Realistic Retargeting of Facial Video

Wolfgang Paier
Fraunhofer HHI
wolfgang.paier
@hhi.fraunhofer.de

Markus Kettern
Fraunhofer HHI
markus.kettern
@hhi.fraunhofer.de

Peter Eisert
Fraunhofer HHI/
Humboldt University Berlin
peter.eisert
@hhi.fraunhofer.de

ABSTRACT

We propose a simple method for realistic retargeting of facial performance from one shot to another. Editors can combine different takes of a shot into one single, optimal take with minimal manual labour and highly realistic results. Using a static proxy mesh of the actor's head, we obtain approximate 3D information of recorded monocular facial video. This 3D information is used to create pose-invariant textures from recorded facial action and to re-render it into a target shot. This can be done for the full face or parts of it, allowing for flexible editing.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Animations; I.4.8 [**Scene Analysis**]: Tracking

Keywords

tracking, geometric proxy, facial texture, facial animation

CVMP '14, November 13 - 14 2014, London, United Kingdom

ACM 978-1-4503-3185-2/14/11...$15.00
http://dx.doi.org/10.1145/2668904.2668935

1. INTRODUCTION

Human faces are one of the most important carriers of emotional as well as any other type of information in movies and videos. Also, viewers are extremely good at noticing artefacts in facial video. Therefore, special care has to be taken when recording and editing facial shots. We present a method for realistic retargeting of facial movie sequences for either the full face or parts of it. This method enables the editor to

- Exchange parts of a sequence between different takes, e.g. a line of text with a mistake that flawed an otherwise very good take
- Change the timing of facial action, e.g. when the actor completed some action and afterwards began talking too early
- Correct unwanted action for parts of the face, e.g. removing or adding an eye blink or a smile

We use time-consistent 3D information to enable correct display and perspective distortion when replacing the facial action recorded in one shot with action from a different shot. This information is obtained by tracking a pre-created 3D proxy model of the actor's head through all recorded sequences. The creation of this proxy mesh can be done in several ways, we use a single shot of the actor in a multi-view still camera rig and an image-based 3D-reconstruction approach. Via this proxy model, the recorded shots can be transformed to animated texture maps. If available, multiple views can be seamlessly integrated into one texture map. This allows the reuse of a captured sequence even in shots where the face is viewed from an angle from which it has not been recorded. We also present the possibility to select sub-regions of the face for retargeting allowing to synthesise different shots into a novel one.

In order to create the synthesised shot, the animated texture maps of a source shot are used to render the proxy mesh at the positions where it has been tracked in the target shot. Since the region to be replaced, be it the whole face or parts of it, does not change its position in the texture regardless of what happens in the shot, it can easily be identified by drawing a mask onto a texture map. The rendered source texture is blended with the target shot along the edges of this mask. The colors of the rendered texture are adapted

Figure 1: Sample output of the presented animation system. Left: original video frame, right: with retargeted facial expression

to the colors of the target shot in each frame in order to account for changes in illumination between the two shots.

2. RELATED WORK

Facial performance capture is often used to make the performance of an actor transferable to situations where it has not been recorded. There is a rich body of research on this topic which is divided into marker-based and marker-less approaches. A good survey can be found in [1]. This paper is related to marker-less facial motion capture [2, 3, 4, 5, 6] insofar as it uses related techniques to accurately track the head position in the video footage. For example, [3] uses optical flow together with a photogrammetric reconstruction to drive a cyberscan model of the actor's face in neutral expression in order to obtain a time-consistent animation mesh. The lack of detail in this smooth mesh is compensated by extracting a bump map from a very high resolution laser scan which adds wrinkles and skin pores. Recently, [7] have proposed a method of capturing a detailed dynamic face model from monocular video that only needs manual intervention to match a detailed 3D-scan of the actor's face in rest to a neutral blend shape model. From this, a personalized blend shape model is created which is being tracked by a combination of a state-of-the-art sparse facial feature tracker and optical flow. After extraction of the blend shape parameters, illumination is estimated and a shape-from-shading method is used to add details to the smooth tracking mesh.

However, most of these approaches are used to drive animatable CG characters of differing levels of realism. That is, the goal is to match the facial expression of an avatar to the expression of a recorded face. Creating and animating fully realistic CG characters is possible as has been shown e.g. by the Digital Emily Project [8] but requires highly sophisticated capturing of an animatable model, both technically and in terms of human labour. In contrast, the approach presented in this paper achieves realism by using approximated geometry to re-render textures that convey the needed expressivity and detail, as for example in [9]. Similar strategies are used in free-viewpoint video techniques [10, 11, 12] where the images captured in a multi-view video camera setup are combined to create realistic renderings of novel viewpoints. The distortions to introduce for changing the viewing angle are obtained via a coarse approximation of the actor's geometry (e.g. the visual hull) and minimized by determining the cameras closest to the viewing position for texture generation during rendering.

