Computer Vision 2 - 2D to 3D

Maurice Frank - 11650656 (UvA)
maurice.frank@posteo.de

David Biertimpel - 12324418 (UvA)
david.biertimpel@student.uva.nl

Leonardo Romor - 12261734 (UvA)
leonardo.romor@student.uva.nl

(All contributed equally)

June 2, 2019

Introduction

Avatar manipulation systems try to map motion and shape cues landmarks from images to generate and animate 3D avatars. Based on this approach, some models are capable to generate realistic videos, where animated facial expressions are transferred to target faces as in [5] or [3]. In the following we implement a miniature version of these approaches by following several steps of the 2D-to-3D reconstruction pipeline. First, we generate faces by using a 3D morphable face model [1]. Afterward, we build a pinhole camera to project the face model's 3D world coordinates onto a camera plane. Based on landmark points from the 3D face model and ground truth landmark points from a portrait image, we optimize latent parameters that modifies the face model to resemble a person in a the portrait image. After extracting the face texture from the image and extending our approach to consider multiple images for optimization, we finally apply the 2D-to-3D reconstruction to a video sequence. Our results show a good reconstruction performance given stationary images and satisfactory performance for a video sequence.

1 Morphable Model

One approach for 2D-to-3D reconstruction is to capture prior knowledge of the desired object by a parametric model. In the case of human faces, one such strategy is to use *Morphable Face Models* like the Basel Face Model 2017 [1]. The Basel Face Model is a collection of generative shape models containing textures, average shapes and expressions which are represented as principle components of the space of all captured faces. This allows us to approximate humans faces as linear combination of the basis vectors of this space.

A single face can be generated with a multilinear PCA, where the latent parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ act as weights on the principle components for facial identity and facial expression respectively and thus parameterize the face model. In Figure 1 we show different faces generated by $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ values that were sampled uniformly between -1 and 1. As described in the exercise, we used the first 30 principal components of the facial identity and the first 20 principal components of the facial expressions.

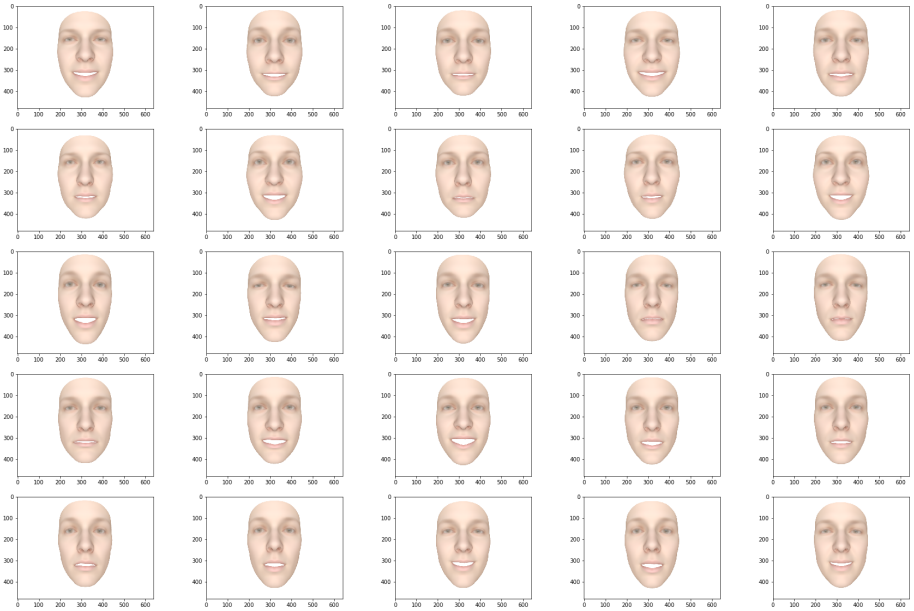


Figure 1: Showing 25 faces generated by $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ values sampled uniformly from values between -1 and 1.

