Deep Energies for Estimating Three-Dimensional Facial Pose and Expression

Jane Wu[1] Xinwei Yao[2]
[1]CS230 [2]CS229
{janehwu,yaodavid}@stanford.edu

Abstract

While much progress has been made in capturing high-quality facial performances using motion capture markers and shape-from-shading, high-end systems typically also rely on rotoscope curves hand-drawn on the image. These curves are subjective and difficult to draw consistently; moreover, ad-hoc procedural methods are required for generating matching rotoscope curves on synthetic renders embedded in the optimization used to determine three-dimensional facial pose and expression. We propose an alternative approach whereby these curves and other keypoints are detected automatically on both the image and the synthetic renders using trained neural networks, eliminating artist subjectivity and the ad-hoc procedures meant to mimic it. More generally, we propose using machine learning networks to implicitly define deep energies which when minimized using classical optimization techniques lead to three-dimensional facial pose and expression estimation.

1. Introduction

Face animation is an open area that remains challenging for researchers and practitioners, particularly in the capturing of details around the eyes and lips. As such, significant manual effort is still required in various stages including initial model alignment and fine-tuning blendshapes such that the resulting render visually resembles the corresponding motion capture image stills, called "plates." Our project seeks to automate manual parts of the face capture pipeline by leveraging pre-trained deep neural networks. We propose the use of sparse facial landmarks per frame to target facial pose (*e.g.* rotation and translation) and expression (*e.g.* blendshape model weights) with an anatomical model of an actor's face. After per-frame alignments, we propose to improve temporal coherence by using optical flow. Further, we modify existing network architectures to allow our system to be fully differentiable for non-linear optimization.

More concretely, we take as input a short facial performance of an actor, and a blendshape rig for the actor's face. The goal of the system is to automatically find for each frame the rotation and translation parameters of the 3D model as well as the blendshape weights, so that when animated, the model reproduces the actor's performance.

2. Related Work

Face Alignment: Deep face alignment networks are generally classified into coordinate regression models [11, 13, 27, 31], where a direct mapping is learned between the image and the landmark coordinates, and heatmap regression models [2, 4, 32], where prediction heatmaps are learned for each landmark. Heatmap-based architectures are generally derived from stacked hourglass [2, 4, 10, 21] or convolutional pose machine [29] architectures used for human body pose estimation. Pixel coordinates can be obtained from the heatmaps by applying the argmax operation; however, [5, 26] use soft-argmax to achieve end-to-end differentiability.

Optical Flow: In this project we focus on deep optical flow networks. End-to-end methods for learning optical flow using deep networks were first proposed by [6] and later refined in [9]. Other methods include DeepFlow [30], etc. use deep networks to detect correspondences. These methods are generally evaluated on the Middlebury dataset [1], the KITTI dataset [8], and the MPI Sintel dataset [3].

Using deep networks such as VGG-16 [25] for losses has been shown to be effective for training other deep networks for tasks such as style transfer and super-resolution [12]. Furthermore, deep networks have been used in energies for traditional optimization problems for style transfer [7], texture synthesis [24], and image generation [19, 28].

3. Dataset

We used two pre-trained networks for this project: 3D-FAN [2] and FlowNet2 [23]. 3D-FAN was trained on the LS3D-W dataset [2], and FlowNet2 was trained on FlyingChairs and FlyingThings3D [20].

We estimate the facial pose and expression on a moderately challenging performance captured by a single ARRI Alexa XT Studio running at 24 frames-per-second with an 180 degree shutter angle at ISO 800 where numerous cap-

(a) (b) (c) (d) (e) (f)

Figure 1. An overview of our approach: we use our three-dimensional face model of the actor (a) and estimate the albedo and spherical harmonics lighting in a neutral pose (b). Then, we can deform the face model into a variety of poses by changing the rigid parameters θ and t and blendshape parameters w (c), and generate synthetic images by rendering the face model in those poses (d). We feed that synthetic render through the network to produce a set of outputs (e), which are then compared to the outputs produced by the same network when feeding it the captured image (f).

tured images exhibit motion blur. These images are captured at a resolution of 2880×2160, but we downsample them to 720×540 before feeding them through our pipeline.

