Real-time Face Reconstruction from a Single Depth Image

Vahid Kazemi*† Cem Keskin† Jonathan Taylor† Pushmeet Kohli† Shahram Izadi†
KTH Royal Institute of Technology* Microsoft Research†
vahidk@csc.kth.se {cemke,jota,pkohli,shahrami}@microsoft.com

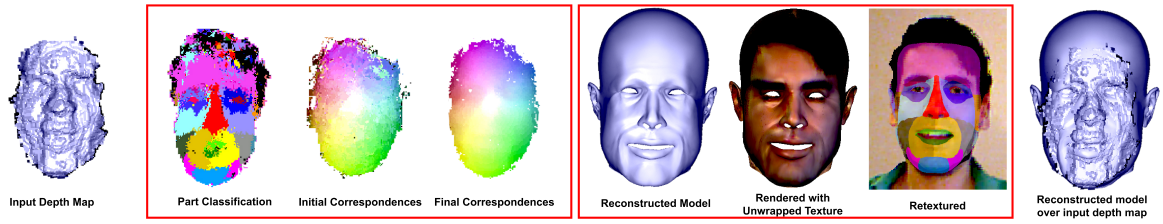


Figure 1: Our method starts with estimating dense correspondences on an input depth image, using a discriminative model. A generative model parametrized by blend shapes is then utilized to further refine these correspondences. The final correspondence field is used for per-frame 3D face shape and expression reconstruction, allowing for texture unwrapping, retexturing or retargeting in real-time.

Abstract

This paper contributes a real time method for recovering facial shape and expression from a single depth image. The method also estimates an accurate and dense correspondence field between the input depth image and a generic face model. Both outputs are a result of minimizing the error in reconstructing the depth image, achieved by applying a set of identity and expression blend shapes to the model. Traditionally, such a generative approach has shown to be computationally expensive and non-robust because of the non-linear nature of the reconstruction error. To overcome this problem, we use a discriminatively trained prediction pipeline that employs random forests to generate an initial dense but noisy correspondence field. Our method then exploits a fast ICP-like approximation to update these correspondences, allowing us to quickly obtain a robust initial fit of our model. The model parameters are then fine tuned to minimize the true reconstruction error using a stochastic optimization technique. The correspondence field resulting from our hybrid generative-discriminative pipeline is accurate and useful for a variety of applications such as mesh deformation and retexturing. Our method works in real-time on a single depth image i.e. without temporal tracking, is free from per-user calibration, and works in low-light conditions.

1. Introduction

We address the problem of reconstructing 3D face shape and expressions in real-time, given only a single depth image. As with the Kinect [36], we are motivated by interactive gaming scenarios, in our case face retargeting or retexturing, where often users will play in low-lighting conditions where color data is unavailable or limited. Our method provides a per-frame estimate of the 3D face shape and expressions using depth data only, avoiding any temporal information or tracking (which is often prone to errors during large motions). It also avoids per-user calibration, which can be a costly step in prior systems.

Per-frame, our method computes a dense correspondence field between an input depth image and a canonical face model in real-time. We fit a deformable face model, parameterized by a set of identity and expression blend shapes, to the data. Minimizing the error in reconstructing the face in the observed depth image lets us estimate the true parameters of the face model and in turn the dense correspondence field. Traditionally this generative approach has been shown to be computationally expensive and non-robust because of the non-linear nature of the reconstruction error. To overcome this problem, we use a discriminatively trained prediction pipeline which provides a robust initial solution. The correspondences are then updated using a variant of the iterative closest point (ICP) algorithm to get an initial fit and then further refined by minimizing the true reconstruction error using particle swarm optimization (PSO).

Our discriminative pipeline first estimates an initial set of correspondences between the deformable model of the face and the data using random forests. Previous methods for computing dense correspondences for deforming objects use the structure of a classification tree for constructing the regression forest [37, 35]. However, we find that employing a joint classification and regression objective [21] leads to more accurate correspondences. The correspondence field resulting from our hybrid generative-discriminative (Figure 1) pipeline is accurate and useful for a variety of applications such as mesh deformation and retexturing.

Related work: Early work on facial tracking and model fitting typically used monocular RGB video sequences and

tracked the motion of *sparse* 2D facial features or triangulated 3D points across frames [3, 30]. Approaches typically used parametric 2D or 3D shape models, which were matched against these sparse correspondences in the video sequence. Approaches are therefore typically generative, but some adopt discriminative methods for feature detection. Early work on facial tracking used variants of active appearance models [12] for parameterizing the face in 2D. Whilst powerful for the initial detection of the face, these 2D linear approaches fail to model complex motions or large deformations of the face. [5] proposed a morphable 3D parametrization for the face, which has been adopted as a richer representation in more recent work.

[10, 4, 39, 14] also extract blend shape parameters but from a sparse set of visually tracked landmarks, to fit 3D morphable models to video sequences. They demonstrate a variety of video editing tasks such as facial animation transfer and face replacement. Kemelmacher-Shlizerman et al. [27] and Li et al. [32] propose purely discriminative approaches, which use sparse feature tracks and matching to retrieve a 3D face model from a single RGB image. [19] use a pipeline that combines sparse feature matching, blend shape estimation, and dense geometry reconstruction (using optical flow and a shading based refinement step) to demonstrate impressive 3D facial reconstructions from a single monocular sequence.

