Computer Vision 2 - Assignment 3
2D-to-3D for FaceSwapping

TAs: Partha Das, Kien Nguyen, Minh Ngo

Monday 3$^{\text{rd}}$ May, 2021

Deadline: 28 May, 2021, 23:59:59.000

1 Intro / Background

Generating images or videos by manipulating facial attributes (i.e. face reenactment and swapping) has gained a lot of attention in recent years due to their broad range of computer vision and multimedia applications such as video dubbing, gaze correction, actor capturing, and virtual avatar creation. Imagine, old Mark Hamill could act as a young Luke Skywalker, or a stuntman acting as Daniel Craig. In this assignment, you will implement your face swapping pipeline with the help of 2D-to-3D reconstruction.

The task of face reenactment is to change the attribute of a face image (i.e. pose, expression or light), while *face swapping* is focused on changing face *identity* of an image while keeping other attributes (pose, expression and background) the same. Face identity can be defined as a set of attributes that identify a person. It can range from skin/eye colors to face geometry and expression. Ignoring geometrical changes and lighting conditions, one can approximate facial identity purely by the texture of the face region.

1.1 2D-to-3D

So far, in assignments 1 and 2, you have experimented with algorithms such as Iterative Closest Point (ICP) and Structure-from-Motion. If 3D data is available from depth sensors / LIDAR, information from multiple frames can be combined using ICP. When only images from a regular camera are available, Structure-from-Motion can be used to extract 3D key points. In both cases, however, multiple image frames from different viewpoints are required to reconstruct a 3D scene. In addition, an observed object is assumed to be rigid.

Can we still do 3D reconstruction of a non-rigid object given a single image frame? An answer is yes, given certain prior knowledge about an object. For example, knowing that an observed object is a ball we will know that its 3D model should be ellipsoidal. If an object of observation is a human face, it should have two eyes, 1 mouth, 1 nose. They should position in a certain way and have some constraints in shape.

Object Prior. If we have a subset of 3D scans of objects of a single class, align them based on key attributes and normalize in scale, we can build a PCA model and represent any 3D geometry of a new unobserved object of the same class as a linear combination of its principal components (Please check **?**Blanz and Vetter [1999] for more details).

$$\mathbf{G} = \boldsymbol{\mu} + \alpha_1 \sigma_1 E_1 + .. + \alpha_n \sigma_n E_n \tag{1}$$

Assuming our PCA model to consist of n principal components, μ is a mean of our subset, σ_i and E_i are i-th standard deviation and principal component respectively. **G** is a point cloud of M points (defined by the precision of 3D models), μ, E_i are of size $M \times 3$. By changing parameters α_i different 3D geometry can be sampled. Closer α_i to 0, closer **G** is to the mean geometry $\boldsymbol{\mu}$. Larger α_i, differ **G** is to the mean geometry.

One can interpret Eq. 4 as a multi-modal normal distribution, with α_i further away from 0 representing 3D geometry less likely to belong to a class. A similar concept you might face before with Fourier decomposition, where one can approximate a wave as a linear combination of sinusoidal functions (Fig. 1).

Object Movement. Now, a 3D object can be defined using a prior model (Eq. 4) and input parameters α_i. Like from the ICP assignment, a 3D point cloud movement in the scene can be modeled using rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^3$.

$$\hat{\mathbf{G}} = \mathbf{G}\mathbf{R}^T + \mathbf{t} \tag{2}$$

Camera Projection. How a 3D object is perceived from a viewpoint can be modelled using a pinhole camera model $\boldsymbol{\Pi} \in \mathbb{R}^{\mathbf{4}\times\mathbf{4}}$ into a camera plane.

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{d} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{V} \end{bmatrix} \times \begin{bmatrix} \mathbf{P} \end{bmatrix}}_{\boldsymbol{\Pi}} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3}$$

where $\boldsymbol{V}$ is a viewport matrix (`http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/projection/viewport_transformation.html`) and $\boldsymbol{P}$ is a perspective projection matrix (`https://bit.ly/300gYmf`). Please check provided links for more details. U,V projection (corresponding 2D pixel coordinate of each 3D point) can be obtained by dividing $\hat{x}$ and $\hat{y}$ by a homogeneous coordinate.

