EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Real-Time Multi-View Facial Capture with Synthetic Training

Martin Klaudiny[† 1], Steven McDonagh [† 1], Derek Bradley[1], Thabo Beeler[1], and Kenny Mitchell[‡ 1,2]

[1]Disney Research [2]Edinburgh Napier University

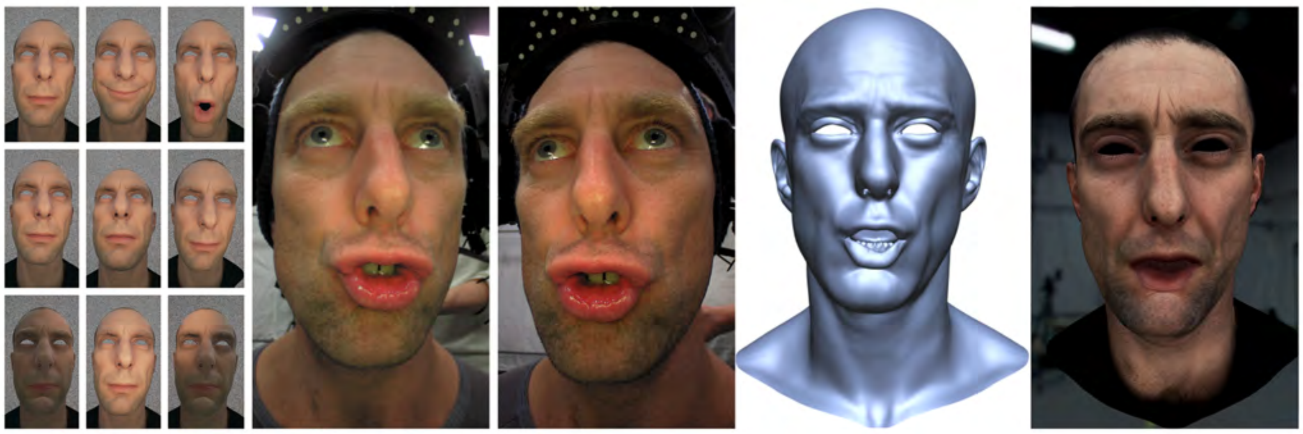

Figure 1: We propose a real-time multi-view face tracking method for high-quality on-set facial performance capture. Using rendered synthetic training (left most) we learn an actor-specific regression function, which tracks the actor's face online using multiple cameras (middle left). An reconstructed expression is depicted (middle right) together a digital-double face rig within a game engine (right).

Abstract

We present a real-time multi-view facial capture system facilitated by synthetic training imagery. Our method is able to achieve high-quality markerless facial performance capture in real-time from multi-view helmet camera data, employing an actor specific regressor. The regressor training is tailored to specified actor appearance and we further condition it for the expected illumination conditions and the physical capture rig by generating the training data synthetically. In order to leverage the information present in live imagery, which is typically provided by multiple cameras, we propose a novel multi-view regression algorithm that uses multi-dimensional random ferns. We show that higher quality can be achieved by regressing on multiple video streams than previous approaches that were designed to operate on only a single view. Furthermore, we evaluate possible camera placements and propose a novel camera configuration that allows to mount cameras outside the field of view of the actor, which is very beneficial as the cameras are then less of a distraction for the actor and allow for an unobstructed line of sight to the director and other actors. Our new real-time facial capture approach has immediate application in on-set virtual production, in particular with the ever-growing demand for motion-captured facial animation in visual effects and video games.

1. Introduction

As feature films and video games require more and more high-quality visual effects, the demand for realistic virtual character animation has never been higher. Ever since iconic films such as *Avatar* and *Lord Of The Rings* relied heavily on motion-capture for digital characters, performance-based animation, particularly for faces, has become the regular approach for bringing virtual characters to life. Today, nearly every production that requires an expressive digital actor employs some form of facial performance capture, and this trend is unlikely to change in the foreseeable future.

The approach typically starts by placing tens or even hundreds of markers on the face of the actor, then mounting a fixed rig of cam-

† Joint 1st authors. [steve.g.mcdonagh,mklaudiny]@gmail.com
‡ email: kenny.mitchell@disneyresearch.com

eras to a helmet that is tightly affixed to the actor's head, providing (mostly) stabilized close-up video footage of the actor's expressions while they move (relatively) freely around a set. The markers are then tracked in the one or more video streams, yielding sparse point constraints for deforming a pre-built higher-resolution facial rig of the actor in order to retrieve the performed animation, which can then be transferred, for example, to a fantasy creature. This approach has become standard in production, with minor variations in marker placement and number of helmet cameras. But there are several avenues for improving on-set facial performance capture, both in terms of workflow as well as quality.

The most limiting aspect for film-makers at the moment is that tracking and reconstruction of the performance typically occurs offline. This prevents directors from inspecting the digital performance while shooting and hence makes directing much more challenging as the director has to rely solely on his own imagination of the final performance. The idea of *real-time virtual production* is a growing trend in entertainment production with the intriguing promise to overcome this limitation by giving the director, at the very least, a real-time preview of the final performance. While the research community has recently seen a growing number of real-time monocular facial tracking methods, so far these have not been extended to multi-camera rigs, or applied on the lower quality footage that is typical of helmet cameras, nor have they been evaluated under the variable conditions that occur during production.

An additional limitation of on-set performance capture is the cumbersome helmet and camera rig that actors are currently required to wear. While variations exist in terms of the number and position of cameras, all rigs place the cameras directly in front of the actor's face, pointing inwards, approximately 20-40 cm away. These camera rigs block the line of sight of the actor and occlude the face from external points of view. They also prevent actors from interacting with props (e.g. drinking a cup of tea), performing in close contact with each other (e.g. in a kissing scene) or acting in tight spaces.

Further still is the problem of *camera shake*. Hinted at earlier, the helmet cameras provide only mostly stabilized video footage, but fast head motion and extreme expressions can easily cause unwanted camera jitter or sliding relative to the head. Tracking algorithms can confuse these movements with a facial expression change. Illumination changes as the actor moves around the set is another very common problem. Any image-based tracker that relies on pixel intensity consistency can easily fail in such cases. And on top of all of that, traditional marker-based trackers do not capture a dense representation of the face, and so high-resolution actor-specific details are often missed.

In this work we alleviate these problems, by proposing a new real-time markerless facial capture algorithm, which estimates dense facial performances from multi-view input videos using regression. Building on state-of-the-art monocular techniques, our primary contribution is a new multi-view framework for real-time regression, which improves reconstruction accuracy by resolving depth ambiguities. Our framework can scale seamlessly from one to many views, making our approach independent of the camera rig configuration. Furthermore, we show that our method allows the placement of cameras in side-view configurations in contrast to direct frontal placement. This improves actor line of sight and flexibility while reducing face occlusion.

We adopt the approach of generating synthetic imagery for our regression training [MKB*16], which ensures robustness to illumination changes and camera shake without requiring the capture and manual annotation of large numbers of facial images. We are able to explicitly train the real-time tracker on synthesized facial renderings from arbitrary synthetic cameras that witness the actor undergoing a wide range of facial expressions from varying viewpoints and under a range of possible on-set lighting conditions.

Our method is flexible and can be applied in different scenarios. In addition to on-set helmet-camera capture, we demonstrate our technique on multi-view high-quality performance capture setups, where our method can provide a reliable real-time preview. To validate our method we show several examples on real-world data, compare to state-of-the-art tracking frameworks and further evaluate performance with a quantitative analysis on synthetic imagery.

In summary, the main contributions of our work are:

1. We generalise a single-view real-time face tracking technique by introducing a novel multi-view regression framework based on multi-dimensional random ferns, which can seamlessly integrate an arbitrary number of viewpoints to obtain high quality tracking;
2. We showcase how the flexibility of synthetic imagery training [MKB*16] can be utilized in conjunction with markerless multi-view tracking to achieve high-quality facial performance capture for virtual production, robust to changes in incident illumination and camera shake;
3. We demonstrate our method in various scenarios and compare with state-of-the-art techniques, including helmet-based on-set performance capture where, for the first time, we allow to place cameras at the side of the face, improving the field of view of the actor while reducing occlusions from the point of view of the director.

As a result, this work represents a large step forward in real-time on-set facial performance capture for film and video game productions.

2. Related work

Facial performance capture continues to be a primary research topic in computer graphics, because the current demand for facial motion capture in visual effects and games is at an all-time high, and industry practitioners seek higher-fidelity, more actor-friendly and faster capture methods. Traditional offline approaches have come a long way and can deliver markerless reconstruction at near-perfect quality [BHPS10, BHB*11, KH12, VWB*12, GVWT13, SWTC14, FJA*14]. However, lengthy reconstruction times make these methods unsuitable for real-time feedback, for example on film sets and virtual production scenarios.