2 Pinhole camera model

In the next step in our 2D-to-3D pipeline, we construct a pinhole camera model that maps world coordinates to image space. For this purpose, we define a transformation matrix $\boldsymbol{T} \in \mathbb{R}^{4\times4}$ that maps world coordinates to the camera space and a perspective projection matrix $\boldsymbol{P} \in \mathbb{R}^{4\times4}$ and viewpoint matrix $\boldsymbol{V} \in \mathbb{R}^{4\times4}$ that finally map the points on the camera plane (image space). We construct $\boldsymbol{P}$ and $\boldsymbol{V}$ by following [1] and [2] respectively. Each homogeneous world coordinate $\boldsymbol{x}$ is projected to the a image plane, by multiplying it with our pinhole camera model with $\hat{\boldsymbol{x}} = \boldsymbol{V} \times \boldsymbol{P} \times \boldsymbol{T} \times \boldsymbol{x}$, where $\hat{\boldsymbol{x}} = \begin{bmatrix}\hat{x} & \hat{y} & \hat{z} & \hat{d}\end{bmatrix}^T$ and $\hat{\boldsymbol{x}}_p = \begin{bmatrix}\hat{x}/\hat{d} & \hat{y}/\hat{d} & \hat{z}/\hat{d} & 1\end{bmatrix}^T$ is the final projected coordinate.

We initialize $\boldsymbol{P}$ by setting the near plane to 300, the far plane to 2000 and the vertical field of view (`FOVY`) to 1. $\boldsymbol{V}$ is initialized by the dimensions of the desired output image plane. The transformation matrix $\boldsymbol{T}$ embodies a rigid transformation consisting of a rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^{3\times1}$.

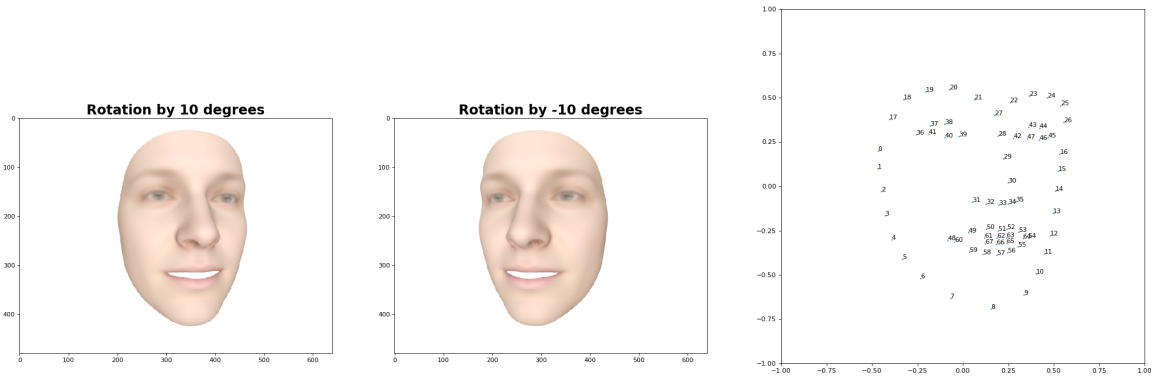
In figure 2a we show the mean face model rotated -10 and 10 degrees around the Y-axis. Considering the provided file `Landmarks68_model2017-1_face12_nomouth.anl` we can extract landmark points from the 3D face model. In figure 2b we see the face model's landmark points rotated by 10 degrees and projected onto the image plane. Further the landmark points are annotated by their index.

3 Latent parameters estimation

After setting up both our face model and pinhole camera, we want our 3D model to resemble a human face from a given portrait image. For this purpose, we optimize our latent parameters $\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{R}$ and $\boldsymbol{t}$ by aligning the facial landmarks from the previous exercise with ground truth landmarks. We infer these ground truth landmarks by applying the face landmark detection from the *Dlib toolkit* [2] on the

[1] `https://www.scratchapixel.com/lessons/3d-basic-rendering/perspective-and-orthographic-projection-matrix/opengl-perspective-projection-matrix`