3. CONNECTING 3D TO MONOCULAR VIDEO

Time-consistent 3D or depth information is the basis of our approach to sequence retargeting. In order to obtain this information for a video sequence, we use a pre-generated, detailed static proxy mesh which is registered to an initial key-frame and tracked in subsequent frames. We create this proxy mesh using a state-of-the-art image-based 3D reconstruction scheme [13] which provides high-resolution 3D data for the complete face as well as a reasonable approximation of the surrounding hair. This allows us to choose the seams for texture exchange more freely than when using a standard model restricted to an arbitrary crop of the face. See figure 2 for some example reconstruction results.

For the purpose of registering the proxy mesh to a single video key-frame, a set of 30 possible landmarks (e.g. eye corner, nose tip, mouth corner, ear lap, ...) was defined in advance. This allows to initialize the tracking, by selecting some landmark positions both on the mesh as well as in the video image.

The landmarks on the proxy mesh are static and need to be selected only once. However, the landmarks in the video need to be selected for each camera, therefore a graphical user interface was developed which greatly eases the initialization step (see fig. 3).

On the mesh, these landmark positions are represented by barycentric 3D-coordinates based on the vertices of the triangle containing the landmark. Since the camera positions are unknown, one rigid transformation per camera, representing the position of the mesh in the camera coordinate frame, is calculated. The computation is done separately for each camera by a nonlinear minimization of the squared sum of reprojection errors

$$\mathcal{E}_{\text{reproj}} = \sum_{\mathbf{x} \in L} \|\mathbf{x} - \text{proj}\left(\mathbf{R}\mathbf{X} + \mathbf{T}\right)\|^2 \qquad (1)$$

over the parameters of a rotation R and a translation vector $\mathbf{T}$. The pinhole camera model is used for projection and its

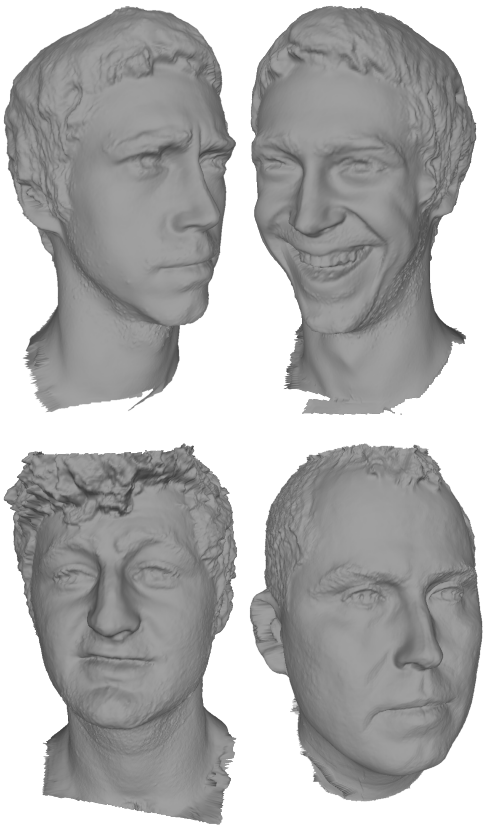

Figure 2: Examples of 3D reconstruction results

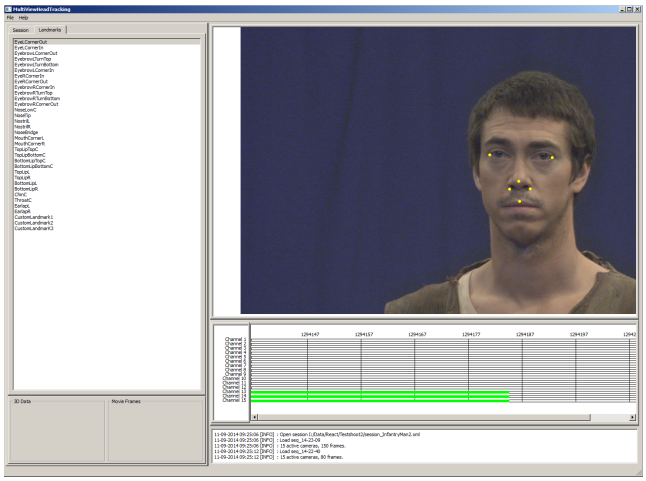

Figure 3: Screenshot of the graphical user interface to ease the manual selection of landmarks. A landmark is set by selecting it in the list of possible landmarks on the left side of the window and clicking on the corresponding position in the video image. Existing video streams are represented as green bars below the main image.