A more recent trend is real-time facial capture, which tackles the problem of reconstruction time, but typically at the cost of reconstruction quality [CXH03, WLVGP09, WBLP11, CWLZ13, BWP13, LYYB13, CHZ14, WCHZ14, CBZB15, HMYL15, SLL16].

Here, a common approach is to utilize machine learning, leveraging large training databases (e.g. FaceWarehouse [CWZ*14]) and then explaining new facial imagery by the best fitting 3D model, for example in a regression framework [CWLZ13, CHZ14]. This approach has the advantage of being fast, since the algorithms are typically based on simple decision trees. On the other hand, reconstruction results tend to be fairly generic and often lack person-specific details. Cao et al [CBZB15] aim to remedy this problem by regressing also person-specific wrinkle geometry, however results are still not at the accuracy of offline methods. Still, lower-resolution real-time facial capture can be used in a variety of interesting scenarios, for example real-time expression transfer for re-enactment [TZN*15, TZS*16] and facial capture for virtual reality [LTO*15, OLSL16].

One approach to obtain higher fidelity in regression-based methods is to train the system on exactly the type of data that will be observed at runtime, for example actor-specific facial expressions, synthetically rendered from different viewpoints and under different lighting conditions [RQH*12, FDG*12, JCK15, FHK*15, MKB*16]. This is a promising avenue for real-time virtual production scenarios, as film sets are created and actors are cast well before shooting, opening the door to such *a priori* synthetic training scenarios. In particular, in concurrent work [MKB*16] we have shown that learning a synthetic high-resolution actor-specific prior that is robust to illumination changes can largely improve the fidelity of real-time monocular performance capture. However, so far the extension of such techniques to multiple views, typical of commodity on-set facial capture rigs, is not trivial and has been thus far under-explored.

To our knowledge, regression-based tracking methods have not been explored for facial capture using multiple cameras. In a wider space, multi-view regression typically relates to multi-modal data, where different types of features are combined to learn a single regression function [KF07]. This can be used for general learning of complicated dependencies in large multi-modal datasets from various domains [ZZ11, ZCD*15] or for object/category recognition in computer vision [SRAG15]. Haopeng et al. [HZ14] use kernel regression on multiple views for object recognition and their pose estimation. De Campos et al. [dCM06] show that multi-camera regression can be used for hand pose estimation. However, they simply stack custom hand feature descriptors into a single large vector. A similar concept was used by Zhao et al. [ZFN*10] for human pose estimation.

We present the first method for real-time facial performance capture that is specifically designed for multiple video inputs. To this end, we propose a novel multi-view regression method based on multi-dimensional random ferns. Following recent insights in synthetic prior design [MKB*16], we construct synthetic training sets for multi-camera scenarios where we sample across the planes of expression, camera viewpoint, and illumination. This allows us to evaluate different possible camera placements, and identify configurations best suited for helmet camera rigs, which are typically used in virtual productions [BGY*13].

3. Method overview

Our approach for real-time face tracking estimates at every frame a model state which describes a facial expression and a pose of a camera rig with respect to the actor's head. Facial expressions are defined by controls of an actor-specific animation rig based on blend shapes. The method uses a new regression framework, which learns a mapping between images captured by *multiple cameras* and the model state during a training stage. Once the regression function is learnt, model states can be computed in real-time from live image streams. Fig. 2 depicts a high-level workflow consisting of an offline training stage and an online tracking stage.

The training stage for the regression (Section 5) uses *synthetic* facial imagery, in contrast to real data used in previous work [CWLZ13, WCHZ14, CHZ14, CBZB15]. This provides accurate ground-truth model states for the corresponding images, and also makes it easy to vary conditions such as expression, illumination, and camera position, allowing to tailor the training to the actor, capture rig and environment.

The training set provides synthetic training pairs of multi-view images and corresponding model states, which are fed into a supervised learning process of transitions between different model states at consecutive frames (Section 4.1). We propose a cascaded regression scheme using novel *multi-dimensional random ferns*. Once trained, the actor-specific regressor estimates the model state in real-time, i.e. facial expression and camera rig pose relative to the head, and is thus suited for online tracking as described in Section 4.2.

4. Multi-view face regression

The regression framework is used to estimate in real-time the current facial expression and rigid model transform $\mathbf{T}$ of a camera rig C with respect to the head given a set of input images at one point in time. Facial expression and relative head pose are subsumed in the model state $\mathbf{S} = (\mathbf{a}, \mathbf{r}, \mathbf{t})$, where the transform $\mathbf{T}$ is separated into its translational ($\mathbf{t}$) and rotational ($\mathbf{r}$) components, and the rotation is encoded as a quaternion. The blend weights $\mathbf{a}$ define the 3D shape of a facial expression $B = B_0 + \sum_{j=1}^{|\mathbf{B}|} \mathbf{a}_j (B_j - B_0)$ using a blend shape rig $\mathbf{B}$. The vector $\mathbf{S}$ of length $|\mathbf{B}| + 4 + 3$ represents the output variables of the regression. The input variables are sampled from the current set of facial images $\mathbf{I} = \{I_c\}_{c=1}^{|C|}$ captured from $|C|$ viewpoints, where $|C| \geq 1$. We assume that all cameras C are calibrated with respect to the world coordinate system of the blend shape rig. Each camera is described by an extrinsic rigid transform $\mathbf{T}_c$ and an intrinsic projection function P_c which can incorporate any lens distortion model. The model transform $\mathbf{T}$ is applied to all $\mathbf{T}_c$ to rigidly move the capture rig in relation to the head. For a helmet camera rig, this can model camera shake or changes in helmet positioning on the head. For static camera rigs, $\mathbf{T}$ can capture change of subject head pose.

4.1. Training

Regression training requires ground-truth examples of mappings between input multi-view images and a corresponding model state. In the case of synthetic training, a training pair $(\mathbf{I}_n, \hat{\mathbf{S}}_n)$ consists of

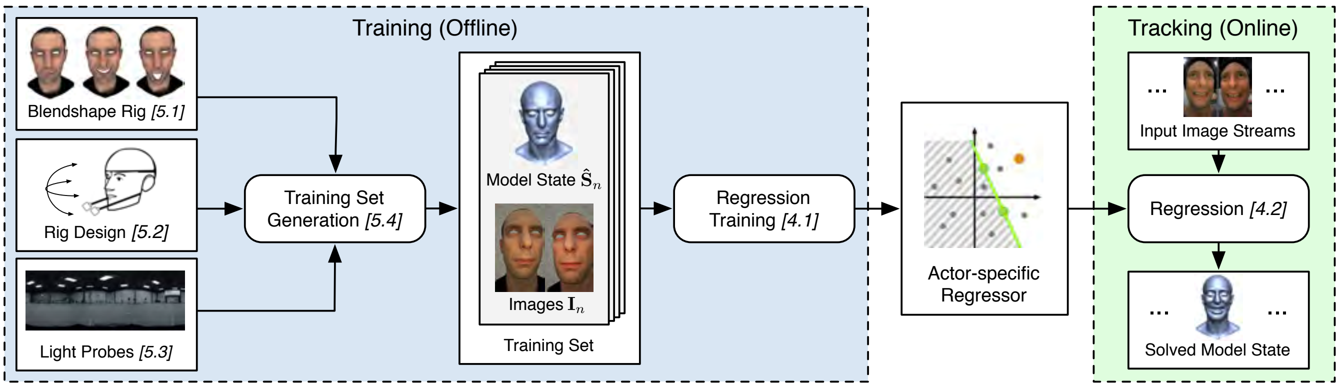

Figure 2: Overview of real-time multi-view regression framework for facial performance capture. We start by synthetically generating a person and environment specific training set with which we train a person-specific regressor. After offline training, the regressor can solve for model states in real-time from input images.

the rendered images $\mathbf{I}_n$ and the ground-truth state $\hat{\mathbf{S}}_n$ which was used to generate them. This process will be discussed in detail in Section 5. N training pairs are used as a basis for the final training set. The regression learns transitions between initial model states $\mathbf{S}_m$ and ground-truth target states $\hat{\mathbf{S}}_m$ given associated images $\mathbf{I}_m$. Therefore, every training pair is augmented with a set of initial states forming M final training samples $(\mathbf{I}_m, \hat{\mathbf{S}}_m, \mathbf{S}_m)$.