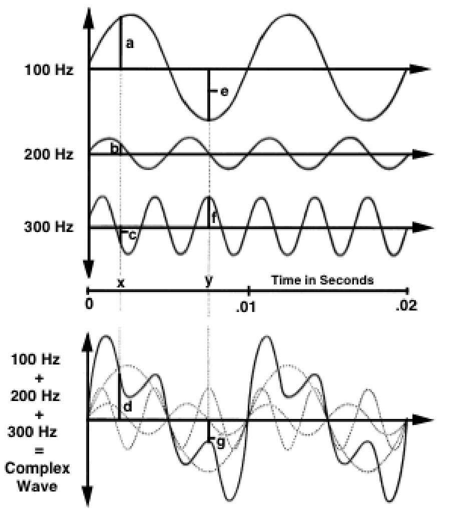

Figure 1: Representing a wave as a linear combination of sinusoidal functions.

Energy Minimization. Given a parametric 2D-to-3D model $f(\alpha, R, t, ...)$, latent parameters α, R, t.. can be found by energy minimization procedure. An idea is to determine ground truth for certain outputs of the model and define a loss (energy) function which tells how far the current estimation is from the ground truth. If the loss function and a parametric model are differentiable (with all partial derivatives defined), latent parameters can be found using the gradient descent procedure.

Deep Learning-based 2D-to-3D reconstruction. Given a 2D-to-3D image formation model and energy minimization procedure, a Deep Neural network can be trained to predict latent parameters of the model Deng et al. [2019], Tewari et al. [2017]. The parametric model becomes $f(DNN(image))$ instead of $f(\alpha, R, t, ...)$ with DNN parameters, which can be trained using gradient descent.

1.2 Face Swapping

Given two face images of different identities, one can produce face swapped results using 2D-to-3D reconstruction knowing U,V projection coordinates by rendering the texture of 1 face on top of a 3D geometry of another face.

1.3 Generative Adversarial Networks

Implicit Generative Model. Without the prior information of data distribution, the implicit generative model needs to model the distribution of a given set of data (e.g., images, audio, or text). In this course, we focus on deep

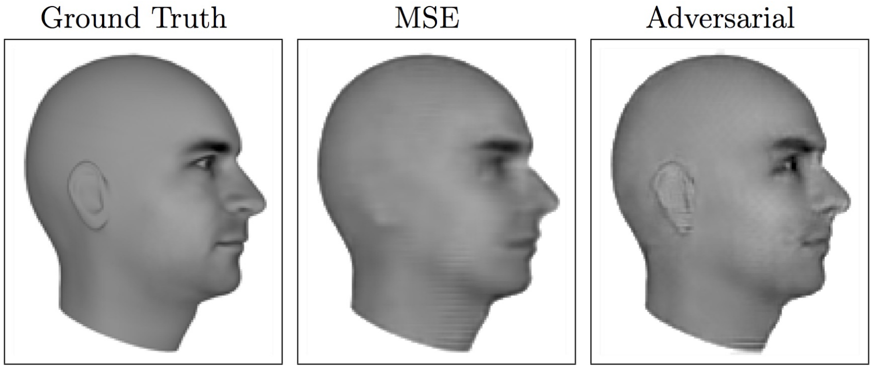

Figure 2: The comparison of the generation capability between GAN and VAE

learning-based generative models which are the current state-of-the-art method in learning distribution of complex datasets. There are four kinds of deep generative models in popular use today:

- Generative adversarial networks (the topic of this course)
- Autoregressive models
- Variational autoencoders
- Reversible architectures

There out of these four methods are trained with maximum likelihood which is a most popular loss across many deep learning applications. However, the maximum likelihood training loss has shown a limitation in generating blurry images. The solution of **Generative Adversarial Networks (GANs)** is a different way to describe the problem in which it treats the generation process as a minimax game. In specific, we want a generator to produce samples that are indistinguishable from the real data, as judged by a discriminator to decide real or fake samples. GAN is widely used due to its power of generating high-quality samples 2.

Generative Adversarial Networks. GANs contain two main components: generator and discriminator. The generator is trained to generate the samples from the training distribution while the discriminator is to classify whether the generated samples are from the training set or produced by the generator. We evaluate the generator based on the discriminator's inability to classify the generated samples.

- The generator network **G** is to generate realistic samples.