The RGB systems so far are non real-time in terms of performance. [44] use a coarse 3D morphable model in combination with a 2D active appearance model and sparse features for real-time facial tracking in video. More recent work has shown how regression forests can learn to find a sparse set of facial features in real-time [15, 26]. In [8], the 3D positions of facial landmark points are inferred by a regressor from 2D video frames of an off-the-shelf web camera or mobile phone. From these 3D points, the pose and expressions of the face are recovered by fitting a user-specific 3D morphable model.

In the computer graphics community, facial tracking and modeling has received much attention. Here algorithms aim at dense detailed facial capture for performances. Given the desire for high-quality, complex multi-camera and motion capture rigs, costly scanner systems, custom lighting and studio conditions are required [34]. Multi-camera rigs have been used to track markers or find dense correspondences using invisible make-up [43, 18, 22]. [24] combines marker-based motion capture with high quality 3D scanning for detailed capture of facial expressions. Other dense 3D methods, track shape templates from a dynamic active 3D scanner [42, 40], including non-facial shapes [29]. Whilst these methods exploit the dense depth data only, they rely on very high quality input for robust tracking and estimation. Our method works with commodity but noisy depth cameras. High-quality facial performances have also been demonstrated with passive stereo camera setups [7, 2, 38]. All these dense approaches produce high-quality results, but most require complex, expensive setups and high computational costs.

With the advent of consumer depth cameras, many real-time head pose, facial tracking and modeling pipelines have been proposed [41, 31, 6, 17]. Whilst demonstrating impressive results, these real-time methods rely on both 2D sparse RGB features and depth data. The RGB data is typically used to increase robustness, given the noisy depth data. As such, these methods are limited to visible lighting conditions, and are non-robust to extreme changes in illumination. Further, all these systems use a personalized blend shape model, which requires either online or offline per-user calibration. In contrast, recent work in full body pose estimation [36] has freed itself from the RGB constraints by considering only depth images. The seminal work uses a random forest to rapidly label the identity of every depth pixel [36]. Our work is most similar to [37] that instead uses a forest to predict a dense set of correspondences back to a canonical human body model. The model is then fit to make the corresponding model and data points agree, but no update to the correspondences is considered. The work [35] demonstrated that improved pose accuracy can be obtained by updating the correspondences in addition to the model parameters. We take a similar approach for the discriminative part of our pipeline, with the advantage that our method is more efficient and operates in real-time. Note that we additionally employ PSO for tracking and further refinement. Our contributions can be therefore summarized as follows:

1. We present a unified framework for dense correspondence estimation, and facial shape and expression reconstruction. Our method operates in real-time, requires only depth data, and reconstructs each frame independently. Therefore our method is not prone to failures due to fast motion, is largely invariant to different lighting conditions, and enables new interactive scenarios. Our method also avoids expensive per-user calibration steps, and uses only a generic face model for fitting.

2. Quantitative and qualitative results are presented verifying that our estimated correspondence field is accurate enough for facial shape and expression reconstruction, and retexturing in real-time.

3. In contrast to both [37] and [35] which have used similar approach for human pose estimation, we directly minimize a true measure of reconstruction error with PSO while still maintaining real-time speeds.

4. We demonstrate that using a classification objective only in the upper levels of tree training helps with the multimodality of the correspondence distributions.

2. The Generative Model

We will use $O = \{z_n\}_{n\in\mathcal{I}}$ to represent the observed depth image where z_n is the depth of pixel n in the set $\mathcal{I}$ of image pixels. Similarly, we will use $f(\theta) = \{\hat{z}_n(\theta)\}_{n\in\mathcal{I}}$ to represent the depth of the pixels in the image rendered from the face model (see below) with parameters θ. Given the observed depth image O, the posterior distribution over the parameters θ of the face model is then defined as

$$\Pr(\theta|I) \propto \exp(-E(O, f(\theta))) \tag{1}$$

where E is the reconstruction error which measures the distances between the observation and rendered image under the parameters θ. The Maximum a Posteriori configuration of the face model parameters can be computed by solving the inverse problem:

$$\theta* = \arg\min_{\theta\in\Theta} E(O, f(\theta)) \,. \tag{2}$$

For the remainder of this manuscript, we will make the dependence of the energy on O and f implicit and simply write $E(\theta)$.

Parameterizing and rendering the 3D face model: We use a blend shape based model for synthesizing 3D faces. This model is able to account for variation in 3D structure caused by both the identity of the user as well as his or her expression. In this model, a base mesh $\{v^b_m\}_{m=1}^M$ consisting of M vertices is deformed using a linear combination of N_{identity} identity blend shapes and $N_{\text{expression}}$ expression blend shapes. The i'th identity blend shape contains a 3D offset v^s_{mi} for each vertex m and likewise the i'th expression blend shape contains an offset v^e_{mi} for vertex m. A set of coefficients $\{\alpha_i\}_{i=1}^{N_{\text{identity}}}$ determine how much of the identity blend shape i to add to the base mesh. Similarly the set $\{\beta_i\}_{i=1}^{N_{\text{expression}}}$ determine the same for the expression blend shapes.