Our framework is based on the cascaded regression scheme introduced by Cao et al. [CWWS12] and subsequently used in similar applications [CWLZ13, WCHZ14, CHZ14]. This scheme consists of a sequence of T stage regressors R^t. Each R^t contains a sequential ensemble of F primitive regressors r^f. The primitive regressors have weak learning capability and they need to be combined into an ensemble to improve overall accuracy. The stage regressors R^t incrementally learn a regression function between input image features from $\mathbf{I}_m$ and state differences $(\hat{\mathbf{S}}_m - \mathbf{S}_m)$ across all training samples m. This can be formalized as a minimization

$$R^t = \operatorname*{argmin}_R \sum_{m=1}^{M} ||(\hat{\mathbf{S}}_m - \mathbf{S}_m^{t-1}) - R(\gamma^t(\mathbf{S}_m^{t-1}, \mathbf{I}_m))||, \qquad (1)$$

where $(\hat{\mathbf{S}}_m - \mathbf{S}_m^{t-1})$ is a current state residual to the targets which needs to be learned. The function $\gamma^t(\cdot)$ samples a set of input image features, collecting them in an *appearance vector* $\mathbf{g}_m^t$ for each stage. This sampling is based on the facial shape and pose $\mathbf{S}^{t-1}$ from the previous stage. After training each regressor R^t all samples are updated according to the learned portion of the regression function as follows

$$\mathbf{S}_m^t = \mathbf{S}_m^{t-1} + R^t(\gamma^t(\mathbf{S}_m^{t-1}, \mathbf{I}_m)). \qquad (2)$$

The starting point $\mathbf{S}_m^0$ equals the initial state $\mathbf{S}_m$. The ensemble of primitive regressors r^f forming R^t is trained in a similar fashion, with the important difference that the appearance vector $\mathbf{g}_m^t$ for each sample m is kept the same for all r^f. The learning target is a residual $\delta\hat{\mathbf{S}}_m = \hat{\mathbf{S}}_m - \mathbf{S}_m^{t-1}$ from Eq. 1. The training minimization is formulated as

$$r^f = \operatorname*{argmin}_r \sum_{m=1}^{M} ||(\delta\hat{\mathbf{S}}_m - \delta\mathbf{S}_m^{f-1}) - r^f(\phi^f(\mathbf{g}_m^t))||, \qquad (3)$$

where the function $\phi^f(\cdot)$ defines a feature selection from the appearance vector for the primitive regressor r^f. The residual $\delta\hat{\mathbf{S}}_m$ is sequentially learned from a starting point $\delta\mathbf{S}_m^0 = \mathbf{0}$ and the updates after each r^f are computed as

$$\delta\mathbf{S}_m^f = \delta\mathbf{S}_m^{f-1} + r^f(\phi^f(\mathbf{g}_m^t)). \qquad (4)$$

4.1.1. Multi-dimensional random ferns

Our primitive regressor r^f is a *random fern*. Previous methods [CWLZ13, WCHZ14, CHZ14] utilised random ferns for monocular face tracking where input features are one-dimensional pixel differences between different sampling points in a single image of a face. A fern takes in D pixel differences and compares them with D randomly chosen one-dimensional thresholds. Results of these split decisions determine an output bin b from 2^D bins in the fern which contains a learned increment $\delta\mathbf{S}_b$ of output variables.

We propose *multi-dimensional random ferns* to make direct use of multi-view image data which is used for the regression training. A feature sampled in a particular region of the face is now a comparison between two surface points which are observed by several cameras. This yields a multi-dimensional input variable with pixel differences from multiple images $\mathbf{I}$. Pixel differences measured in multiple images, yet representing the same feature sample, are similar (subject to view-dependent appearance factors) if the spatial location of the feature provides an accurate correspondence between images. The intuition here is that this inherent multi-view feature constraint implicitly encodes depth information and can be exploited by regression to aid the accurate inference of facial shape and camera locations. Therefore, it is desirable to use multi-dimensional input variables to define fern splitting decisions, as it preserves all information provided by multi-view inputs as opposed to, for example, projecting multi-view differences into a lower dimensional space. This requires an introduction of multi-dimensional split decisions. Training samples are thus partitioned by hyperplanes instead of just single-value thresholds.

The outlined approach is superior to simpler alternatives for handling multiple views. Independent single-view regressions and subsequent fusion of the results would lead to longer computational time at runtime and higher instability of results over time because each single-view regression would suffer from different errors in camera transform and expressions. Limited previous work in different application domains [dCM06, ZFN*10] approached multi-camera regression by concatenating image features across individual views. We experimented with such a baseline approach that stacked one-dimensional pixel differences sampled across individual cameras. These can be passed as an input to a cascaded regression with standard random ferns. This is, however, much slower to train due to feature selection from a larger pool, and more prone to accuracy errors than the introduced multi-view technique.

To formalise our principled approach, we re-define the appearance vector $\mathbf{g} = \gamma(\mathbf{S}, \mathbf{I})$ for the multi-view scenario, where now each component $\mathbf{g}_u$ is a multi-view feature defined in $\mathbb{R}^{|C|}$. A set of 3D sampling points $\{\mathbf{x}_u\}_{u=1}^{U}$ is uniformly distributed on the surface of the blend shape rig, limited to a predefined region of the mesh between the hair- and jawline, as depicted in Fig. 3. We ensure that only points that are visible by at least one camera are selected as samples. Note that the sampling point sets are randomly drawn for each stage regressor R^t, which expands the feature pool and reduces over-fitting to training data. Visibility information with respect to all cameras is precomputed for each sampling point $\mathbf{x}_u$ on the neutral expression B_0 and identity transform $\mathbf{T}$ of the camera rig. A foreshortening vector $\boldsymbol{\alpha}_u$ with length $|C|$ contains the cosine between the surface normal at $\mathbf{x}_u$ and a reversed camera viewing direction, or zero if $\mathbf{x}_u$ is not visible in a camera c. The foreshortening vectors are fixed throughout regression training and testing, since the relative head pose is only expected to change to small extent. Given a model state $\mathbf{S}$, a 3D sampling point $\mathbf{x}_u$ deforms based on blend weights $\mathbf{a}$, which we denote by $\mathbf{x}_u|_\mathbf{a}$, and moves according to a transform $\mathbf{T}$. Grayscale pixel values for $\mathbf{x}_u$ are sampled from all images $\{I_c\}_{c=1}^{C}$ and collected in the multi-dimensional appearance sample $\mathbf{g}_u$. The projection to I_c is defined as $P_c(\mathbf{T}\,\mathbf{T}_c\,\mathbf{x}_u|_\mathbf{a})$, where $\mathbf{T}_c$ is the pose of camera c within the capture rig and $P_c(\cdot)$ describes the intrinsic projection.

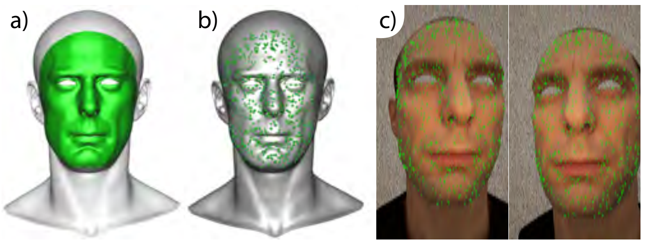

Figure 3: Sample point set $\{\mathbf{x}_u\}_{u=1}^{U}$. We distribute samples (b) on a region of the 3D mesh surface that corresponds to the face between chin and hairline (a). These sample points are being projected into the cameras to compute their appearance vectors (c).

The input feature to our ferns is a multi-dimensional difference vector $\boldsymbol{\delta}_{us} = (\mathbf{g}_u - \mathbf{g}_s)$ between the appearance samples of surface points $\mathbf{x}_u$ and $\mathbf{x}_s$ over all views. Since the feature might be occluded in some views, we define a compact difference vector $\boldsymbol{\delta}'_{us}$ which contains only components from visible views and has thus a variable dimensionality $|C'| \leq [1, |C|]$ as depicted in Fig. 4.

A particular image feature is associated with a split decision in a fern according to a selection process described in Section 4.1.2. The split decision is defined by a hyperplane $\mathbf{h}'$, which splits input features according to $[\boldsymbol{\delta}'^{\,T}_{us}\ 1]\mathbf{h}' > 0$. Because of the nature of random ferns, $\mathbf{h}'$ is computed using $|C'|$ randomly chosen samples from the set of M training samples. For every training sample, the appearance feature $\boldsymbol{\delta}'_{us}$ is computed. From these feature vectors the splitting hyperplane $\mathbf{h}'$ in $|C'|$-dimensional space is computed as follows

$$[\mathbf{Q}\,\mathbf{1}]\mathbf{h}' = \mathbf{0}, \tag{5}$$

where the rows of a square matrix $\mathbf{Q}$ are the feature vectors. The vector $\mathbf{h}'$ is of length $|C'| + 1$. This is an under-constrained linear system and the null-space corresponding to $\mathbf{h}'$ is solved by SVD. Such hyperplanes are calculated for D multi-view input features feeding into a fern. Each feature and associated hyperplane can have a different dimensionality. In the monocular case ($|C| \equiv |C'| \equiv 1$), the structure reduces seamlessly to the standard random fern with one-dimensional thresholds.