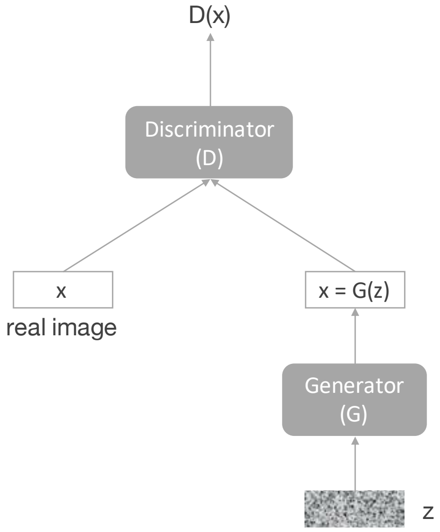

Figure 3: The overall structure of GAN.

- The discriminator network $\mathbf{D}$ performs a binary classification task which tries to classify real vs. fake samples with $\mathbf{D}(x)$ where x can be a real or generated image.

We train the discriminator in the same way as the binary classification network using the cross-entropy loss function:

$$\mathbb{L}_{\mathbf{D}} = \mathbb{E}_{x\sim\mathbf{D}}\big[-\log\mathbf{D}(x)\big] + \mathbb{E}_z\big[-\log(1-\mathbf{D}(\mathbf{G}(z)))\big]$$

where x denotes images from training set and z denotes images generated by the generator. Since the discriminator tries to *minimize* the error of predicting z, the simple way to train the generator to *maximize* the discriminator's error:

$$\mathbb{L}_{\mathbf{G}} = -\mathbb{L}_{\mathbf{D}} = \mathbb{E}_z\big[-\log(1-\mathbf{D}(\mathbf{G}(z)))\big]$$

2 Before Doing the Assignment

If you are familiar with other tensor manipulation frameworks with gradient-based optimization feel free to use them. If you are not familiar with any, we recommend you to go through the Pytorch Core tutorial beforehand (`https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html` 1 hours).

Before doing the assignment, it's recommended to go through reference papers.

3 Report Submission Guidelines

- Both report and implementation are graded (equal weights), if the question requires no code implementation then only the report is graded.

- Students should work on the assignments in a **group of 3**. Some minor additions and changes might be done during these three weeks. Students will be informed for these changes via Canvas.

- Any questions regarding the assignment content can be discussed on Canvas Discussions. Students are encouraged to contact the TAs if something is not clear. Please note that if you want a quicker answer to your queries, then we recommend the Canvas forums, since chances there are higher that a TA will see your inquiry and respond to it.

Students are expected to do this assignment in Python and Pytorch, however students are free to choose other tools (like Tensorflow). However, supporting code and additional tutorials will be provided in Python an Pytorch. Please provide requirements.txt with all external dependencies listed and README.txt file with description about how to reproduce results.

Source code and report must be handed in together in a zip file (ID_lastname.zip) before the deadline on Canvas. **Fine-tuned model please upload separately to UvA OneDrive and provide a link in the report**. For full credit, make sure your report follows these guidelines:

- Include an introduction and a conclusion to your report.

- The maximum number of pages is 10 (single-column, including tables and figures). Please express your thoughts concisely. The number of words does not necessarily correlate with how well you understand the concepts.

- Answer all given questions. Briefly describe what you implemented.

- Reports with just results and no explanation or text will automatically have lower grades.

- Try to understand the problem as much as you can. When answering a question, give evidences (qualitative and/or quantitative results, references to papers, etc.) to support your arguments.

- Analyze your results and discuss them, e.g. why algorithm A works better than algorithm B in a certain problem.

- Tables and figures must be accompanied by a brief description. Do not forget to add a number, a title, and if applicable name and unit of variables in a table, name and unit of axes and legends in a figure.

Late submissions are not allowed. Assignments that are submitted after the strict deadline will not be graded. In case of submission conflicts, TAs' system clock will be taken as the reference. We strongly recommend submitting well in advance, to avoid last minute system failure issues.

Plagiarism note: Keep in mind that plagiarism (submitted materials which are not your work) is a serious crime and any misconduct shall be punished according to the university regulations.

