couldn’t recognize any faces. This hindered normalization, highlighting the need for adjustments. Exploring alternative landmark detection algorithms or fine-tuning parameters may enhance face recognition in challenging conditions, improving the preprocessing workflow.

7. Evaluation

7.1. MSE (Mean Squared Error): 4.73

The formula for calculating Mean Squared Error (MSE) is:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (X_i - Y_i)^2$$

where:

- N is the number of pixels in the image.
- X_i is the pixel value at position i in the ground truth image.

- Y_i is the pixel value at position i in the output image.

MSE calculates the average squared error between the pixels of the ground truth image and the output image.

A value of 4.73 is quite small relative to the range of pixel values. This suggests that overall, the squared errors are not substantial, meaning there are no extreme discrepancies between the ground truth and the output images. It could imply that the model has a good overall performance but still exhibits some localized errors.

7.2. SSIM (Structural Similarity Index): 0.982

The formula for calculating the Structural Similarity Index (SSIM) is:

$$\text{SSIM}(x, y) = \frac{(2xy + C_1)(2\sigma_{xy} + C_2)}{(x^2 + y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where:

- x and y are the mean values of images x and y .
- x^2 and y^2 are the variances of images x and y .
- σ_{xy} is the covariance between images x and y .
- C_1 and C_2 are constants used to stabilize the formula when dividing by small numbers.

SSIM evaluates the structural similarity between two images, with values closer to 1 indicating higher similarity, meaning the output image is more similar to the ground truth image.

An SSIM of 0.982 is excellent. It indicates that the model is able to preserve the structural features of the images very well, with minimal perceptual differences. This is a positive result, suggesting that the model maintains high image quality and structural fidelity.

7.3. PSNR (Peak Signal-to-Noise Ratio): 40.58 dB

The formula for calculating Peak Signal-to-Noise Ratio (PSNR) is:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right)$$

where:

- MAX_I^2 is the maximum value of a pixel in the image.
- MSE is the Mean Squared Error between the ground truth image and the output image.

A PSNR of 40.58 dB is very good. This suggests that the output image is of high quality and very similar to the ground truth, with minimal distortion or noise.

8. Conclusion

In this project, we successfully used the AutoToon system as the first supervised deep learning method for cartoonization, and achieved superior warping quality while preserving facial detail. By disentangling warping from stylization,

AutoToon offers flexibility and detail, outperforming state-of-the-art methods in exaggerating facial features. With the dataset of 101 image pairs, alongside the comparison with the prior art with respect to a lot of key criteria, AutoToon is highlighted to be a potential project in addressing real-world applications.

However, while this system shows good results in caricature exaggeration, it still faces some challenges. There are instances where the outputs take from half to very few exaggerations, even though we can not find any significant difference between input image and output one. Additionally, changing some indicators is also a cause of the issue of unusual exaggeration, reducing the quality of the model.

Furthermore, the future work can explore refining warping smoothness, improving identity preservation, and adapting to diverse artist styles through few-shot learning.

9. References

1. Y. Ganin, D. Kononenko, D. Sungatullina, and V. Lempitsky. Deepwarp: Photorealistic image resynthesis for gaze manipulation. In European conference on computer vision, pages 311–326. Springer, 2016. (<https://arxiv.org/pdf/1607.07215>)
2. Y. Zhao, Z. Huang, T. Li, W. Chen, C. LeGendre, X. Ren, J. Xing, A. Shapiro, and H. Li. Learning perspective undistortion of portraits. arXiv preprint arXiv:1905.07515, 2019. (https://openaccess.thecvf.com/content_ICCV_2019/papers/Zhao_Learning_Perspective_Undistortion_of_Portraits_ICCV_2019_paper.pdf)
3. F. Cole, D. Belanger, D. Krishnan, A. Sarna, I. Mosseri, and W. T. Freeman. Synthesizing normalized faces from facial identity features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3703–3712, 2017 (<https://arxiv.org/pdf/1701.04851>)
4. F. Zhang and F. Liu. Parallax-tolerant image stitching. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '14, pages 3262–3269, Washington, DC, USA, 2014. IEEE Computer Society. (https://openaccess.thecvf.com/content_cvpr_2014/papers/Zhang_Parallax-tolerant_Image_Stitching_2014_CVPR_paper.pdf)
5. K. Cao, J. Liao, and L. Yuan. Carigans: Unpaired photo-to-caricature translation, 2018. (<https://arxiv.org/pdf/1811.00222>)
6. D. D. Shi, Yichun and A. K. Jain. WarpGAN: Automatic caricature generation. In Conference on Computer Vision and Pattern Recognition, 2019. (<https://arxiv.org/pdf/1811.10100>)