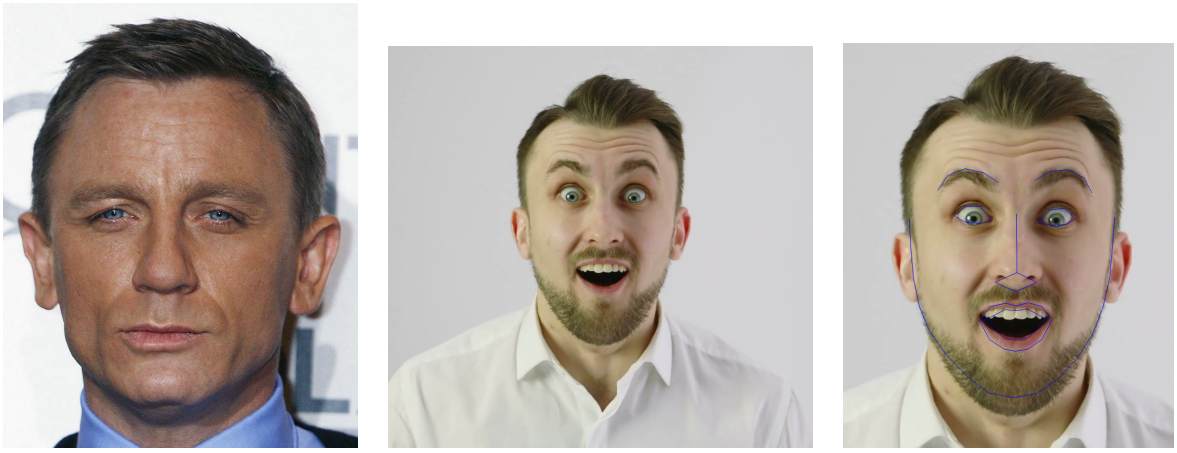
[2] `http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/projection/viewport_transformation.html`



(a) (b)

Figure 2: (a) Faces rotated by 10 and -10 degrees respectively. (b) Facial landmark points rotated by 10 degrees and projected to 2D.

portrait image. An example of the detected facial landmarks can be found in figure 3c. Generally, we are going to test our implementation on two example images. One image showing a neutral face (see Figure 3a) and one image showing strong surprise (see Figure 3b).



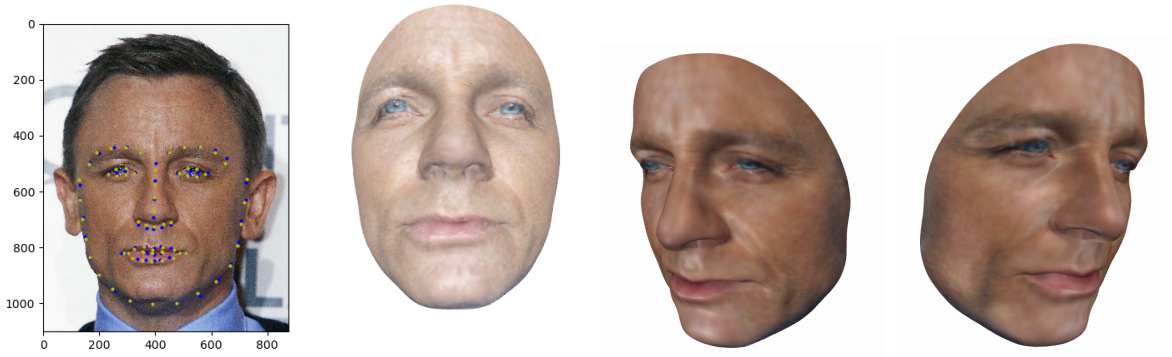
(a) (b) (c)

Figure 3: (a) & (b) show the example images we are going to use to test our 2D-to-3D reconstruction. (c) Shows the ground truth landmarks estimated by *Dlib toolkit*