4 Assignment

4.1 Deep Learning Based Face Swap (15 points)

In this part of the assignment, we will play with Deep Learning-based Face Swap algorithm. To simplify the setup, we have prepared for you a Google Colab notebook

`https://colab.research.google.com/drive/12wPG41wak3PktRAOkUF6MMHXFXf3VTOb?usp=sharing`

Feel free to reproduce on Surfsara server if necessary.

- Face landmark detection is quite a challenging task on extreme boundary cases (with extreme pose, expression, or lighting). Try out with different images to evaluate the robustness of the landmark detection algorithm.
- Try out naive face swap implementation and deep-learning-based face swap algorithm on different images. Compare results.

To implement the entire Deep Learning Face Swap would be infeasible for a single course assignment, consequently, we will further focus on implementing specific components of the pipeline.

4.2 Optimization-based Face Swapping (55 points)

4.2.1 Morphable model (5 points)

Human face geometry can be represented as a point cloud of N vertices $\mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta}) \in \mathbb{R}^{N \times 3}$ using a multilinear PCA model [Blanz and Vetter [1999]]:

$$\mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta}) = \boldsymbol{\mu}_{id} + \mathbf{E}_{id}[\boldsymbol{\alpha} \cdot \boldsymbol{\sigma}_{id}] + \boldsymbol{\mu}_{exp} + \mathbf{E}_{exp}[\boldsymbol{\delta} \cdot \boldsymbol{\sigma}_{exp}] \tag{4}$$

where $\boldsymbol{\mu}_{id} \in \mathbb{R}^{N \times 3}$, $\boldsymbol{\mu}_{exp} \in \mathbb{R}^{N \times 3}$ are mean neutral geometry (facial identity) and mean facial expression; $\mathbf{E}_{id} \in \mathbb{R}^{N \times 3 \times 30}$ and $\mathbf{E}_{exp} \in \mathbb{R}^{N \times 3 \times 20}$ are principal components for neutral geometry and facial expression with their standard deviations $\boldsymbol{\sigma}_{id} \in \mathbb{R}^{30}$, $\boldsymbol{\sigma}_{exp} \in \mathbb{R}^{20}$; $\boldsymbol{\alpha} \in \mathbb{R}^{30}$ and $\boldsymbol{\delta} \in \mathbb{R}^{20}$ are latent parameters we need to estimate.

PCA models constraint the search space of possible face models and different face geometry can be generated by changing parameters $\boldsymbol{\alpha}$ or $\boldsymbol{\delta}$.

For this assignment, we will use the first 30 principal components for facial identity and 20 principal components for facial expression. However, the morphable model provides more parameters, which can be used to increase the expressiveness of the PCA model. Morphable model Basel Face Model 2017 (BFM) is available for download from (`https://faces.dmi.unibas.ch/bfm/bfm2017.html`, `model2017-1_face12_nomouth.h5`). Note that a non-commercial license doesn't allow you to distribute the model, consequently each student should download it from the website himself/herself.

BFM can be loaded using h5py library:

```
import h5py
import numpy as np

bfm = h5py.File(”model2017−1_face12_nomouth.h5”, ’r’)

# Select a specific weight from BFM
weights = np.asarray(bfm['shape/model/mean'], dtype=np.float32)
# Sometimes you will need to reshape it to a proper shape
weights = np.reshape(weights, (−1, 3))
```

The PCA model for facial identity can be extracted via keys `shape/model/mean` $\in \mathbb{R}^{3N}$, `shape/model/pcaBasis` $\in \mathbb{R}^{3N \times 199}$, `shape/model/pcaVariance` $\in \mathbb{R}^{199}$ from `model2017-1_face12_nomouth.h5`; PCA model for expression - via keys `expression/model/mean` $\in \mathbb{R}^{3N}$, `expression/model/pcaBasis` $\in \mathbb{R}^{3N \times 100}$, `expression/model/pcaVariance` $\in \mathbb{R}^{100}$, triangle topology - via keys `shape/representer/cells` $\in \mathbb{R}^{3 \times K}$ where K is an amount of triangles. Slice top M columns if you need M components.

- Extract 30 PC for facial identity and 20 PC for expression and generate a point cloud using Eq. 4. Uniformly sample $\boldsymbol{\alpha} \sim U(-1, 1)$, $\boldsymbol{\delta} \sim U(-1, 1)$. In the report show multiple point cloud samples. For vertex colour you can use mean face colour available from `color/model/mean`. Mean color should be reshaped to $N \times 3$

- We have provided save_obj function which saves a point cloud, its color and triangle topology into a 3D model. *.obj files can be opened using MeshLab `http://www.meshlab.net/` or other 3D viewers.