Figure 4: Initial matching of a 3D reconstructed model to a video key-frame. Left: original video, middle: texture overlay, right: diffference image

projection function proj $(\cdot)$ is given by

$$\text{proj}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \mathbf{c} - \begin{bmatrix} f_x \\ f_y \end{bmatrix} \circ \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix} \tag{2}$$

with $\mathbf{c}$ denoting the principal point and $\mathbf{f} = [f_x, f_y]^T$ representing the scaled focal lengths in x and y directions. Experience shows that when using reasonably accurate annotations of the movie frames, 5 landmarks are enough for a registration serving the purposes of this paper. Figure 4 shows a rendered overlay of the 3D-reconstruction texture over a video frame together with a difference image.

Once the proxy mesh is registered to a key-frame I_0, we use rigid model tracking based on optical flow to infer its position in subsequent frames I_k. To prevent texture drift, we do not use the optical flow between subsequent images I_k, I_{k+1} but rather between a distorted version of the key-frame I_0 and image I_{k+1}. The distortion reflects the positional update of the proxy between I_0 and I_k and the distorted image is created by image-based rendering.

Similar to [14], we use a linearized relation between Euclidean transformations applied to the 3D model and the 2D pixel displacements they impose on the rendered image. If we assume the rotational component of the transformation to be reasonably small, the rotation update to be applied to each vertex of the model may be approximated by a multiplication with the matrix

$$\mathbf{R}_{\text{lin}} = \begin{bmatrix} 1 & -r_z & r_y \\ r_z & 1 & -r_x \\ -r_y & r_x & 1 \end{bmatrix} \tag{3}$$

with r_x being the angle of rotation around the x-axis etc. The position of a vertex $\mathbf{X}$ of a mesh rotated around the coordinate origin by angles r_x, r_y, r_z and then translated by $\mathbf{T}$ is thus linearly approximated by

$$\mathbf{X}' \approx R_{\text{lin}}\mathbf{X} + \mathbf{T}. \tag{4}$$

A first order approximation of the projection of this displaced vertex into the image is given by

$$\mathbf{x}' = \mathbf{x} + \delta_{\mathbf{x}} \tag{5}$$

$$\delta_{\mathbf{x}} \approx \mathbf{J}_{\mathbf{X}}\left(\mathbf{X}' - \mathbf{X}\right) \tag{6}$$

where $J_{\mathbf{X}}$ is the Jacobian of proj$(\cdot)$ and is given by

$$\mathbf{J_X} = \begin{bmatrix} -f_x \frac{1}{z} & 0 & f_x \frac{x}{z^2} \\ 0 & -f_y \frac{1}{z} & f_y \frac{y}{z^2} \end{bmatrix}. \quad (7)$$

After some arithmetic manipulations, substituting (4) into (6) yields the displacement of the image point

$$\delta_{\mathbf{x}} \approx \frac{1}{z}\mathbf{f} \circ \mathbf{H} \quad (8)$$

with

$$\mathbf{H} = \begin{bmatrix} -r_y z + r_z y - t_x + r_x \frac{xy}{z} + r_y \frac{x^2}{z} + t_z \frac{x}{z} \\ -r_z x + r_x z - t_y + r_y \frac{xy}{z} - r_x \frac{y^2}{z} - t_z \frac{y}{z} \end{bmatrix} \quad (9)$$

$$= \frac{1}{z}\mathbf{f} \circ \mathbf{C_X} \begin{bmatrix} \mathbf{R} \\ \mathbf{T} \end{bmatrix} \quad (10)$$

$$\mathbf{C_X} = \begin{bmatrix} \frac{xy}{z} & -z + \frac{x^2}{z} & y & -1 & 0 & \frac{x}{z} \\ z - \frac{y^2}{z} & \frac{xy}{z} & -x & 0 & -1 & -\frac{y}{z} \end{bmatrix} \quad (11)$$

which is linear in all unknowns, namely $\mathbf{R} = [r_x\, r_y\, r_z]^T$ and $\mathbf{T}$. The offset $\delta_{\mathbf{x}}$ can be used to minimize the optical flow error between the rendered frame $\hat{I}_k$ and the following frame I_{k+1} defined by

$$\mathcal{E}_{\text{flow}} = \sum_{\mathbf{x} \in I} \left(\left(\nabla \hat{I}_k (\mathbf{x}) \right)^T \delta_{\mathbf{x}} - \left(I_{k+1} (\mathbf{x}) - \hat{I}_k (\mathbf{x}) \right) \right)^2 \quad (12)$$

with $\nabla \hat{I}_k (\mathbf{x})$ being the image gradient of $\hat{I}_k$ at pixel $\mathbf{x}$.