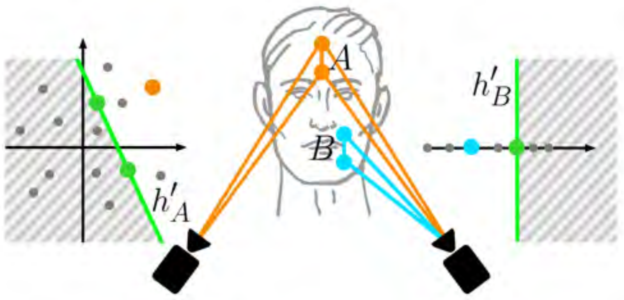

Figure 4: Two different sampling point pairs illustrate multi-view image features A and B. They are visible from differing camera counts, therefore their feature spaces have different dimensionality $|C'|$. Green lines represent fern hyperplanes $\mathbf{h}'$ which partition the feature values across training samples (grey dots).

Once the fern structure is established, it needs to be trained. Training involves partitioning all M training samples according to the D hyperplanes. The binary splits will associate a sample to a bin b from 2^D fern bins. For every bin b, the model state increment $\delta\mathbf{S}_b$ is computed as

$$\delta\mathbf{S}_b = \frac{\sum_{m\in\Omega_b} (\delta\hat{\mathbf{S}}_m - \delta\mathbf{S}_m^{f-1})}{\omega + |\Omega_b|} \tag{6}$$

from the current residuals of the training samples in this bin, denoted as Ω_b. This represents a piece-wise constant approximation of the residuals $(\delta\hat{\mathbf{S}}_m - \delta\mathbf{S}_m^{f-1})$ across the training set. The shrinkage parameter ω influences the learning rate of the ferns and we compute it as $\omega = M/2^D$. The formulation in Eq. 6 reduces the rate for a bin b if there is a low number of samples Ω_b.

4.1.2. Feature selection

Selecting good features to construct the ferns is very important. To this end, we follow the correlation-based feature selection proposed by [CWWS12], but extend it to multi-dimensional features. The selection is based on an assumption that a good feature $\boldsymbol{\delta}'_{us}$ for learning is highly correlated with the regression target $(\delta\hat{\mathbf{S}} - \delta\mathbf{S}^{f-1})$ of a fern f throughout all training samples. To compute the correlation, the input and output vectors across the training set are projected to 1D. We define $\mathbf{y} \in \mathbb{R}^M$ as a vector of one-dimensional projections of M model state residuals, where each residual $(\delta\hat{\mathbf{S}} - \delta\mathbf{S}^{f-1})$ is projected onto a random vector $\boldsymbol{\upsilon}$ of the same dimensionality as the residual, yielding a scalar value. Since the individual variables of the model state have different units and different importance, we use a different random distribution per variable. This is realized by multiplying a uniform distribution within [-1,1] with a weight β per variable. Unlike previous work [WCHZ14] who experimentally determined suitable values for β, we compute the weight automatically. To identify how strong a variable influences the residual, we apply a unit change to the variable and compute the resulting change of the 3D mesh. The larger the change, the higher the influence of the variable, so we set β to the computed 3D change and normalize over all components. In addition to the residual, we also need to project the multi-dimensional appearance vector $\mathbf{g}'_u$ to 1D. This differs from previous implementations, where the appearance was already a scalar. We propose to conduct the projection using the foreshortening vector $\boldsymbol{\alpha}_u$ which is fixed for all training samples. This down-weights pixel values from oblique views and reduces their influence on the selection process.

The correlation between a feature with point pair $\mathbf{x}_u$ and $\mathbf{x}_s$ and model state residuals is calculated as follows

$$\operatorname{corr}(\mathbf{y}, \boldsymbol{\rho}_u - \boldsymbol{\rho}_s) = \frac{\operatorname{Cov}(\mathbf{y}, \boldsymbol{\rho}_u) - \operatorname{Cov}(\mathbf{y}, \boldsymbol{\rho}_s)}{\operatorname{Var}(\mathbf{y})\operatorname{Var}(\boldsymbol{\rho}_u - \boldsymbol{\rho}_s)}. \tag{7}$$

This formulation proposed by [CWWS12] allows efficient correlation computation between $\mathbf{y}$ and the variable difference $\boldsymbol{\rho}_u - \boldsymbol{\rho}_s$. Theoretically, there are U^2 sample point differences which yield potential features for each split decision in a fern. We limit this pool by enforcing the same visibility of the sample points $\mathbf{x}_u$ and $\mathbf{x}_s$ in the cameras C. Another important constraint is locality of point differences on a face [KJ14] which increases their robustness against illumination changes. A simple 3D distance threshold ($10mm$ in our case) is used to further reduce the number of considered features $\boldsymbol{\delta}'_{us}$. We rank all features from the reduced pool according to Eq. 7 and select the one with the highest $\operatorname{corr}(\mathbf{y}, \boldsymbol{\rho}_u - \boldsymbol{\rho}_s)$. We repeat this process D times to determine all input feature variables for a fern. The hyperplanes associated with individual features are then computed as described in the previous section.

4.1.3. Training sample augmentation

The regression learns transitions between $\mathbf{S}_m$ and $\hat{\mathbf{S}}_m$ in the training set so it can estimate a change of model state between consecutive images in live streams (for instance expression changes or camera shake). This section explains how ground-truth training pairs $(\mathbf{I}_n, \hat{\mathbf{S}}_n)$ are augmented with initial states to form training samples $(\mathbf{I}_m, \hat{\mathbf{S}}_m, \mathbf{S}_m)$. The strategy differs from the previous methods [CWLZ13, WCHZ14, CHZ14] in simpler design and provides good regression stability. Each ground-truth state $\hat{\mathbf{S}}_n = (\hat{\mathbf{a}}_n, \hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n)$ is perturbed several times to generate $\mathbf{S}_m$ of different types:

- expression initial states $\mathbf{S}_m = (\mathbf{a}_m, \hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n)$ where $\mathbf{a}_m$ is a similar expression to $\hat{\mathbf{a}}_n$ in terms of a facial shape. We find m_e closest expressions from all $\hat{\mathbf{a}}$ in N training pairs (including the original $\hat{\mathbf{a}}_n$ itself).
- rotation initial states $\mathbf{S}_m = (\hat{\mathbf{a}}_n, \hat{\mathbf{r}}_n + \delta\mathbf{r}, \hat{\mathbf{t}}_n)$ where $\delta\mathbf{r}$ is a small rotational deviation from the ground truth transform $(\hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n)$. We generate m_r rotations with a step s_r in positive and negative direction around each axis ($6m_r$ initial states altogether).
- translation initial states $\mathbf{S}_m = (\hat{\mathbf{a}}_n, \hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n + \delta\mathbf{t})$ where $\delta\mathbf{t}$ is a small translational deviation from the ground truth transform $(\hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n)$. We generate m_t translations with a step s_t in positive and negative direction along each axis ($6m_t$ initial states altogether).

The similarity metric between expressions is based on a difference between sampling points $\{\mathbf{x}_i\}$ deformed by the compared blend weight vectors $\mathbf{a}_n$. A sum of 3D point distances represents a magnitude of shape difference between expressions. Note that the transform $(\hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n)$ is not used in the comparison. The expression initial states should not define transitions between very dissimilar facial shapes because it increases instability of regression. However, there is a competing requirement of good connectivity between the ground-truth $\mathbf{a}_n$ in the blend shape space such that the regression learns all plausible transitions between the blend shapes $\mathbf{B}$. Sampling of rotation and translation initial states is also related to the transforms $(\hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n)$ provided in the training pairs. The numbers of steps m_r, m_t and the step sizes s_r, s_t should define transform deviations which cover the space in between $(\hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n)$ for which we have image data $\mathbf{I}_n$. Overall, the augmentation step expands the number of final training samples to $M = (m_e + 6m_r + 6m_t) \cdot N$.