4.2.2 Pinhole camera model (10 points)

Given a 3D model from eq. 4 we assume face vertices $\{x, y, z\} \in \mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta})$ is rigidly transformed using transformation matrix $\boldsymbol{T} \in \mathbb{R}^{4\times4}$ in the scene and projected using $\mathbf{\Pi} \in \mathbb{R}^{\mathbf{4\times4}}$ into a camera plane (Equation 3).

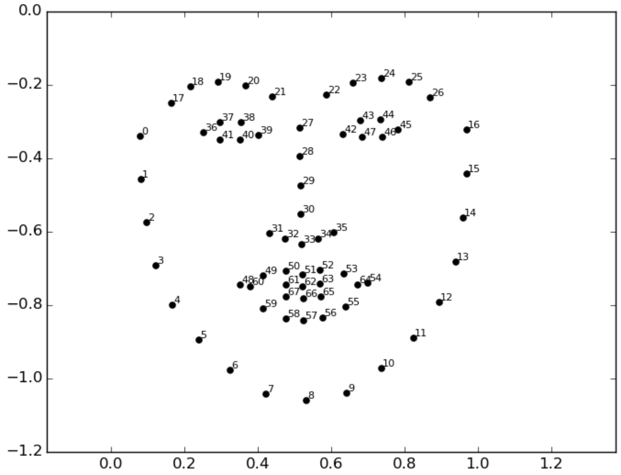

Figure 4: DLib format landmark points. Source: StackOverflow

- Assuming object translation to be 0. Rotate an object 10^o and -10^o around Oy and visualize results. `MeshLab` can be used together with save_obj.
- Assuming object translation to be (0; 0, -500), and rotation around Oy to be 10^o visualize facial landmark points on the 2D image plane using Eq. 3 (Fig.4 is an example). Vertex indexes annotation is available in the provided file `Landmarks68_model2017-1_face12_nomouth.anl`.

4.2.3 Latent parameters estimation (15 points)

So far we have built a pipeline which can infer facial landmarks given facial geometry latent parameters $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$ and object transformation $\boldsymbol{\omega}$, $\boldsymbol{t}$. In this section, we will estimate those parameters for a specific 2D image with a human face using Energy minimization. Given 68 ground truth facial landmarks the following energy can be optimized:

$$\mathcal{L}_{fit} = \mathcal{L}_{lan} + \mathcal{L}_{reg} \tag{5}$$

$$\mathcal{L}_{lan} = \frac{1}{68} \sum_{j=1}^{68} \left\| \mathbf{p}_{k_j} - \mathbf{l}_j \right\|_2^2 \tag{6}$$

where $\mathbf{p}_{k_j} = \{u_{k_j}, v_{k_j}\}$ is a 2D projection of a landmark point k_j from `Landmarks68_model2017-1_face12_nomouth.anl` and $\mathbf{l}_j$ is its ground truth 2D coordinate. We regularize the model using Tikhonov regularization to enforce the model to predict faces closer to the mean:

$$\mathcal{L}_{reg} = \lambda_{alpha} \sum_{i=1}^{30} \alpha_i^2 + \lambda_{delta} \sum_{i=1}^{20} \delta_i^2 \tag{7}$$

- Take a single picture of yourself or pick random one from the web. Extract ground truth landmarks using Dlib (`http://dlib.net/face_landmark_detection.py.html`). Keep face closer to the frontal and neutral for now. Visualize results. detect_landmark function can be found also in the supplemental code
- Assuming $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$, $\boldsymbol{\omega}$, $\boldsymbol{t}$ to be latent parameters of your model optimize an Energy described above using Adam optimizer until convergence. Visualize predicted landmarks overlayed on ground truth (**5 points**). Hint: in Pytorch $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$, $\boldsymbol{\omega}$, $\boldsymbol{t}$ can be declared as `Variable`.
- **Hint**: initializing transformation parameters $\boldsymbol{\omega}$ and $\boldsymbol{t}$ closer to the solution may help with convergence. For example translation over z dimension can be set to be -500 in the case of projection matrix with principal point $\{\frac{W}{2}, \frac{H}{2}\}$ and fovy = 0.5.
- Select hyperparameters such that $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ to be obtained in a proper range. Report findings.