Defined over the area Ω of the rendered 3D model, this error can be minimized directly by solving an overdetermined system of linear equations

$$\mathbf{A} \begin{bmatrix} \mathbf{R} \\ \mathbf{T} \end{bmatrix} = \left(I_{k+1} (\Omega) - \hat{I}_k (\Omega) \right) \quad (13)$$

where $I(\Omega)$ is the vector containing all pixel intensities of image I in Ω. From equation (10), each pair of rows in A depending on a point $\mathbf{X}$ is given by

$$\mathbf{A_X} = \frac{1}{z} \nabla \hat{I}_k (\Omega) (\mathbf{f} \circ C_{\mathbf{X}}) \quad (14)$$

where $\nabla \hat{I}_k (\Omega)$ is the $2 \times |\Omega|$-matrix containing the x and y gradients of $\hat{I}_k$ in region Ω and the rows of $\mathbf{A}$ being dependent on the 3D points projected to the individual pixels in Ω.

This simple optical flow-based image correspondence calculation represents one step of a Gauss-Newton optimization and is carried out iteratively until the updates obtained are reasonably small. Since traditional optical flow can only handle small disparities, the optimization is performed in a hierarchical coarse to fine manner.

4. MULTIVIEW TEXTURE EXTRACTION

All camera views of one time frame are integrated into a single texture, compressing a synchronized multi camera video stream into a single stream of textures. As a representation of the multiview video stream, this has several advantages: it is more memory efficient than the source videos since redundant texture information is removed, it can be easily integrated into common rendering pipelines and, being temporally consistent, it allows the synthesis of the captured data plus changes in viewpoint.

Figure 5: Illustration of the quality measure. Top row: original video frames, bottom row: quality measure rendered on the tracked proxy model

The fusion is performed by projecting each camera image into a common texture space using the tracked 3D proxy and previously generated texture coordinates. We use only one source image per triangle to avoid blur or ghosting artefacts caused by the unavoidable geometric inaccuracies when using a static mesh for tracking a deformable surface like the face. To generate a texture with high details, a quality measure $\mathcal{W}(f_i, l_i)$ is employed that relates to the textural details provided by each camera image. In general, the textural detail depends mostly on viewing direction and distance between triangle f_i and camera l_i. A camera that provides more details because of a shorter distance or better viewing angle should be used more likely than a camera that is far away or is looking from an acute angle. However, since the distance between actor and cameras does not vary considerably in our setup, we restrict our quality measure to the dot product between viewing direction $\mathbf{v}$ and surface normal $\mathbf{n}$, see figure 5.

$$\mathcal{W}(f_i, l_i) = max(0, -\mathbf{n}^{\mathbf{T}}{}_{f_i} \mathbf{v}_{l_i}) \quad (15)$$

More sophisticated variants of $\mathcal{W}$ can be found in [15, 11, 16, 17]. Furthermore, a visibility test is performed and occluded or partially visible triangles receive a zero weight in the corresponding camera view. The texture is then constructed by sampling each texel in its corresponding source camera.

The task of selecting one source image for each triangle in a mesh can be conveniently formulated as discrete optimization problem. Using the quality measure (15), we define an energy term $\mathcal{D}(f_i, l_i)$ that ranges from 0 to 1 if the triangle f_i is visible and is set to ∞ if the triangle is fully or partially occluded.

$$\mathcal{D}(f_i, l_i) = \begin{cases} 1 - \mathcal{W}(f_i, l_i) & f_i \text{ is visible} \\ \infty & f_i \text{ is occluded} \end{cases} \quad (16)$$

Equation (17) describes the objective function that optimizes the textural detail in the resulting texture mosaic

$$\mathcal{E}_{\text{texA}}(L) = \sum_{i}^{n} \mathcal{D}(f_i, l_i) \quad (17)$$

where $L = \{l_1, ..., l_n\}$ is a label vector that assigns each triangle f_i an image I_{l_i} as texture source. The solution of (17) can be found by minimizing $\mathcal{D}(f_i, l_i)$ for each f_i separately. However, treating all triangles independently leads to a non-smooth solution that results in a high number of visible edges between adjacent triangles.