4.2. Online tracking

After the offline training stage, an actor-specific regressor can track the actor's face from multi-view image streams in real-time. The image streams need to be captured by the same camera rig as assumed during training time. A benefit of multi-dimensional random ferns is fast execution which yields real-time model state estimation. An initial state $\mathbf{S}_l$ for the regression is derived from the solution $\tilde{\mathbf{S}}$ in the previous frame. The vector $\mathbf{S}_l$ is sequentially updated by stage regressors R^t as defined in Eq. 2 where the appearance vector $\mathbf{g}^t$ is resampled from the current multi-view images $\mathbf{I}$ before each stage t. A stage increment of the model state is computed by an internal chain of random ferns according to Eq. 4. Evaluation of individual ferns happens in parallel because their input image features are extracted from the common vector $\mathbf{g}^t$ sampled beforehand. Note that visibility of multi-view features is determined in the training phase and it is not dynamically changed during online tracking. This is unproblematic, since the helmet cameras can be assumed to be largely fixed relative to the face.

To increase robustness of the final solution, multiple independent regressions are performed simultaneously and their results are fused together. The regressions are initialised from different model states $\mathbf{S}_l$. The previous solution $\tilde{\mathbf{S}} = (\tilde{\mathbf{a}}, \tilde{\mathbf{r}}, \tilde{\mathbf{t}})$ is perturbed to generate $\mathbf{S}_l$ of different types:

- expression initial states $\mathbf{S}_l = (\mathbf{a}_l, \tilde{\mathbf{r}}, \tilde{\mathbf{t}})$ where $\mathbf{a}_l$ is a similar expression to $\tilde{\mathbf{a}}$ in terms of a facial shape. We find the l_e closest expressions in the training set.
- transform initial states $\mathbf{S}_l = (\tilde{\mathbf{a}}, \mathbf{r}_l, \mathbf{t}_l)$ where the transform $(\mathbf{r}_l, \mathbf{t}_l)$ is similar to $(\tilde{\mathbf{r}}, \tilde{\mathbf{t}})$ in terms of a facial movement. We find l_T closest transforms in the training set.

The similarity metric based on the sparse points V is used for the expression comparison as in the training augmentation. The same fast metric is applied to the transform comparison where the similarity of $(\mathbf{r}_l, \mathbf{t}_l)$ and $(\tilde{\mathbf{r}}, \tilde{\mathbf{t}})$ is given by point distances. We seed the regressions from the ground-truth states presented at training time because this constrains solutions into known expression space and camera movement range. Multiple regression solutions $\mathbf{S}$ are averaged together per output variable. This reduces temporal jitter in the tracking, which we further attenuate using a light-weight temporal filter. Camera transform variables are smoothed by a half-Gaussian kernel with standard deviation σ_t. Blend weights are smoothed by bilateral filter with the same temporal σ_t and a Gaussian kernel with standard deviation σ_v for variable value changes.

5. Synthetic training set generation

It is well understood that in addition to a dependence on underlying features and choice of regression algorithm, the performance of a learning system is heavily influenced by the size and quality of the training data. A regressor trained on data similar to that seen at runtime will typically offer accurate performance. Inspired by this, we learn a regression that is specifically suited to the real-time performance capture task at hand. To this end, we train the regressor on imagery of the actor, rendered from expected camera positions, lit using on-set environment lighting information.

Acquiring such a training set using real photos can be considered a tremendous burden, as the combination of facial expressions, camera positions and lighting conditions could require thousands of training images. Not only is this impractical, but the availability of the actor is also typically extremely limited. Furthermore, once the imagery is acquired it would have to be annotated (manually or semi-automatically) with facial landmark positions in order to determine the expression and camera parameters and subsequently learn the mapping between multi-view images $\mathbf{I}$ and the corresponding model state $\hat{\mathbf{S}}$.

To alleviate the burden of training image acquisition and in order to automatically provide ground truth model states we follow [MKB*16] and learn actor-specific regressors by preparing synthetic training data using rendering techniques. In this work, each ground-truth training pair consists of synthetic face images $\mathbf{I}_n$, generated by rendering the blend shape rig $\mathbf{B}$ from multiple virtual cameras, and the corresponding model state $\hat{\mathbf{S}}_n = (\hat{\mathbf{a}}_n, \hat{\mathbf{r}}_n, \hat{\mathbf{t}}_n)$ that encodes both the facial shape and the pose of the virtual cameras (see Section 4).

In order to generate training imagery we sample from three semantic axes that affect image appearance. Axes represent the abstract dimensions; facial expression E, camera viewpoint V and illumination I. Training images are sampled in this space through combinations of sample points on these three axes. This allows for various design strategies for training set generation. In this work we follow sampling strategies proposed in [MKB*16] that are found to be favourable in terms of empirical tracking quality. As a result our training set design contains typically $N_E = |\mathbf{B}| + 1 = 73$ facial expressions given by original blendshapes. We sample N_V unique rigid camera transforms based on the physical camera helmet design. An exhaustive training set would involve a complete axis sample cross-product combination however in practice we find more frugal sampling to provide desirable camera movement estimation. Sampling the illumination axis adds robustness to lighting change at runtime. We therefore additionally relight the facial rig using N_I different illumination conditions derived from real light probe data. The illumination samples are combined with the expression samples and the viewpoint axis to provide: $N = (N_E \times N_I) + N_V$ training images corresponding to the [E×I,V] design found in [MKB*16].

6. Results

The proposed method tracks an actor's facial performance in real-time using multiple cameras without aid of any markers. This section provides quantitative evaluation of the method on synthetic data, and qualitative results are shown on real video streams from head-mounted and static camera rigs. Examples of robustness to challenging on-set conditions are also provided. For the best illustration, we encourage the reader to view the accompanying video material.

6.1. Experimental configuration

Two types of camera rigs have been employed for our evaluations. The helmet rigs have cameras placed approximately $20\,cm$ away from the actor's face. The cameras have a wide field of view and the imagery also exhibits minor lens distortion. The image streams are captured at $60\,fps$ @ 720p. The static rig used for high-quality performance capture has cameras surrounding the actor at the distance of roughly $1\,m$. These cameras use better lenses with narrower field of view and almost no distortion. The image streams are captured at $42\,fps$ in 1176×864 resolution.

Different training sets have been generated for our experiments depending on the type of input image data. Table 1 describes the design of individual training sets and their size. They contain different numbers of camera viewpoints but the same N images are rendered per view. The subset of cameras from the training set is reported for each experiment. The associated training times are reported for the stereo camera setup. The typical parameter configurations used in the experiments are:

- offline training: $T = 7, F = 200, U = 800 - 1600, D = 5, m_e = 20, m_r = 2, s_r = 1^\circ, m_t = 2, s_t = 1.5\,mm$
- online tracking: $l_e = 10, l_T = 10, \sigma_t = 3.5, \sigma_v = 0.05$

Our framework is implemented using parallelised OpenMP C++ code without the use of CUDA or other GPU specific code. The experimental evaluation has been performed on a standard PC with an Intel Core i7 5960x (3GHz) CPU and 32GB RAM. The runtime regression for a stereo stream input takes $7ms$ per frame (compared to $5ms$ for a monocular stream). Our regression-based tracker is integrated into a live performance capture pipeline. Image data is streamed from a helmet capture rig into the tracker which solves in real-time for the animation controls on the digital-double rig of the actor. A live preview of the performance for a director is visualised in a video game engine.

6.2. Synthetic test data

We provide quantitative evidence towards the hypothesis that *additional per frame image information significantly improves regression quality*. To test this claim we investigate monocular and

Training set	N_E	N_V	N_I	N	M	Time[min]
HelmetRigBasic	73	72	1	145	6670	3.9
HelmetRigA	73	72	45	3357	154422	61.5
HelmetRigB	119^+	327	74	9133	420118	154.3
StaticRigReal	73	73	1	73^*	4234	3.4
StaticRigSynth	73	73	20	1460^*	84680	17.8

Table 1: Characteristics of training sets used in experiments: N_E - number of expressions, N_V - number of viewpoints, N_I - number of illuminations, N - number of training pairs $(\mathbf{I}_n, \hat{\mathbf{S}}_n)$, M - number of final training samples $(\mathbf{I}_m, \hat{\mathbf{S}}_m, \mathbf{S}_m)$. $^+$ Original blendshapes $\mathbf{B}$ extended with additional expressions created by stepping selected weights between 0 and 1. *Different expressions and transforms are mixed together in the training pairs.

multi-view regression solves using synthetic test data with constant illumination. Plausible synthetic facial test-image sequences can be generated in a similar fashion to our training data by creating artist-driven, keyframed expression poses of a blend shape rig and generating an interpolated animation sequence that is then used as per frame test input to the trained models. Although our experiments make use of a blend shape rig with similar appearance for both training and synthetic testing purposes, we generate test keyframes using expression and camera pose combinations not previously seen at training time. Such test sequences provide a sanity check and controllable first regression quality test for our learned models. Making use of synthetic test data additionally affords us the ability to quantify and measure tracking accuracy by comparing to the available synthetic ground-truth data. This provides information and model assessment capabilities not previously available with facial performance capture methodology, which is typically evaluated only qualitatively.