To optimize the latent parameters we minimize the energy formulated in the task over 100 iterations. Further, we set the regularization terms to $\lambda_{alpha} = 50$ and $\lambda_{delta} = 1$. Since we want to begin our optimization with a neutral face, we initialize $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ as zero vectors. We further initialize the rotation matrix $\boldsymbol{R}$ as a identity matrix and $\boldsymbol{t} = [0 \quad 0 \quad -400]$ to correctly align the face model with the pinhole camera. During optimization, we use the Adam optimizer with an initial learning rate of $\eta = 0.1$. In a single optimization step, we first use the current $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ to transform the landmarks before projecting them to 2D using the pinhole camera model as described in section 2. Afterwards, we calculate the current energy by plugging both the predicted and ground truth landmarks into the energy equation. We finally use this energy to update our latent parameters via gradient descent. For the whole optimization process we use the deep learning platform *Pytorch* [4].

After optimizing we apply the estimated latent parameters to the whole 3D face model (all the points) to obtain a 3D face that resembles the person in the given portrait image. In Figure 4 we see the results achieved with the neutral face. We see that the optimization managed to capture the facial identity and expression well, which is especially evident when looking at the areas around the mouth

and the eyebrows. In Figure 5 we can observe the results achieved with the surprised face. While we also consider this reconstruction to be a good fit, we observe inaccuracies in the mouth area, as some parts of the teeth are projected onto the upper lip.



(a) (b) (c) (d)

Figure 4: Depiction of the results for the neutral face. (a) Shows the landmarks, where blue dots are our model's prediction and the yellow dots are the ground truth. (b) Rendering of the final 3D face with applied texturing (see section 4). (c) & (d) Two additional perspectives of the final 3D model.

(a) (b) (c) (d)

Figure 5: Depiction of the results for the surprised face. (a) & (b) & (c) & (d) see Figure 4.

We now consider the energy over the iterations during optimization. As shown in Figure 6a, the energy starts high and rapidly decreases in the first 50 iterations. Afterwards, it starts approaching a plateau. The energy for the neutral face is overall higher than that of the surprised face. Here it should be noted that the scale of the energy is connected with the size of the images (pixel coordinates of the face), as we do not normalize the landmarks (stated on piazza). Also the correct values for λ_{alpha} and λ_{delta} are dependent on the image size. Although we choose images of similar size, the energy over the iterations can thus only be used to draw conclusions about general trends, not for elaborated statements.

Lastly, we consider the individual values of $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ and assess if they are reasonable. For this reason we visualize the values for both the neutral and surprised face in box plots which can be observed in Figure 7. When looking at the box plots, we first notice that all four are roughly centered around zero and no plot shows significant amounts of outliers. When looking at the values in $\boldsymbol{\delta}$ (range neutral: -7.93 to 2.04; range surprised: -2.88 to 2.95), we observe that their scale is significantly larger than the values in $\boldsymbol{\alpha}$ (neutral: -0.34 to 0.23; surprised: -0.13 to 0.09). This difference in scale is due to the empirically determined regularization values λ_{alpha} and λ_{delta}. We found that a small value for λ_{alpha} leads to a poor representation of the facial identity. We then empirically set it to a larger value ($\lambda_{alpha} = 50$), which introduces more regularization and thus keeps the $\boldsymbol{\alpha}$ values small. Regarding

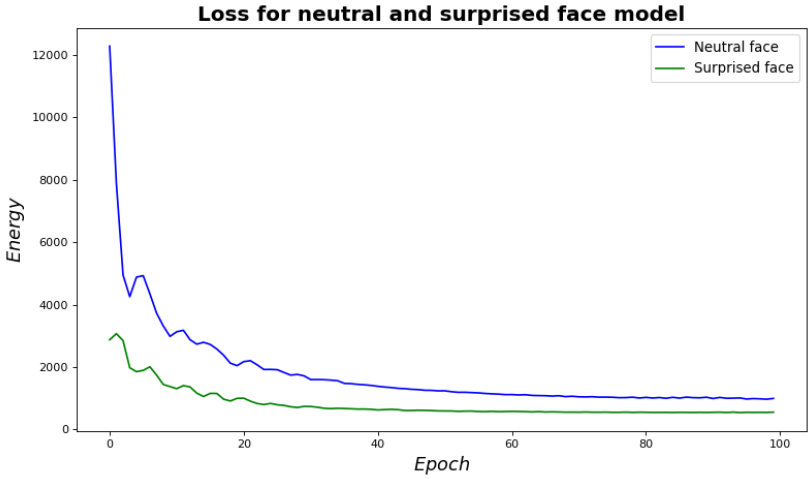


Figure 6: Loss obtained during optimization for both the neutral face and the surprised face.