Figure 6: Texture mosaic (top row) and color coded labels (bottom row). Left: without smoothness-term, right: with smoothness-term. Blending was disabled for the generation of these images.

Therefore, a smoothness term $\mathcal{V}_{i,j}$ is added for each pair (f_i, f_j) of adjacent triangles. If both triangles have the same texture source, $\mathcal{V}_{i,j}$ is set to zero. If two adjacent triangles have different texture sources, $\mathcal{V}_{i,j}$ adds the sum of color differences $\Pi_{e_{i,j}}$ along the common edge $e_{i,j}$ of both triangles f_i and f_j to the overall objective function.

$$\Pi_{e_{i,j}} = \int_{e_{i,j}} \left\| I_{l_i}(x) - I_{l_j}(x) \right\| dx \quad (18)$$

$$\mathcal{V}_{i,j}(l_i, l_j) = \begin{cases} 0 & l_i = l_j \\ \Pi_{e_{i,j}} & l_i \neq l_j \end{cases} \quad (19)$$

$$\mathcal{E}_{\text{texB}}(L) = \sum_{i}^{N} \mathcal{D}(f_i, l_i) + \lambda \sum_{i,j \in \mathcal{N}} \mathcal{V}_{i,j}(l_i, l_j) \quad (20)$$

This way, seams are more likely to occur in regions where adjacent triangles look similar, making them less visible (see figure 6). Minimizing (20) yields a label vector $L = \{l_1, ..., l_n\}$ that contains the optimal source image for each triangle in terms of spatial resolution and visibility of seams for a single texture. However, for video based facial animation it is also necessary to consider temporal consistency of the generated texture mosaics in order to avoid temporal artefacts, like a constantly changing source camera or a jittering seam. In order to prevent temporal artifacts, we also add a temporal smoothness term that penalizes changing the source camera of a triangle between two consecutive time steps.

$$\mathcal{T}(l_i^t, l_i^{t-t}) = \begin{cases} 0 & l_i^t = l_i^{t-t} \\ 1 & l_i^t \neq l_i^{t-t} \end{cases} \quad (21)$$

$$\mathcal{E}_{\text{tex}}(L) = \sum_{t}^{T} \sum_{i}^{N} \mathcal{D}(f_i^t, l_i^t) + \eta \mathcal{T}(l_i^t, l_i^{t-1}) + \lambda \sum_{i,j \in \mathcal{N}} \mathcal{V}_{i,j}(l_i^t, l_j^t) \quad (22)$$

The optimization is performed simultaneously for each time step. Each triangle f_i has an additional temporal index t that allows f_i to have different source cameras in each time step t. η and λ are scalar weights that determine the impact of the spatial and temporal smoothness terms. Since objective functions similar to (20) or (22) are commonly used in computer vision (e.g. image mosaicing, segmentation, image restoration), there exist several optimization techniques (e.g. graph cuts, belief propagation or tree re-weighted message passing) that can be employed to solve (22). In general, (22) is an NP-hard problem and we use the alpha-expansion algorithm [18] to efficiently find a close-to-optimum approximative solution. This is done by transforming the original multi-label problem into a series of binary label problems that each can be solved optimally using s-t graph cuts.

For the final construction of a texture we perform a global color correction on the input images, which is explained in the next section. This is necessary to reduce the effect of different color responses among the source cameras. We also use a simplified version of seam leveling [17] to remove low frequency differences at seams. This technique adds piecewise continuous functions to the constructed texture mosaic, see figure 7.

These functions correspond to connected components of the mesh that are textured from the same source image. At seams, they are constrained to stay at a fixed value that minimizes the color difference along the seam (e.g. mean colour). All other values are obtained by a diffusion like approach, which distributes and smooths them based on the neighbourhood and mesh topology. To remove the remaining high frequency discrepancies, we use feathering with a

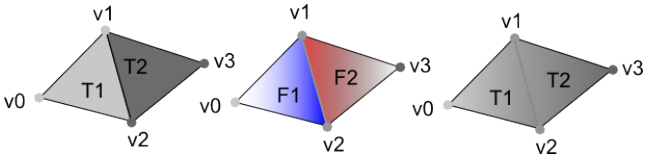

Figure 7: Left: 2 adjacent triangles with a visible seam textured from two different images T1 and T2, center: Levelling function (white=0, red > 0, blue < 0), right: adding the levelling function removed the visible seam

blending radius of approximately 20 pixels.