4.2.4 Texturing (5 points)

Given latent parameters $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$, $\boldsymbol{\omega}$, $\boldsymbol{t}$ u,v projection coordinates for each point of the 3D model (Eq. 4) can be obtained by bilinear interpolation (`https://en.wikipedia.org/wiki/Bilinear_interpolation`) of neighborhood pixels.

- Compute 2D projection for each point given estimated latent parameters. For each point extract corresponded RGB value from a 2D image as a vertex colour. Visualize obtained results using a provided `render` function.

4.2.5 Energy optimization using multiple frames (5 points)

So far we have built a 2D-to-3D reconstruction algorithm which accepts 1 single monocular images and predicts latent parameters $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$, $\boldsymbol{\omega}$, $\boldsymbol{t}$ for a 3D model. Assuming we have multiple (M) images of the single person we can extend our model to constraint neutral face geometry using multiple frames with corresponding object pose $\boldsymbol{\omega}_i$, $\boldsymbol{t}_i$, i = 1..M.

$$\forall \text{i} = 1..\text{M } \mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta}) = \boldsymbol{\mu}_{id} + \mathbf{E}_{id}[\boldsymbol{\alpha} \cdot \boldsymbol{\sigma}_{id}] + \boldsymbol{\mu}_{exp} + \mathbf{E}_{exp}[\boldsymbol{\delta}_i \cdot \boldsymbol{\sigma}_{exp}] \tag{8}$$

- Extend the current Energy minimization pipeline (Sections 4.2.2, 4.2.3) for multiple frames optimization. Report transformed, textured mesh for each frame.

4.2.6 Face Swapping (15 points)

Face swapping is focused on changing face identity of an image while keeping other attributes. To swap faces of 2 different person, one need to take a texture from one face (Section 4.2.4) and to render it using a predicted geometry from another face.

- Try out the optimization-based face swapping on different images. Compare results with Deep Learning-based face swap results from Section 4.1.

4.3 Blending (30 points)

One common problem with face swapping is the unrealistic appearance of the target with the swapped face. Just placing the 3D reconstructed and pose-corrected face will not give a natural-looking image, rather a mask-like appearance. To make sure that the 3D reconstruction placed on the target image looks realistic, blending techniques can be employed. In this section, we will implement a blending pipeline that takes as input the 1) source image, 2) the 3D reconstruction that you obtain from the previous steps, and 3) the target image. The pipeline will output a single image that is then blended 3D mask onto the target, which should make it look more natural. To help you along, we will be providing you with a pretrained model that you will need to finetune on the provided dataset. The blending pipeline is based from FSGAN (Nirkin et al. [2019]).

4.3.1 GAN Training Pipeline

- Complete the training and data loading script files provided. For the training, you need to train a GAN with 1 Generator update along with 1 Discriminator update. The training loss will initially be the same Poisson blending functions as described in the original work. Use the dataset provided. **(5 points)**
- The network takes as input a concatenation of the target face overlaid with the 3D reconstructed face $\hat{t}$, the target image t and the face mask m. The mask can be generated from the 3D reconstruction face (f) that is provided for each source image. The f is already pose aligned to the target. So you can use a simple thresholding operation to generate the masks for the target. Use the function "transfer_mask" in the script to get the overlaid face. That is:

  $$I_{fake} = net(x)x = cat(\hat{t}, t, m)$$

  where I_{fake} is the network output. *net* is the generator model being trained. *cat* is a channel-wise concatenation operation. **(2.5 points)**
- The ground truth (gt) is generated using the Poisson blending function in opencv. Used the function "blend_imgs" to obtain the gt. You can refer to the OpenCV documentation on the function seamlessClone for more details about the function. **(2.5 points)**