λ_{delta} we found that large values restrict the facial expression to much, such that the expression from the portrait image is poorly transferred. A smaller value ($\lambda_{delta} = 1$) results in less regularization, which increases the values in $\boldsymbol{\delta}$ and allows a better expression transfer.

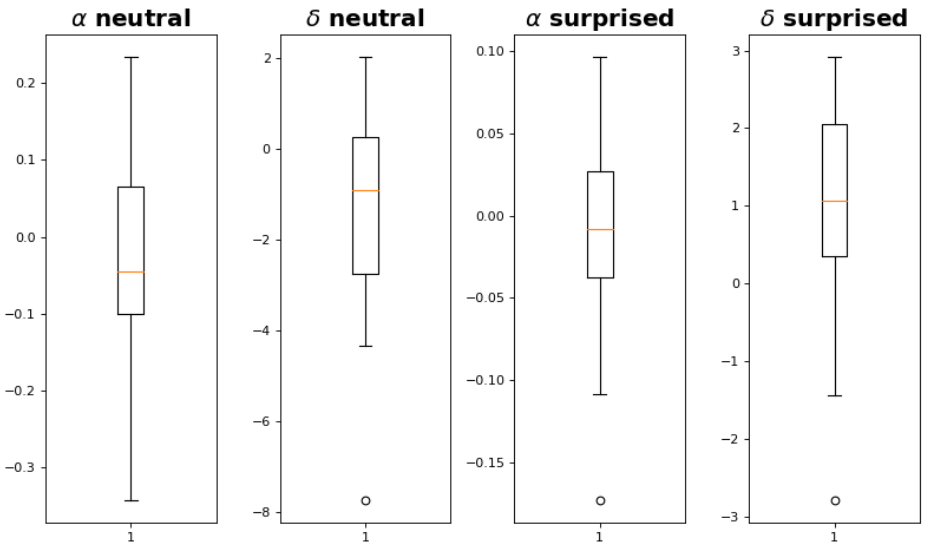


Figure 7: Loss obtained during optimization for both the neutral face and the surprised face.

4 Texturing

Once the latent parameters are estimated we proceed to extract the texture of the 3D face from the source image. For this purpose we project the complete 3D face into the image space using the optimized parameters, so that the points of the face model are aligned with the face in the portrait image. We now could estimate each points color by using nearest neighbour. However, this procedure would lead to a discontinuous color function and thus negatively influence the texture quality. To improve the quality of the color estimation, we infer each point's color using bilinear interpolation. Since each projected point p lies in between four pixel values, we can calculate a weighted average of these surrounding pixel colors. For this we first find the four surrounding pixel coordinates p_{11} (top left), p_{12} (top right), p_{21} (bottom left) and p_{22} (bottom right) by flooring and ceiling p_x and p_y (x-