5. ANIMATION RETARGETING

After tracking and texture extraction, each captured scene is represented as a sequence of textures, rigid body transformations and the proxy head model. This representation enables us to create novel views by exchanging parts of the face between different sequences, change facial actions or to make an already good shot even better. Our animation retargeting approach consists of three tasks:

1. Selection of source and target sequence and manipulation mask
2. Re-rendering of the source sequence into the target video
3. Integration of the replaced parts by color correction and blending

The selection of replaced facial areas is very intuitive by simply masking the desired area. This can be done directly in texture space which eases the following processing steps, as it provides a consistent representation of the manipulated region over time. This is important as selection in texture space is not affected by rigid head motion.

However, depending on the parametrization of texture-space it can be a more convenient approach to mark areas to be replaced on the 3D-mesh itself in camera space, see figure 8. Then, the user does not need to take care if the modified area expands over multiple tiles in texture-space. A further advantage of the latter approach is that the user perceives an immediate feedback, since he/she sees which parts of the video are actually manipulated. The mesh-based selection can be easily transformed to texture space, for example by rendering each selected triangle into the texture map. After all regions have been marked, the selection is represented by a binary mask in texture space.

To perform a smooth integration of the re-rendered face, we use alpha blending which generates smooth transitions at the border between original and replaced regions in the video footage. Therefore, we generate a smooth weight map $\mathcal{W}$ in texture space using a distance transform [19] $\mathcal{D}_{i,j}$ of the binary selection mask

$$\mathcal{W}_{i,j} = 1 - min\left(1.0, \frac{\mathcal{D}_{i,j}(\mathbf{M})}{r}\right) \tag{23}$$

Figure 8: Highlighted selection of modified facial region. Top: in camera space, bottom: in texture space

with $\mathbf{M}$ being the binary selection mask and r being the blending radius. The weight map and the source texture are both rendered into the target video, where they are used to calculate the weighted average of the original video and the source video. Due to the convenient data representation, this step can be performed with currently available graphics APIs in a single rendering pass by simply adding the weight map to the extracted textures as alpha-channel.

However, simple alpha blending alone does not achieve a satisfactory integration, since changing light conditions or different camera settings during a capture session can cause obvious mismatches in color and/or brightness, see figure 9. Therefore, a color correction step is necessary that adapts the source texture to better fit the color characteristics of the target video frame. Such discrepancies are typically modelled by a low dimensional global color transform, since camera settings and light affect each pixel in a similar manner. For example, [20] presents a global method that transfers color statistics between images to make them look more similar.

Each image is transformed to the decorrelated CIE Lab color space and a color statistic $\mathcal{N}(\mu, \sigma^2)$ is calculated for each channel. The correction is then performed for the l, α and β component as follows:

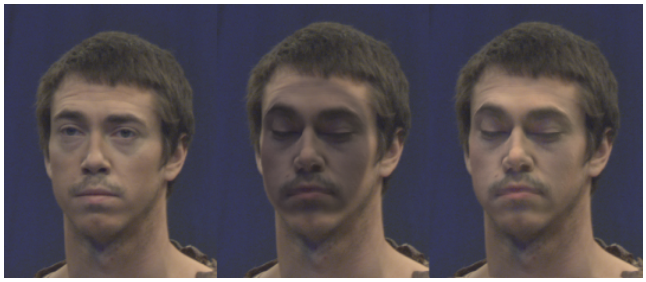

Figure 9: Left: Original frame, middle: modified frame without color correction, right: with color correction

$$\begin{aligned} c^*_{\text{src}} &= c_{\text{src}} - \mu_{\text{src}} \\ c_{\text{new}} &= c^*_{\text{src}} \frac{\sigma_{\text{dst}}}{\sigma_{\text{src}}} + \mu_{\text{dst}} \end{aligned} \tag{24}$$

with c corresponding to a single component of the pixel's color vector $\begin{bmatrix} l & \alpha & \beta \end{bmatrix}^T$ and μ and σ being the mean and the standard deviation of the corresponding channel. c_{new} is calculated by scaling the centered value of c and adding μ_{dst} to better fit the target color distribution $\mathcal{N}_{\text{dst}}(\mu, \sigma^2)$. Despite its simplicity, this method already provides great improvement, see figure 9.