We perform two tests on synthetic data, which are described in the following sections. The first is to support our hypothesis that multi-view regression is more accurate than monocular. The second is to compare different camera rig baselines, which is an effort to identify plausible wide-baseline configurations that provide suitable reconstruction quality while allowing better line-of-sight of the actor.

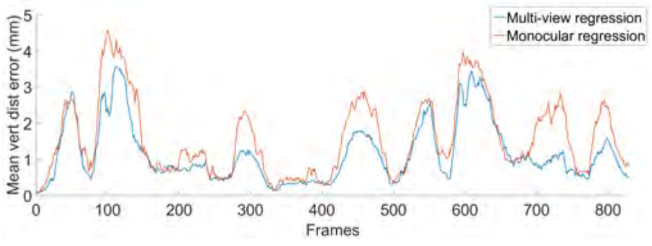

Figure 5: Per-frame mean vertex distance from the ground-truth mesh for frontal monocular and stereo tracking on a synthetic test sequence (830 frames). Mean errors typically increase during frames exhibiting challenging expression and camera movement. This is significantly lessened by our multi-view approach.

6.2.1. Monocular regression vs. multi-view regression

Using models trained on *HelmetRigBasic*, we solve for expression and camera motion parameters and reconstruct the rig pose such that we are able to compare to the corresponding ground-truth mesh. Using the additional image information provided by adding a second viewpoint (±50 deg. stereo baseline), our regression method significantly improves the estimation of the transform when the actor moves his head as well as for challenging expressions. Fig. 5 provides the error per frame, calculated as the mean vertex distance from ground-truth (using the face region highlighted in Fig. 3) for both monocular and stereo tracking on the synthetic test sequence. Fig. 6 highlights corresponding monocular frontal and stereo-view regression reconstruction results for test frame 733 where the quality improvement is clearly visible. For every vertex we compute the Euclidean distance to ground-truth and visualize it as a heat map.

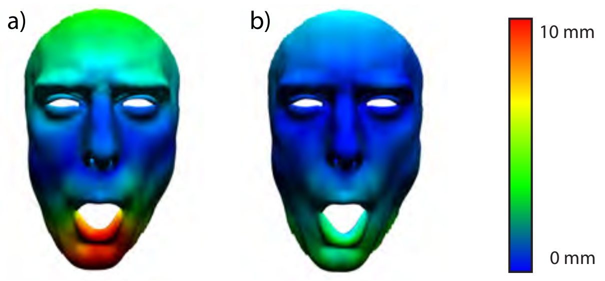

Figure 6: Single (a) and two-view (b) regression results visualised with Euclidean distance to ground-truth. Vertex difference from 0mm (blue) to 10mm (red).

6.2.2. Camera rig baseline comparison

Utilizing synthetic training affords an ability to inform desirable camera positioning and physical head-rig setup. The goal of this experiment is to position the cameras in different configurations and quantitatively evaluate the trade-off between camera configuration and tracking quality. In practice, we find high quality reconstructions remain feasible even when making use of wide stereo baselines. Such configurations are attractive in production scenarios as they move the cameras out of the actor's line-of-sight.

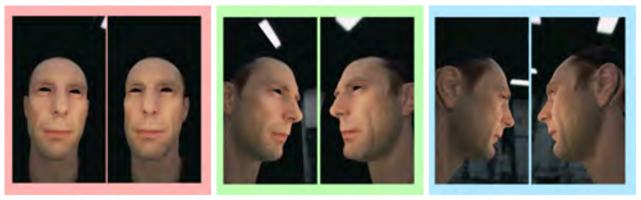

Figure 7: Synthetic test stereo imagery generated by various camera baselines. From left:"*narrow*" , "*wide*" , "*oblique*" stereo pair configurations respectively.

Experimentally, we rotate two synthetic cameras with respect to the head and evaluate several camera baseline configurations (±5 deg., ±50 deg., ±90 deg.). Example training images from *HelmetRigBasic*, rendered from these stereo baselines, can be found in Fig. 7. Additionally Fig. 8 plots the baseline error measurement where we again evaluate performance using the mean vertex error per frame for each baseline configuration. We find that commonly used narrow baseline configurations (±5 deg., "*narrow*") can be pushed to a far wider horizontal pairwise configuration (±50 deg., "*wide*") without significant loss of reconstruction quality. However, if this baseline is extended excessively (eg. ±90 deg., "*oblique*") we begin to observe a significant loss of reconstruction quality. Synthetic view placement investigation of this nature informs real world camera configurations as presented in Section 6.3.

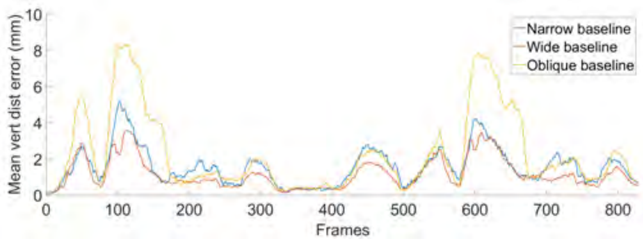

Figure 8: "*narrow*" vs "*wide*" vs "*oblique*" camera baselines evaluated on synthetic test data. The mean vertex distance error is calculated with respect to the synthetic ground truth testing animation.

6.3. Real test data

Further to our synthetic experiments, we perform monocular and multi-view tracking using real image data that contains moderate head motion, a range of challenging facial expressions, and illumination changes. We use single and multi-view video to compare resulting tracking quality.

6.3.1. Monocular regression vs. multi-view regression

To evaluate the performance on real image test sequences we additionally obtain high-quality tracking using the offline system of [BHB*11]. This allows numerical evaluation comparisons analogous to our previous synthetic animation experiments. Fig. 9 shows example input frames and the corresponding mono and stereo regression results trained using the training set *StaticRigSynth*. Regression tracking output is rendered with alpha transparency from a profile view and reprojected onto the corresponding input frame to visualize alignment and reconstruction quality. We highlight a frame that proves challenging for monocular regression yet is improved by the implicit depth information provided by multiple views. This is confirmed by the corresponding error-per-frame reported in Fig. 10. In examining the sequence, we observe that head motion orthogonal to the monocular image plane results in a prominent transform error as the learned monocular model fails to solve an ill-posed problem. In general, the single-view regression provides less precise camera transforms which in turn may affect accuracy by way of the expression solve attempting to explain and compensate for the error. The multi-view technique is better at locking camera pose precisely in 3D space which benefits expression inference. Apart from the accurate camera pose, multiple views help to estimate better facial shape (e.g. mouth 'volume' in expressions such as lip funneler).

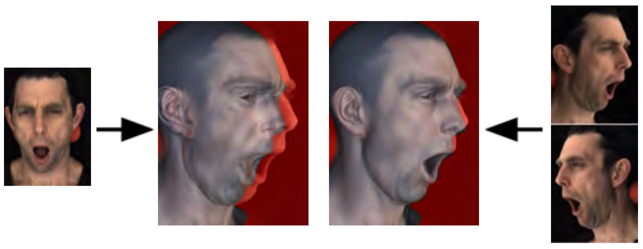

Figure 9: Example frame regression from real imagery. Left: Monocular frontal camera regression exhibits large depth shifts. Right: Regression from stereo side cameras shows improved performance. Note that projections of the facial rig into the frontal view are visually similar for both regressions yet the side view highlights the improved multi-view performance, locking the rig to the true face surface.

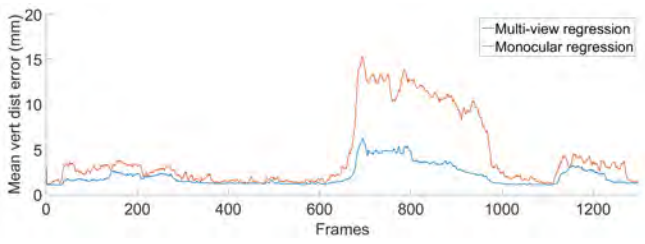

Figure 10: Monocular vs. stereo regression on real imagery from the static capture rig. Large errors occur for monocular tracking during forward head movement. Multi-view information significantly mitigates this type of tracking error. Errors are calculated with respect to the high-quality offline method of [BHB*11].