4. Overview

In this project we propose a general strategy that incorporates pre-trained deep neural networks into classical optimization methods. Specifically, the energy to be minimized in the optimization is based on outputs of neural networks. We take the following approach: First, we estimate an initial rigid alignment of the 3D face model to the 2D plate image using a facial alignment network. Next, we estimate an initial guess for the jaw and mouth expression using the same network. Finally, we temporally refine the results and insert/repair failed frames whenever necessary using an optical flow network.

We use a blendshape model with linear blend skinning for a single six degree of freedom jaw joint. Let w denote the parameters that drive the face triangulated surface $x(w)$ [15]. The rigid transformation is given by the Euler angles θ, its rotation matrix $R(\theta)$ and a translation t such that the final vertex positions are

$$x_R(\theta, t, w) = R(\theta)x(w) + t \tag{1}$$

Our goal is to determine the parameters θ, t and w that best match a given captured image F^*. The geometry x_R is rendered using OpenDR [17] obtaining a rendered image $F(x_R)$. Both the pixels of the captured image F^* and the pixels of the rendered image $F(x_R)$ are fed through the same deep network to get two sets of outputs $N(F^*)$ and $N(F)$. See Figure 1. We use the L_2 norm of the difference between them

$$\|N(F^*) - N(F(x_R(\theta, t, w)))\|_2 \tag{2}$$

as the objective(energy) function to minimize via nonlinear least squares. The resulting nonlinear least squares problem is solved using the Dogleg method [18] as implemented by Chumpy [16]. This method requires computing the Jacobian of the energy function via the chain rule with respect to the parameters to solve. We use OpenDR to compute $\partial F/\partial x_R$, and Equation 1 yields $\partial x_R/\partial\theta$, $\partial x_R/\partial t$, $\partial x_R/\partial x$. $\partial x/\partial w$ is handled by the differentiable blendshape rig, and $\partial N/\partial F$ is computed by back-propagating through the network using PyTorch [22].

5. Rigid Alignment

We first solve for θ and t using the pre-trained 3D-FAN network [2] as N. The images need to undergo several transformations before being fed into the network. First, we use a CNN-based face detector implemented by Dlib [14] to find the bounding boxes of the face. We then scale the bounding box to contain the entire face and crop it out, resizing it to 256×256 to feed into the network. We denote the face detection with D, cropping with C and resizing with S. The input into the network is thus $S(C(F(x_R), D(F(x_R))), D(F(x_R)))$, where the crop and resizing depends on the face bounding box.

3D-FAN outputs a tensor of size $68 \times 64 \times 64$, i.e. each of the 68 landmarks has a 64×64 heapmap specifying the likelihood of a particular 4×4 patch on the 256×256 image. Using argmax to determine landmarks position is not differentiable, so we instead follow the approach of [5, 26] and apply a soft-argmax function to the heatmaps to get an expected value of the landmark coordinate.

That is, given the marker position m_i computed using the argmax function on heatmap H_i, we use a 3×3 patch of pixels M_i around m_i to compute the soft-argmax position as

$$\hat{m}_i = \frac{\sum_{m \in M_i} m e^{\beta H_i(m)}}{\sum_{m \in M_i} e^{\beta H_i(m)}} \tag{3}$$

where $\beta = 50$ is set experimentally and $H_i(m)$ returns the

heatmap value at a pixel coordinate m. We found that using a small patch around the argmax landmark positions gives better results than running the soft-argmax operation on the entire heatmap.

The soft-argmax function returns an image coordinate on the 64×64 image, and these image coordinates need to be remapped to the full resolution image to capture translation between the synthetic face render and the captured image. Thus, we apply inverse rescale S_m^{-1} and crop operations C_m^{-1}, *i.e.* $\tilde{m}_i := C_m^{-1}(S_m^{-1}(4\hat{m}_i, D), D)$. The multiplication by 4 rescales from the 64×64 heatmap to the original 256×256.

6. Expression Estimation

After the rigid alignment determines θ and t, we solve for an initial estimate of the mouth and jaw blendshape parameters w. We use 3D-FAN in the same manner as discussed in previous section to solve for w while keeping the rigid parameters θ and t fixed. It is also sometimes beneficial or even preferred to allow θ and t to be modified somewhat at this stage as well. A prior energy term that penalizes these deviations from the values computed from the rigid alignment stage can also be added to the minimization.