However, in contrast to [20] we do not only have corresponding color distributions but also local correspondences, since our head model is tracked and every local region in the re-rendered face should correspond to the same part of the actor's face in the original video. This allows us to perform an improved color correction by re-writing the color matching scheme using a linear model:

$$\boldsymbol{\phi}(c) = \begin{bmatrix} 1 & c \end{bmatrix} \tag{25}$$

$$x_{\text{new}} = \boldsymbol{\phi}(x_{\text{src}}) * \mathbf{b}, \tag{26}$$

with $\boldsymbol{\phi}(c)$ being a feature vector and $\mathbf{b}$ being a set of linear coefficients. This corresponds directly to (24) as it has a constant offset and a scaling factor. On closer examination of the relation between each $\begin{bmatrix} l & \alpha & \beta \end{bmatrix}^T_{\text{src}}$ and its corresponding $\begin{bmatrix} l & \alpha & \beta \end{bmatrix}^T_{\text{dst}}$ we found that, in our dataset, l fits perfectly to the linear model. However, α and β can be explained better by a more complex model. Therefore we extend the feature vector $\boldsymbol{\phi}(c)$ by a squared term:

$$\boldsymbol{\phi}(c) = \begin{bmatrix} 1 & c & c^2 \end{bmatrix}^T \tag{27}$$

Now the color matching can also model a non linear relation between $\begin{bmatrix} l & \alpha & \beta \end{bmatrix}^T_{\text{src}}$ and $\begin{bmatrix} l & \alpha & \beta \end{bmatrix}^T_{\text{dst}}$ but it still can be solved using an ordinary linear least squares solver. To prevent unnecessarily high impact of c^2, we add a regularization term. Now, the objective function is given by

Figure 10: Improved color correction. Left: target video, middle: color correction by transfer of image statistics, right: our linear least squares approach

$$\mathcal{E}_{\text{col}}(\mathbf{b}) = \sum_{i=0}^{n} \left(\boldsymbol{\phi}\left(c^i_{src}\right)^T * \mathbf{b} - c^i_{dst} \right) + \mathbf{b}^T \mathbf{W} \mathbf{b} \tag{28}$$

$$\mathbf{W} = \begin{bmatrix} 0 & & \\ & 0 & \\ & & \lambda \end{bmatrix} \tag{29}$$

with λ being a scalar factor that determines the impact of the regularization. It is also possible to make the color correction more robust by rejecting corresponding color values if they exceed a certain threshold, or by using a robust estimator. Figure 10 shows the result of the improved color correction. The results are similar, but regions like the forehead and the ridge of the nose (first and third row), which seem to be too bright when using Reinhard's method look more realistic. The second row shows another example where the simple transfer of image statistics produces a slightly over saturated result. In most cases our color correction blends better into the background or produces at least equally good results.

6. RESULTS AND DISCUSSION

Experimental results

In this section, some results of the facial retargeting are presented. We use the video footage of three synchronized HD cameras, 1920x1080@30 fps. The cameras were not

Figure 11: Top: Final texture with seam optimization, color correction and blending enabled. Bottom: close up on the facial region

equally white balanced, which is an additional challenge. Each row of figure 12 shows the original video frame on the left and the synthesized video frame on the right side. The source video frames from which the new facial expression is synthesised, are located in the center. Figure 12 shows the good quality of our presented facial animation method. The produced results were evaluated visually, since the proposed method generates new content from existing video footage which makes it not easy to provide a proper groundtruth. A video containing further results like animated sequences can be found in the supplementary material: http://youtu.be/tTLytzbEZr0. The tracking and the geometric proxy are accurate enough to allow a realistic re-rendering of video sequences. The extracted textures have no noticeable seams (see figure 11) and even the different white balancing is handled by our global per-pixel color correction to allow for a seamless integration of new facial expressions. A non-optimized implementation of our system tracks up to 4 frames per seconds and texture extraction takes several seconds per frame. Re-rendering and blending can be directly implemented using standard graphic APIs and is therefore very fast.

Conclusion

We presented a method for realistic retargeting of facial video sequences. Our method uses existing video footage and allows the editor to modify the facial animation or expressions of an actor during post-production. This includes for example, changing the timing of a facial action, correcting or removing facial actions or importing parts of the face from a different take. Our method produces realistic results since all animations are taken from existing video footage and there is no need for manual animation of meshes. We use video footage of three synchronized full HD cameras and a pre-generated geometric proxy of the actor's head which is being tracked rigidly in each movie stream. Using the 3D geometry we generate one texture per frame that integrates the color information of all cameras. Now, facial action can be transferred from one video into another by rendering the textured proxy mesh into the target video. All animated facial regions are marked in texture space using a binary texture, since it is consistent over time. Finally, we use a global color correction and alpha blending to integrate the new facial animation.