6.3.2. Camera rig baseline comparison

Our framework is additionally capable of providing high quality tracking on helmet-rig imagery captured in production environments. Using a traditional *narrow* baseline stereo configuration, partially obscuring actor line-of-sight, we produce high quality tracking of a compelling test sequence (Fig. 11). Informed by evidence gained from synthetic testing experiments, we also explore stereo-view regression using a side-view, wide baseline configuration that do not obstruct the actor line-of-sight. Aligned with our synthetic tests, we train a regressor for the wide baseline stereo rig using the training set *HelmetRigA* and demonstrate successful side-view performance capture in real production environments (see supplementary video).

6.4. Illumination robustness

Training synthetically provides the ability to add illumination robustness cheaply in comparison to the prohibitive process of real image data capture under a large number of varying lighting conditions. We illustrate this advantage by synthetically relighting a

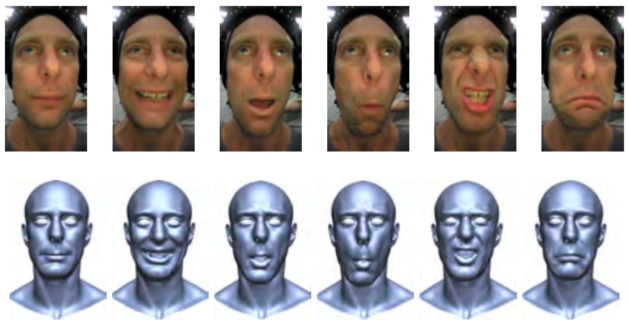

Figure 11: Using a helmet camera rig with a narrow stereo camera configuration, our method provides high quality tracking on a challenging performance on a set.

real test image sequence to provide novel illumination appearance variation not seen during training. We compare tracking results on a regressor trained using combinations of synthetic expression and camera transform under *varying* synthetic illumination conditions, *StaticRigSynth*, to a regressor trained on real imagery exhibiting a similar amount of real expression and camera movement variance yet captured under a *single* lighting condition in the static capture rig, *StaticRigReal*. The training set *StaticRigReal* represents a common case when a limited number of real images has been captured and ground truth model states have been established. Such a dataset does not extend well to new conditions, such as novel illumination, in comparison to synthetically generated training data. Fig. 12 provides an example test frame where real training imagery clearly fails to encode sufficient information to distinguish expressions in previously unseen lighting conditions.

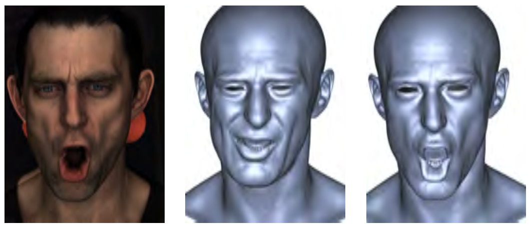

Figure 12: Left: Synthetically relit real test data. Middle: Real training imagery regression. Right: Synthetic training imagery regression. The model trained synthetically more accurately reproduces the observed expression.

Live lighting variations are equally handled using the synthetic training set *HelmetRigB* with many different illumination conditions. Our method performs well on a sequence where the illumination of the face dynamically changes due to large head motion, which provide challenging test conditions. This sequence is provided in the supplementary video material.

6.5. Comparison to recent facial tracking frameworks

Finally, we compare our multi-view regression approach to the facial capture and tracking methods of Beeler et al. [BHB*11], Thies et al. [TZS*16] and Kazemi et al. [KJ14].

6.5.1. Comparison to high-quality offline capture [BHB*11]

We compare our approach to a high-quality performance capture method [BHB*11] that uses 7 cameras and offline processing. In this case, our real-time technique is trained for a side stereo camera pair. As seen in Fig. 13, fine grained skin details are not modelled precisely in our blend shape rig, but we recover expression changes from the actor's footage with very high accuracy. This allows us to potentially drive a face rig with arbitrary level of detail in real-time. We perform quantitative comparison to [BHB*11], which tracks at $\sim$ 15 minutes / frame. The difference between tracked mesh sequences is represented by the blue curve (multi-view regression) in Fig. 10. The mean vertex position difference is 2.1032mm across the entire test sequence.

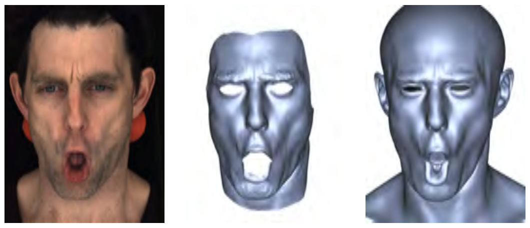

Figure 13: Left: An input frame. Middle: Hiqh-quality offline method of Beeler et al. Right: Our real-time method.

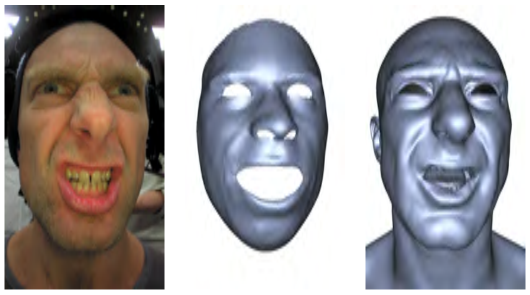

Figure 14: Left: An input frame. Middle: Real-time monocular method of Thies et al. Right: Our real-time method using 2 cameras.

6.5.2. Comparison to real-time monocular capture [TZS*16]

We provide qualitative comparison to the original implementation of a real-time monocular facial capture method [TZS*16], who frame the tracking problem as a non-linear optimization of a person-independent facial rig. The expensive real-time minimization of their objective function is mitigated by a data-parallel GPU-based approach which still requires downsampling of the input stream to 654×352 resolution. In comparison of the results Fig. 14, our multi-view regression method produces more accurate facial reconstructions with a computationally cheaper framework (even using only standard CPU hardware). Note that, as the method of Thies et al. uses a generic face rig, a better facial shape could be obtained by initialisation of the on a sequence with large head pose change (see [TZS*16] for details).

6.5.3. Comparison to sparse landmark tracking [KJ14]

Finally, we compare to a sparse 2D landmark tracking approach [KJ14]. This person-independent method detects 68 2D facial landmarks in every frame. The publically available implementation of this regression technique in the Dlib library [Kin09], trained on generic facial imagery from the iBUG 300-W dataset [STZP13], is used for comparison. The supplementary video displays visual comparison between 2D tracks of face rig vertices (manually selected to correspond with the detected 2D landmarks). Our result is more stable over time and handles large expression changes more successfully. This highlights the dual benefits of additional view information and our tailored synthetic training imagery. We concede that our method and Kazemi et al. [KJ14] are algorithmically different with dissimilar regression targets however this experiment highlights limitations of existing face datasets with annotations. Any learning-based method, trained on these datasets, is likely to suffer loss of precision under less common imaging conditions such as helmet camera footage.

7. Conclusion

We presented a real-time multi-view facial capture system that achieves very high quality markerless tracking. The key to achieving this quality is twofold. On the one hand we employ an actor-specific regressor, trained for the specific illumination environment and capture equipment used during production. On the other hand we introduce a novel multi-dimensional regression framework that seamlessly integrates multiple live camera streams. We assess the improvement in performance that can be gained from these components both qualitatively and quantitatively. We further explore alternate helmet camera designs and propose a variant where the cameras are placed outside the field of view of the actor. This proves very beneficial in practice, since the cameras will be less of a distraction for the actor and allow for an unobstructed line of sight to the director.

Our method requires an actor-specific face rig which is not readily available in general scenarios. This is not deemed as a limitation in the targeted area of film production where facial animation rigs are commonly constructed for main cast members. The actor-specific training data, effectively available as a biproduct, allows for higher accuracy tracking. Further accuracy gains can likely be achieved by improving the underlying blendshape rig as well as more realistic rendering of training imagery. There is a limit on the disparity between training and runtime camera configurations, especially in relative camera positioning. However, greater freedom in camera viewpoint change can easily be achieved by extending viewpoint variance in the training set. Moreover, camera intrinsics can differ at runtime as long as a calibration is provided. We also highlight that changes to the camera rig design are accommodated with ease because synthetic training imagery can be flexibly altered and cheaply generated. While the algorithm models camera motion relative to the head, so far we do not explicitly handle relative camera motion in multi-view rigs. Therefore, adding a relative transform per camera for training set generation could be beneficial. Finally, we would like to further explore alternate helmet rig designs, since they can open up completely novel actor interactions, such as kissing etc. In these cases the regressor would, in addition, have to deal with and reason about partial occlusions.

Despite these limitations, the proposed method presents a large step towards practical on-set real-time markerless facial performance capture, since it provides a recipe to efficiently train highly specialized actor specific regressors by feeding in the helmet rig specifications, some environment samples of the location, as well as the actor's facial rig. Such specialized trackers are very robust and more accurate than generic facial trackers, which render them well suited for virtual production. Applying the method in general scenarios outside of film production is a direction for future work.