7. Optical Flow for Missing Frames

The face detector used in Section 5 can sometimes fail, *e.g.* on our test sequence, the Dlib's HOG-based detector failed on 20 frames while Dlib's CNN-based detector succeeded on all frames. We thus propose using optical flow networks to infer the rigid and blendshape parameters for failed frames by "flowing" these parameters from surrounding frames where the face detector succeeded. This is accomplished by assuming that the optical flow of the synthetic render from one frame to the next should be identical to the corresponding optical flow of the captured image. That is, given two synthetic renders F_1 and F_2 and two captured images F_1^* and F_2^*, we can compute two optical flow fields $N(F_1, F_2)$ and $N(F_1^*, F_2^*)$ using FlowNet2 [9]. We resize the synthetic renders and captured images to a resolution of 512×512 before feeding them through the optical flow network. Assuming that F_2^* is the image the face detector failed on, we solve for the parameters p_2 of F_2 starting with an initial guess p_1, the parameters of F_1, by minimizing the L2 difference between the flow field vectors $\|N(F_1^*, F_2^*) - N(F_1, F_2)\|_2$. $\partial N / \partial F_2$ can be computed by back-propagating through the network.

8. Temporal Refinement

Since we solve for the rigid alignment and expression for all captured images in parallel, adjacent frames may produce visually disjointed results either because of noisy facial landmarks detected by 3D-FAN or due to the non-linear optimization converging to different local minima. Thus, we also use optical flow to refine temporal inconsistencies between adjacent frames. We adopt a method that can be run in parallel. Given three sequentially captured images F_1^*, F_2^*, and F_3^*, we compute two optical flow fields $N(F_1^*, F_2^*)$ and $N(F_2^*, F_3^*)$. Similarly, we can compute $N(F_1, F_2)$ and $N(F_2, F_3)$. Then, we solve for the parameters p_2 of F_2 by minimizing the sum of two L2 norms $\|N(F_1^*, F_2^*) - N(F_1, F_2)\|_2$ and $\|N(F_2^*, F_3^*) - N(F_2, F_3)\|_2$. The details for computing the Jacobian follow that in Section 7. Optionally, one may also wish to add a prior penalizing the parameters p_2 from deviating too far from their initial value. Here, step k of smoothing to obtain a new set of parameters p_i^k uses the parameters from the last step $p_{i\pm1}^{k-1}$; however, one could also use the updated parameter values $p_{i\pm1}^k$ whenever available in a Gauss-Seidel style approach.

Alternatively, one could adopt a self-smoothing approach by ignoring the capture image's optical flow and solving for the parameters p_2 that minimize $\|N(F_1, F_2) - N(F_2, F_3)\|_2$. Such an approach in effect minimizes the second derivative of the motion of the head in the image plane, causing any sudden motions to be smoothed out; however, since the energy function contains no knowledge of the data being targeted, it is possible for such a smoothing operation to cause the model to deviate from the captured image.

While we focus on exploring deep learning based techniques, more traditional smoothing/interpolation techniques can also be applied in place of or in addition to the proposed optical flow approaches. Such methods include: spline fitting the rigid parameters and blendshape weights, smoothing the detected landmarks/bounding boxes on the captured images as a preprocess, smoothing each frame's parameters using the adjacent frame's estimations, etc.

9. Results

In this section we present the results of applying our method to the performance data described in Section 3. We assume that the camera intrinsics and extrinsics have been pre-calibrated, the captured images have been undistorted, and that the face model described in Equation 1 has already been created. Furthermore, we assume that the face's rigid transform has been set such that the rendered face is initially visible and forward-facing in all the captured viewpoints.