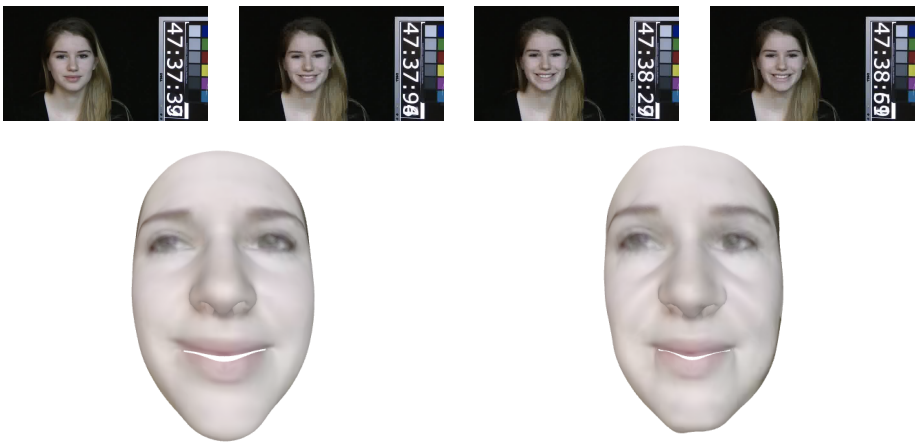


Figure 8: First row are the four frames. Bottom row we show the reconstructed face for the first frame. *Left* using only the first frame and *right* using all 4 frames.

and y coordinates of p). We then linear interpolate in the x-direction with

$$p_h = \frac{p_x - p_{11}}{p_{12} - p_{11}}; \quad p_v = \frac{p_y - p_{21}}{p_{22} - p_{21}}.$$

We then use the obtained p_h (percentage between horizontal positions) and p_v (percentage between vertical positions) to calculate weighted averages of the colors. When c_{11}, c_{12}, c_{21} and c_{22} are the colors of the surrounding pixels, we can find the final interpolated color with

$$\begin{aligned} c_{h1} &= c_{11} \cdot (1 - p_h) + c_{12} \cdot p_h \\ c_{h2} &= c_{21} \cdot (1 - p_h) + c_{22} \cdot p_h \\ c_{final} &= c_{h1} \cdot (1 - p_v) + c_{h2} \cdot p_v, \end{aligned}$$

where c_{h1} and c_{h2} are the weighted averages of the colors in the x-direction and c_{final} the weighted average between c_{h1} and c_{h2} using p_v. The results of this bilinear interpolation can be observed in Figure 4 and Figure 5.

5 Energy optimization using multiple frames

We now update the energy optimization to learn the appearance $\boldsymbol{\alpha}$ factors from multiple frames. The new loss then becomes:

$$\mathcal{L} = \sum_{m=1}^{M} \mathcal{L}_{lan}^{m} + \sum_{m=1}^{M} \mathcal{L}_{fit}^{m} \tag{1}$$

$$\mathcal{L}_{lan} = \sum_{j=1}^{68} ||\mathbf{p}_{kj} - \mathbf{l}_j^m||_2^2 \tag{2}$$

$$\mathcal{L}_{reg}^{m} = \lambda_{alpha} \sum_{i=1}^{30} \boldsymbol{\alpha}_i^2 + \lambda_{delta} \sum_{i=1}^{30} (\boldsymbol{\delta}_i^m)^2 \tag{3}$$

We optimize our latent variables with the same procedure as before and set $\lambda_{alpha} = 2$ $\lambda_{delta} = 1e-4$.

Figure 9: Reconstruction of the the smiling woman face at 10 different frames during the video.

See Figure 8 for the results using a video from the *UvA-NEMO Smile Database*[3][4]. As the appearance does not show too much variance we do not see big changes estimating $\boldsymbol{\alpha}$ with multiple frames.

We see some differences between using only the single image compared to all four frames. Notice in the right reconstruction the chin being more straight as it is in the image but on other parts we see worse results with multiple frames. For example the nose gets deformed. Mostly we suspect that with more frames it is more difficult to weight the loss with multiple frames between $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$.

6 Expression manipulation

After managing to reconstruct a 3D face from M portrait images, we can now extend this procedure to estimate J consecutive frames of a video sequence. To achieve this, we first take the first M frames of the video and optimize $\boldsymbol{\alpha}$ to capture the facial identity as before. From then on we are only interested in the facial expression and therefore we fix the facial identity so that $\boldsymbol{\alpha}$ becomes a shared parameter for the following frames. This leads us to optimize the parameters $\boldsymbol{\delta}_i$, $\boldsymbol{R}_i$ and $\boldsymbol{t}_i$ for all frames $i \in 1 \dots J$, while $\boldsymbol{\alpha}$ remains the same.