Acknowledgements We appreciate the help of Maggie Kosek with video and game engine preparation, and the constructive feedback of anonymous reviewers. We thank Industrial Light & Magic, London for creating actor blendshape rigs with real-time appearance shaders and Jethro Skinner for his acting performances. This work was funded by the InnovateUK project - Real-time Digital Acting (# 101857).

References

[BGY*13] BHAT K. S., GOLDENTHAL R., YE Y., MALLET R., KOPERWAS M.: High fidelity facial animation capture and retargeting with contours. In *Proc. SCA* (2013), pp. 7–14. 3

[BHB*11] BEELER T., HAHN F., BRADLEY D., BICKEL B., BEARDSLEY P., GOTSMAN C., SUMNER R. W., GROSS M.: High-quality passive facial performance capture using anchor frames. *ACM Trans. Graphics (Proc. SIGGRAPH) 30* (2011), 75:1–75:10. 2, 9, 10

[BHPS10] BRADLEY D., HEIDRICH W., POPA T., SHEFFER A.: High resolution passive facial performance capture. *ACM Trans. Graphics (Proc. SIGGRAPH) 29* (2010), 41:1–41:10. 2

[BWP13] BOUAZIZ S., WANG Y., PAULY M.: Online modeling for realtime facial animation. *ACM Trans. Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 40:1–40:10. 2

[CBZB15] CAO C., BRADLEY D., ZHOU K., BEELER T.: Real-time high-fidelity facial performance capture. *ACM Trans. Graph. 34*, 4 (2015), 46:1–46:9. 2, 3

[CHZ14] CAO C., HOU Q., ZHOU K.: Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Trans. Graphics (Proc. SIGGRAPH) 33*, 4 (2014), 43:1–43:10. 2, 3, 4, 6

[CWLZ13] CAO C., WENG Y., LIN S., ZHOU K.: 3d shape regression for real-time facial animation. *ACM Trans. Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 41:1–41:10. 2, 3, 4, 6

[CWWS12] CAO X., WEI Y., WEN F., SUN J.: Face alignment by explicit shape regression. In *IEEE CVPR* (2012), pp. 2887–2894. 4, 6

[CWZ*14] CAO C., WENG Y., ZHOU S., TONG Y., ZHOU K.: Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (Mar. 2014), 413–425. 3

[CXH03] CHAI J.-X., XIAO J., HODGINS J.: Vision-based control of 3d facial animation. In *SCA* (2003). 2

[dCM06] DE CAMPOS T., MURRAY D.: Regression-based Hand Pose Estimation from Multiple Cameras. In *CVPR* (2006), no. Cvpr, IEEE, pp. 782–789. 3, 5

[FDG*12] FANELLI G., DANTONE M., GALL J., FOSSATI A., GOOL L.: Random Forests for Real Time 3D Face Analysis. *International Journal of Computer Vision* (2012). 3

[FHK*15] FENG Z.-H., HU G., KITTLER J., CHRISTMAS B., WU X.: Cascaded Collaborative Regression for Robust Facial Landmark Detection Trained using a Mixture of Synthetic and Real Images with Dynamic Weighting. *IEEE Transactions on Image Processing 7149*, c (2015), 1–1. 3

[FJA*14] FYFFE G., JONES A., ALEXANDER O., ICHIKARI R., DEBEVEC P.: Driving high-resolution facial scans with video performance capture. *ACM Trans. Graphics 34*, 1 (2014), 8:1–8:14. 2

[GVWT13] GARRIDO P., VALGAERTS L., WU C., THEOBALT C.: Reconstructing detailed dynamic face geometry from monocular video. In *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* (2013), vol. 32, pp. 158:1–158:10. 2

[HMYL15] HSIEH P.-L., MA C., YU J., LI H.: Unconstrained realtime facial performance capture. In *Proc. of CVPR* (2015). 2

[HZ14] HAOPENG Z., ZHIGUO J.: Multi-view space object recognition and pose estimation based on kernel regression. *Chinese Journal of Aeronautics 27*, 5 (2014), 1233–1241. 3

[JCK15] JENI A., COHN J. F., KANADE T.: Dense 3D Face Alignment from 2D Videos in Real-Time. In *Face and Gesture* (2015). 3

[KF07] KAKADE S., FOSTER D.: Multi-view regression via canonical correlation analysis. *Learning Theory*, 1 (2007), 1–15. 3

[KH12] KLAUDINY M., HILTON A.: High-detail 3d capture and non-sequential alignment of facial performance. In *3DIMPVT* (2012). 2

[Kin09] KING D. E.: Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research 10* (2009), 1755–1758. 11

[KJ14] KAZEMI V., JOSEPHINE S.: One Millisecond Face Alignment with an Ensemble of Regression Trees. *Computer Vision and Pattern Recognition (CVPR), 2014* (2014). 6, 10, 11

[LTO*15] LI H., TRUTOIU L., OLSZEWSKI K., WEI L., TRUTNA T., HSIEH P.-L., NICHOLLS A., MA C.: Facial performance sensing head-mounted display. *ACM Trans. Graphics (Proc. SIGGRAPH) 34*, 4 (2015). 3

[LYYB13] LI H., YU J., YE Y., BREGLER C.: Realtime facial animation with on-the-fly correctives. *ACM Trans. Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 42:1–42:10. 2

[MKB*16] MCDONAGH S., KLAUDINY M., BRADLEY D., BEELER T., MATTHEWS I., MITCHELL K.: Synthetic prior design for real-time face tracking. In *International Conference on 3D Vision* (2016). 2, 3, 7

[OLSL16] OLSZEWSKI K., LIM J. J., SAITO S., LI H.: High-fidelity facial and speech animation for vr hmds. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 35*, 6 (2016). 3

[RQH*12] RÄTSCH M., QUICK P., HUBER P., FRANK T., VETTER T.: Wavelet reduced support vector regression for efficient and robust head pose estimation. In *Proc. Computer and Robot Vision* (2012), pp. 260–267. 3

[SLL16] SAITO S., LI T., LI H.: Real-time facial segmentation and performance capture from rgb input. In *Proc. of ECCV* (2016). 2

[SRAG15] SHI C., RUAN Q., AN G., GE C.: Semi-supervised sparse feature selection based on multi-view Laplacian regularization. *Image and Vision Computing 41* (2015), 1–10. 3

[STZP13] SAGONAS C., TZIMIROPOULOS G., ZAFEIRIOU S., PANTIC M.: 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *ICCV Workshops* (2013). 11

[SWTC14] SHI F., WU H.-T., TONG X., CHAI J.: Automatic acquisition of high-fidelity facial performances using monocular videos. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 33*, 6 (2014). 2

[TZN*15] THIES J., ZOLLHÖFER M., NIESSNER M., VALGAERTS L., STAMMINGER M., THEOBALT C.: Real-time Expression Transfer for Facial Reenactment. *ACM Trans. Graph. 34*, 6 (2015). 3

[TZS*16] THIES J., ZOLLHÖFER M., STAMMINGER M., THEOBALT C., NIESSNER M.: Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *Computer Vision and Pattern Recognition (CVPR)* (2016). 3, 10, 11

[VWB*12] VALGAERTS L., WU C., BRUHN A., SEIDEL H.-P., THEOBALT C.: Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012). 2

[WBLP11] WEISE T., BOUAZIZ S., LI H., PAULY M.: Realtime performance-based facial animation. *ACM Trans. Graphics (Proc. SIGGRAPH) 30*, 4 (2011), 77:1–77:10. 2

[WCHZ14] WENG Y., CAO C., HOU Q., ZHOU K.: Real-time facial animation on mobile devices. *Graphical Models 76* (2014), 172–179. 2, 3, 4, 6

[WLVGP09] WEISE T., LI H., VAN GOOL L., PAULY M.: Face/off: live facial puppetry. In *Proc. SCA* (2009), pp. 7–16. 2

[ZCD*15] ZHENG S., CAI X., DING C., NIE F., HUANG H.: A Closed Form Solution to Multi-View Low-Rank Regression. *Aaai* (2015), 1973–1979. 3

[ZFN*10] ZHAO X., FU Y., NING H., LIU Y., HUANG T. S.: Human Pose Estimation with Regression by Fusing Multi-View Visual Information. *Transactions on Circuits and Systems for Video Technology 2010* (2010), 1–10. 3, 5

[ZZ11] ZHANG J., ZHANG D.: A novel ensemble construction method for multi-view data using random cross-view correlation between within-class examples. *Pattern Recognition 44*, 6 (2011), 1162–1171. 3