9.1. Rigid Alignment

We estimate the rigid alignment (*i.e.* θ and t) of the face using 3D-FAN. We use an energy $E_1 = W(N(F) - N(F^*))$ where N are the image space coordinates of the facial landmarks as described in Section 5 and W is a per-landmark weighting matrix. Furthermore, we use an edge-

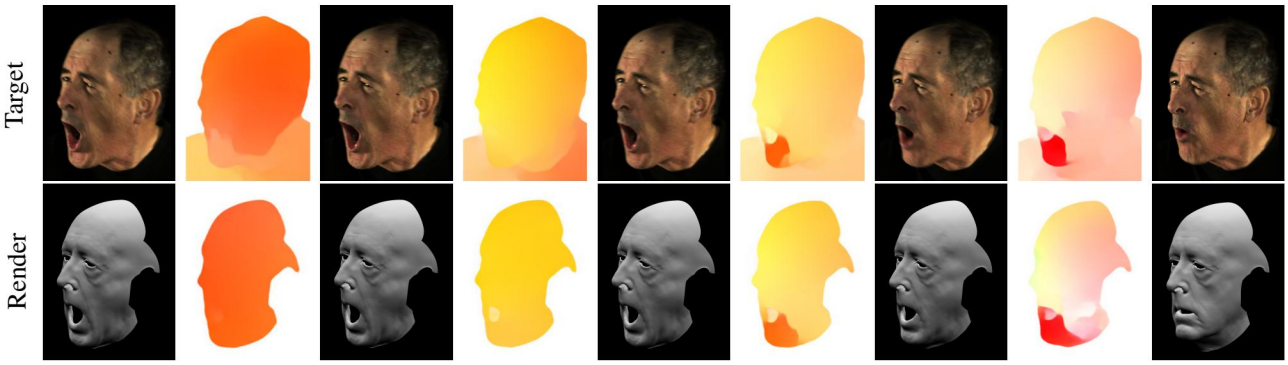


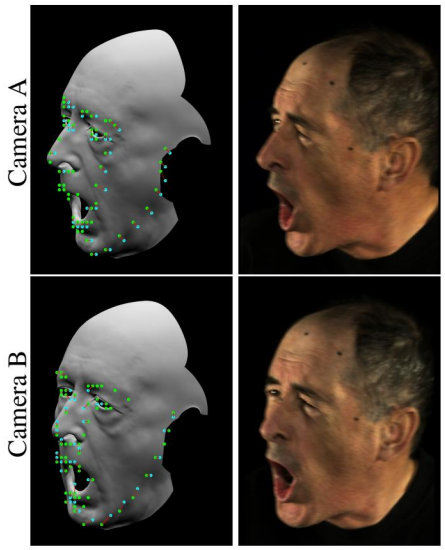
Figure 2. We use optical flow to infill frames where the face detector fails. Starting from frame 1142, the optimization moves the head to match the optical flow of the synthetic render to the optical flow of the captured image. After solving for frame 1145, we perform another round of optimization except that we start from frame 1146 instead; this way, each frame will capture optical flow information from both anchor frames. Using optical flow allows the mouth to stay open longer (*e.g.* in frame 1144) than what one would obtain using simple interpolation.

preserving energy $E_2 = \sum_i (\tilde{m}_i^{F^*} - \tilde{m}_{i-1}^{F^*}) - (\tilde{m}_i^F - \tilde{m}_{i-1}^F)$ where $\tilde{m}_i^{F^*}$ are the landmark positions on the captured image and $\tilde{m}_i^F$ are the landmark positions on the synthetic renders to ensure that the face does not erroneously grow/shrink in projected size as it moves towards the target landmarks, which may prevent the face detector from working.

First, we only solve for t using all the landmarks except for those around the jaw to bring the initial state of the face into the general area of the face on the captured image. We prevent the optimization from overfitting to the landmarks by limiting the maximum number of iterations. Next, we solve for both θ and t in three steps: using the non-jaw markers, using only the jaw markers, and using all markers. We perform these steps in stages as we generally found the non-jaw markers to be more reliable and use them to guide the face model to the approximate location before trying to fit to all existing markers.

9.2. Expression Estimation

We run a similar multi-stage process to estimate facial expression using the detected 3D-FAN landmarks. We use the same energy term E_1 as Section 9.1, but also introduce L2 regularization on the blendshape weights $E_2 = \lambda w$ with $\lambda = 1 \times 10^2$ set experimentally. In the first stage, we weight the landmarks around the mouth and lips more heavily and estimate only the jaw open parameter along with the rigid alignment. The next stage estimates all available jaw-related blendshape parameters using the same set of landmarks. The final stage estimates all available jaw and mouth-related blendshapes as well as the rigid alignment using all available landmarks. This process will also generally correct any overfitting introduced during the rigid alignment due to not being able to fully match the markers along the mouth.