Future work

The use of a static proxy mesh creates difficulties in shots where facial action involves strong deformations of the facial geometry. These inaccuracies become more visible when the viewpoint has to be changed in re-rendering. We aim at creating a deformable head model and track deformation parameters along with the head pose in video in order to overcome this issue. The integration of facial animations into a new video is currently done using simple alpha blending. A graph cut based approach could be used to find an optimal seam in a certain area around the determined border between the imported facial animation and the background video. We use a global color correction approach to make the retargeted video look more like the original video, however this does not consider illumination based intensity differences. Therefore, a reasonable extension of our current system would be an additional estimation of a local illumination model, to improve realism of the inserted textures, especially for scenes involving artistic lighting.

Acknowledgement

The work presented in this paper has been funded by the Seventh Framework Programme EU projects RE@CT (FP7-ICT-288369) and Reverie (FP7-ICT-287723).

7. REFERENCES

[1] F. Pighin and J. Lewis, "Facial motion retargeting," in *ACM SIGGRAPH Courses*, 2006.

[2] K. Li, Q. Dai, R. Wang, Y. Liu, F. Xu, and J. Wang, "A data-driven approach for facial expression retargeting in video," *IEEE Transactions on Multimedia*, vol. 16, pp. 299–310, 2014.

[3] G. Borshukov, D. Piponi, O. Larsen, J. P. Lewis, and C. Tempelaar-Lietz, "Universal capture - image-based facial animation for "the matrix reloaded"," in *ACM SIGGRAPH Courses*, 2005.

[4] J.-X. Chai, J. Xiao, and J. Hodgins, "Vision-based control of 3d facial animation," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003.

Figure 12: Left: original target video frame, center: source video frames, right: target video with retargeted facial expression

[5] D. Sibbing, M. Habbecke, and L. Kobbelt, "Markerless reconstruction and synthesis of dynamic facial expressions," *Computer Vision and Image Understanding*, vol. 115, no. 5, pp. 668–680, 2011.

[6] P. Eisert and B. Girod, "Analyzing facial expressions for virtual conferencing," 1998.

[7] P. Garrido, L. Valgaert, C. Wu, and C. Theobalt, "Reconstructing detailed dynamic face geometry from monocular video," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 158:1–158:10, 2013.

[8] O. Alexander, M. Rogers, W. Lambeth, M. Chiang, and P. Debevec, "Creating a photoreal digital actor: The digital emily project," in *European Conference on Visual Media Production (CVMP)*, London, UK, 2009.

[9] P. Eisert and J. Rurainsky, "Geometry-assisted image-based rendering for facial analysis and synthesis." *Sig. Proc.: Image Comm.*, vol. 21, no. 6, pp. 493–505, 2006.

[10] J. Kilner, J. Starck, and A. Hilton, "A comparative study of free-viewpoint video techniques for sports events," in *European Conference on Visual Media Production (CVMP)*, 2006.

[11] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," in *ACM SIGGRAPH*, 2003.

[12] C. Lipski, F. Klose, K. Ruhl, and M. Magnor, "Making of who cares hd stereoscopic free viewpoint video," in *European Conference on Visual Media Production (CVMP)*, 2011.

[13] D. Blumenthal-Barby and P. Eisert, "High-resolution depth for binocular image-based modelling," *Computers & Graphics*, vol. 39, pp. 89–100, 2014.

[14] P. Eisert and B. Girod, "Model-based 3d-motion estimation with illumination compensation," in *Image Processing and Its Applications*, vol. 1, 1997, pp. 194–198 vol.1.

[15] Z. Janko and J.-P. Pons, "Spatio-Temporal Image-Based Texture Atlases for Dynamic 3-D Models," in *IEEE International Workshop on 3-D Digital Imaging and Modeling*, 2009.

[16] C. Allene, J.-P. Pons, and R. Keriven, "Seamless image-based texture atlases using multi-band blending," in *ICPR*. IEEE, 2008.

[17] V. Lempitsky and D. Ivanov, "Seamless Mosaicing of Image-Based Texture Maps," in *CVPR*, 2007.

[18] Y. Boykov and V. Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[19] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," Cornell Computing and Information Science, Tech. Rep., 2004.

[20] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics Applications*, vol. 21, no. 5, pp. 34–41, 2001.