After each frame, we do not initialize $\boldsymbol{\delta}$ and the rigid transformation by values from the previous iteration, but rather reset the values to their initial state. This is because we found that inheriting previous parameters does not actively benefit the reconstruction, but in some cases even produce worse results.

For testing the expression manipulation we use the same video sequence from the *UvA-NEMO Smile Database* we already used in section 5. However, we set $\lambda_{alpha} = 50$ and $\lambda_{delta} = 0.1$. To obtain a full smiling gesture of the subject, we capture the expression transfer for 100 frames, which leads to a video sequence that is 4 seconds long given 25 frames per second. For estimating $\boldsymbol{\alpha}$ we take the first 4 frames of the video ($M = 4$). In the figure 9 we provide samples of the generated video by displaying 10 frames at equidistance. We also provide a video file in the *results* folder (`exercise_7_smiling.mp4`) showing the smiling face in motion.

Looking at the individual results we see that the expression is accurately transferred across the frames. However, we observe the same imprecisions in the mouth area that we have already observed on the surprised face (see Figure 5). When considering the generated video, we see the projected texture to slightly flicker across the frames, which introduces an unrealistic feeling to the smiling gesture.

[3] `http://www.uva-nemo.org/`

[4] `http://www.uva-nemo.org/samples/subject_020_posed_smile_2_640x360_30.mp4`

7 Discussion

7.0.1

First we implemented ICP. Vanilla ICP's advantage is that given the correct data the reconstruction will be of high quality but we need the actual 3D scans of the scene to perform this. Thus ICP has higher demands on the data. Further we saw that ICP only works good if the the consecutive frames are close enough.

Second we looked at SfM. Directly is clear that reconstructing the scene with mere moving image frames, introduces a less demanding data collection which makes it more applicable. On the other hand we showed that in SfM the quality of reconstruction delicately depends on how good we can match our image descriptors. Given a more complex scenery the descriptors will be harder to match, which results in a sparse Point-View-Matrix.

7.0.2

ICP and SfM can be combined in certain situations. For example if we reconstruct from multiple diverse views at different scales it is difficult to find the common scale for the images. Here we can use ICP to triangulate more points and merge them into the scene.

7.0.3

The 2D-to-3D method we explored here has the obvious advantage that it can estimate a reconstruction from one single image, even at low quality. This of course is because we apply the prior of the mean face geometry which also is the methods big disadvantage. The method only works for objects were we assume a general prior geometry with not too much variance. Assuming this is possible as it is with faces we can improve the reconstruction by introducing the previous methods. If we have a 3D scan of course we can use these depth information to improve the fine structure of a head. Same holds for using SfM. We can use the image frames to estimate a separate reconstruction that we can merge into the other model. As we already find the landmarks in the images in our method here we actually already have matching keypoints and can directly estimate a reconstruction from this.

Conclusion

In this assignment we build a 2D-to-3D reconstruction method for portrait images of faces. Our results show that the algorithm can extract appearance and expression information from already one image. While we achieved good reconstruction performance for stationary images in section 3, the performance slightly decreased over the following exercises. In the end, we achieved to generate a video by transferring expressions from another video to a target avatar. It should be noted that the approach uses an external face landmark detector and heavily relies on the prior information provided by the morphable face model. The good results are mostly due to these two parts and thus makes this not easily transferable to other domains.

References

[1] Thomas Gerig, Andreas Forster, Clemens Blumer, Bernhard Egger, Marcel Lüthi, Sandro Schönborn, and Thomas Vetter. Morphable face models - an open framework. *CoRR*,

abs/1709.08398, 2017.

[2] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[3] Koki Nagano, Jaewoo Seo, Jun Xing, Lingyu Wei, Zimo Li, Shunsuke Saito, Aviral Agarwal, Jens Fursund, and Hao Li. pagan: Real-time avatars using dynamic textures. *ACM Trans. Graph.*, 37(6):258:1–258:12, December 2018.

[4] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[5] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nie. Face2face: Real-time face capture and reenactment of rgb videos. *Commun. ACM*, 62(1):96–104, December 2018.