(a) Render (b) Image

Figure 3. We trivially extend Sections 9.1 and 9.2 to handle landmarks from two cameras by simply combining the objective functions.

9.3. Optical Flow Infill

Consider, for example, Figure 2 where frames 1142 and 1146 were solved for successfully and we wish to fill frames 1143, 1144, and 1145. We visualize the optical flow fields using the coloring scheme of [1]. We adopt our proposed approach from Section 7 whereby the parameters of frames 1143, 1144, and 1145 are first solved for sequentially starting from frame 1142. Then, the frames are solved again in reverse order starting from frame 1146. This back-and-forth process which can be repeated multiple times ensures that the infilled frames at the end of the sequence have not accu-

mulated so much error that they no longer match the other known frame.

Using optical flow information is preferable to using simple interpolation as it is able to more accurately capture any nonlinear motion in the captured images (*e.g.* the mouth staying open and then suddenly closing). We compare the results of our approach of using optical flow to using linear interpolation for t and w and spherical linear interpolation for θ in Figure 4.

9.4. Multi-Camera

Our approach can trivially be extended to multiple calibrated camera viewpoints as it only entails adding another duplicate set of energy terms to the nonlinear least squares objective function. We demonstrate the effectiveness of this approach by applying our approach from Sections 9.1 and 9.2 to the same performance captured using an identical ARRI Alexa XT Studio from another viewpoint. See Figure 3.

We also compare the rigid alignment estimated by our automatic method to the rigid alignment created by a skilled matchmove artist for the same performance. The manual rigid alignment was performed by tracking the painted black dots on the face along with other manually tracked facial features. In comparison, our rigid alignment was done using only the markers detected by 3D-FAN on both the captured images and the synthetic renders. Our approach using only features detected by 3D-FAN produces visually comparable results. In Figure 5, we assume the manually done rigid alignment is the "ground truth" and quantitatively evaluate the rigid alignment computed by the monocular and stereo solves. Both the monocular and stereo solves are able to recover similar rotation parameters, and the stereo solve is able to much more accurately determine the rigid translation. We note, however, that it is unlikely that the manually done rigid alignment can be considered "ground truth" as it more than likely contains errors as well.

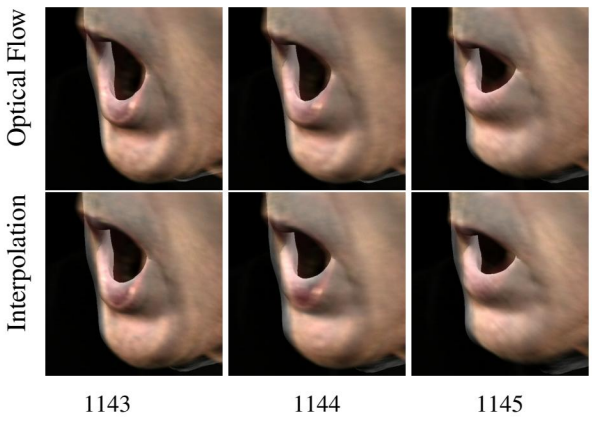


Figure 4. Using optical flow to interpolate missing frames produces results that better match the plate than simple linear interpolation. Notice how the lips and the face boundary match better in the optical flow results (particularly in frame 1144) than in the simple interpolation results.

9.5. Temporal Refinement

As seen in the supplementary video, the facial pose and expression estimations are generally temporally inconsistent. We adopt our proposed approach from Section 8. This attempts to mimic the captured temporal performance which not only helps to better match the synthetic render to the captured image but also introduces temporal consistency between renders. While this is theoretically susceptible to noise in the optical flow field, we did not find this to be a problem.

10. Conclusion and Future Work

We have proposed and demonstrated the efficacy of a fully automatic pipeline for estimating facial pose and expression using pre-trained deep networks as the objective functions in traditional nonlinear optimization. Such an approach is advantageous as it removes the subjectivity and inconsistency of the artist. Our approach heavily depends upon the robustness of the face detector and the facial alignment networks, and any failures in those cause the optimization to fail. Currently, we use optical flow to fix such problematic frames, and we leave exploring methods to automatically avoid problematic areas of the search space for future work. Furthermore, as the quality of these networks improve, our proposed approach would similarly benefit, leading to higher-fidelity results. While we have only explored using pre-trained facial alignment and optical flow networks, using other types of networks (*e.g.* face segmentation, face recognition, etc.) and using networks trained specifically on the vast repository of data from decades of visual effects work are exciting avenues for future work.