4.3.2 Losses

For the initial loss, we will use the same training scheme as in the FSGAN paper. The GAN losses are provided in the file "gan_loss" and the VGGLoss (VL) is provided in the file "vgg_loss". For the training use the configuration of "ls_gan" when initializing the gan loss. Include the following generator losses in your training pipeline **(5 points)**:

$$\mathcal{L}_{id} = VL(I_{fake}, gt) \tag{9}$$

$$\mathcal{L}_{pixel} = L1(I_{fake}, gt) \tag{10}$$

$$\mathcal{L}_{attr} = VL(I_{fake}, gt) \tag{11}$$

$$\mathcal{L}_{G} = GAN(D(G(x)), True) \tag{12}$$

$$\mathcal{L}_{\mathbf{G}} = -\mathcal{L}_{\mathbf{D}} = \mathbb{E}_x\big[-\log(1 - \mathbf{D}(\mathbf{G}(x)))\big] \tag{13}$$

where λ_{rec} is set as 1.0, λ_{pix} is 0.1 and λ_G is 0.001. The discriminator losses are weighted by 0.5.

You can set the initial optimizers and LR Schedulers settings as follows:

- Generator LR = $1e - 4$
- Discriminator LR = $1e - 4$
- Momemtum = 0.9
- Weight Decay = 1e-4
- Step size = 30
- Gamma = 0.1

In addition to Poisson blending as described in the paper, we will also investigate the quality of different blending techniques. For this part, we will focus on two types of blending.

- **Alpha Blending:** Swap the GT generation with alpha blending. For this you can use the inbuilt function in OpenCV. Run your training with this loss. **(2.5 points)**
- **Laplacian Pyramid Blending:** For this loss, implement a Laplacian Pyramid Blending function. Replace this as your gt generator. For the scope of this assignment, you don't have to implement it in pure Pytorch. Instead, implement it using opencv libraries and then convert the resulting image into a tensor. You can refer to the implementation of the Poisson blending under the function "blend_imgs" for this. **(2.5 points)**

4.3.3 Analysis

- Make a qualitative comparison on the outputs from your different implementations ([a] Naive Face Swap, [b] Deep Learning-based, and [c] optimization-based). Check the swap edges of the predicted image. Does it look realistic? Compare with the generated gt. What does the network learn? Where does it fail? **(3 points)**
- For losses, analyze the effect of them. How does the different loss affect the output? Are they better or worse than the Poisson loss? How do they handle the change of illumination between the source and target faces? What might you suggest for further improvements? Analyze and report your findings. **(2 points)**
- So far the evaluation metric was qualitative comparisons. Due to missing ground-truths traditional quantitative (numerical) metrics are not applicable. Use the Inception Score metric (Heusel et al. [2017]). You can use the pytorch implementation found here: https://github.com/mseitzer/pytorch-fid

  Explore the literature to apply a suitable quantitative metric that might be better able to describe the "realness" of an image. **(5 points)**

5 Bonus points

5.1 Face-swapping on Video (10 points)

- Try your implemented algorithm on a video (¡ 1 min length). Analyze results. (5 points).
- Improve the quality of results. Justify your improvement based on benchmarks and metrics you have introduced in the previous part of this assignment. (5 points)

5.2 Segmentation Masks and Analysis (5 points)

In this section, we will study the influence of the various mask. Since we generate the masks from the 3D reconstructed face, oftentimes occlusions such as hair and headwear are ignored. The dataset provided to you, however, comes with multiple mask modalities like hair, skin, nose, mouth etc. For bonus points, try to play around with different mask modalities that are provided in the dataset to replace the previous mask m that you were using. Does it make the output look more realistic? What other masks or algorithms might be more helpful? Analyze and report your findings.

Note: This section is for bonus points and is optional. Please only attempt it if you have additional time. Focus on the previous sections first if you are pressed for time.

6 Self-Evaluation

Please describe briefly in an appendix to the report about contribution of each team member to this assignment. Self-evaluation will be taken into consideration in the case if contributions are significantly unequal.

References

V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 187–194, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605. doi: 10.1145/311535.311556. URL `https://doi.org/10.1145/311535.311556`.

Y. Deng, J. Yang, S. Xu, D. Chen, Y. Jia, and X. Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *IEEE Computer Vision and Pattern Recognition Workshops*, 2019.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL `http://arxiv.org/abs/1706.08500`.

Y. Nirkin, Y. Keller, and T. Hassner. FSGAN: Subject agnostic face swapping and reenactment. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7184–7193, 2019.

A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Pérez, and C. Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. *CoRR*, abs/1703.10580, 2017. URL `http://arxiv.org/abs/1703.10580`.