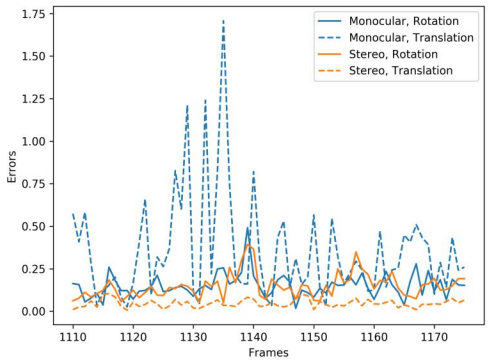


Figure 5. Assuming the manually done rigid alignment is the "ground truth," we measure the errors for rigid parameters for the monocular and stereo case.

11. Contributions

Work for this project was divided into the two deep networks used for our optimization: 3D-FAN and FlowNet2. Jane worked on writing the optimization solver for rigid alignment/expression estimation using the 3D-FAN landmark detection network, and David worked on writing the optimization solver for in-fill and temporal refinement using the FlowNet2 optical flow network. The full paper for this project can be found at: `https://arxiv.org/abs/1812.02899` and supplementary video is at: `https://drive.google.com/file/d/1QluEeq3N6JkrkgCNlAhJMIPVdmV2unYG/view?usp=sharing`. Source code specific to this project can be found at: `https://github.com/janehwu/cs230`.

We would like to thank Michael Bao and Professor Ron Fedkiw for their guidance on this project.

References

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[2] A. Bulat and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, volume 1, page 4, 2017.

[3] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625. Springer, 2012.

[4] J. Deng, Y. Zhou, S. Cheng, and S. Zaferiou. Cascade multi-view hourglass model for robust 3d face alignment. In *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*, pages 399–403. IEEE, 2018.

[5] X. Dong, S.-I. Yu, X. Weng, S.-E. Wei, Y. Yang, and Y. Sheikh. Supervision-by-registration: An unsupervised approach to improve the precision of facial landmark detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 360–368, 2018.

[6] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[9] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[10] A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos. Large pose 3d face reconstruction from a single image via direct volumetric cnn regression. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1031–1039. IEEE, 2017.

[11] L. A. Jeni, J. F. Cohn, and T. Kanade. Dense 3d face alignment from 2d video for real-time use. *Image and Vision Computing*, 58:13–24, 2017.

[12] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[13] A. Jourabloo and X. Liu. Pose-invariant face alignment via cnn-based dense 3d model fitting. *International Journal of Computer Vision*, 124(2):187–203, 2017.

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

[15] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. H. Pighin, and Z. Deng. Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports)*, 1(8), 2014.

[16] M. Loper. Chumpy autodifferentation library. `http://chumpy.org`, 2014.

[17] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014.

[18] M. Lourakis and A. A. Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1526–1531. IEEE, 2005.

[19] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[20] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[21] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[23] F. Reda, R. Pottorff, J. Barker, and B. Catanzaro. flownet2-pytorch: Pytorch implementation of flownet 2.0: Evolution of optical flow estimation with deep networks. `https://github.com/NVIDIA/flownet2-pytorch`, 2017.

[24] O. Sendik and D. Cohen-Or. Deep correlations for texture synthesis. *ACM Transactions on Graphics (TOG)*, 36(5):161, 2017.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei. Integral human pose regression. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[27] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[28] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 18, 2018.

[29] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[30] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.

[31] J. Xing, Z. Niu, J. Huang, W. Hu, X. Zhou, and S. Yan. Towards robust and accurate multi-view and partially-occluded face alignment. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):987–1001, 2018.

[32] A. Zadeh, Y. C. Lim, T. Baltrusaitis, and L.-P. Morency. Convolutional experts constrained local model for 3d facial landmark detection. In *ICCV Workshops*, pages 2519–2528, 2017.