Having specified the face model, we now describe how we use it to render an image that can be matched with the given observations. We first generate a deformed mesh $\{v_m\}_{m=1}^M$ from the face model by applying the weighted vertex offsets to the base mesh, and applying a global scaling s through

$$v_m = s\left(v^b_m + \sum_{i=1}^{N_{\text{identity}}} \alpha_i v^s_{mi} + \sum_{i=1}^{N_{\text{expression}}} \beta_i v^e_{mi}\right) . \tag{3}$$

The deformed mesh is then positioned in 3D using a rotation $R \in SO(3)$ and translation t generate a set of $3D$ points $\{p_m\}_{m=1}^M$ via

$$p_m = Rv_m + t \,. \tag{4}$$

In total, the parameter vector θ of our model is simply the concatenation of the identity blend shape weights $\{\alpha_i\}_{i=1}^{N_{\text{identity}}}$, the expression blend shape weights $\{\beta_i\}_{i=1}^{N_{\text{expression}}}$, global scale s, rotation R and translation t.

The observations $f(\theta) = \{\hat{z}_n(\theta)\}_{n\in\mathcal{I}}$ ultimately take the form of a rendered image that is produced by sweeping over each image pixel index $n \in \mathcal{I}$ and calculating a depth $\hat{z}_n(\theta)$. If the pixel index back projects to a point within the bounds of our model (i.e. into the convex hull of the positioned vertex set $\{p_m\}_{m=1}^M$), we simply employ standard graphics techniques to render the depth $\hat{z}_n(\theta)$. We denote this set of foreground pixel indices as $\mathcal{I}_{\text{fg}} \subseteq \mathcal{I}$. For a pixel n in the background $\mathcal{I}_{\text{bg}} = \mathcal{I} - \mathcal{I}_{\text{fg}}$, we simply render a fixed background depth $\hat{z}_n(\theta) = 5000mm$ far behind the mesh to simulate a wall.

The golden energy: We now describe the reconstruction error that implicitly encodes the likelihood of seeing an observed image given the model parameters θ. Given a depth image $\{z_n\}_{n\in\mathcal{I}}$, we assume that pixels within the foreground come from our generative model, whereas pixels in the background are not likely to. We thus use a truncated L-1 difference between the rendered and the observed image as the reconstruction error:

$$E_{gold}(\theta) = \sum_{n\in\,\mathcal{I}} \min(|z_n - \hat{z}_n(\theta)|, \zeta) \tag{5}$$

We refer to this error as the 'golden' energy of the model parameters, as it represents how well the model, under parameters θ, fits the observed image modulo the known deficiencies of our model (e.g. our naive constant background model).

Substituting equation 5 into 2, we get the model fitting optimization problem:

$$\theta* = \arg\min_{\theta\in\Theta} \sum_{n\in\,\mathcal{I}} \min(|z_n - \hat{z}_n(\theta)|, \zeta). \tag{6}$$

This is a hard non-linear optimization problem as it has numerous local minima and even locally differentiating it is non-trivial [16]. One way to handle such problems is to use a derivative free optimizer with a good initial guess. For this we employ the PSO method [28] that works by evolving a population of P particles (i.e. solutions) $\{\theta_1, ..., \theta_P\}$. The rules for updating these particles are standard and we refer the reader to [28] for more details. Briefly though, the swarm's movement is designed to strike a balance between global exploration of the parameter space and local exploitation of the collective knowledge that it has obtained from each particle's evaluation. In theory, PSO is capable of performing a robust global optimization when enough particles are allowed to evolve for sufficiently many generations, however, doing so is prohibitive for a real-time application. It is thus crucial that we have a good initial guess and that we only use PSO to perform a fast local derivative free optimization of (5). We thus return to the use of PSO in section 4 and now consider an alternative energy which we can use to obtain such a good initial guess.

The silver energy: To this end, we back project the foreground depth pixels to obtain a 3D data point cloud $\{x_n\}_{n=1}^{N_{fg}}$, where N_{fg} is the number of such pixels. We assume that each data point x_n is a noisy observation of a point $S(u;\theta)$ on the surface of our model. Here $u \in \Omega$ is a coordinate (*i.e.* a triangle index and barycentric coordinate) in the (2D) surface domain Ω of our surface. Assuming a Gaussian noise model, this allows us to define a new energy based on the distance from each observation to the models surface as

$$E_{silver}(\theta) = \sum_{n=1}^{N_{fg}} \min_{u \in \Omega} \|S(u;\theta) - x_n\|^2 \,. \tag{7}$$

By naming, for each data point x_n, a corresponding model coordinate u_n we can pass the inner minimizations through the summation and rewrite this as

$$E_{silver}(\theta) = \min_{u_1,...,u_{N_{fg}}} \sum_{n=1}^{N_{fg}} \|S(u_n;\theta) - x_n\|^2 \,. \tag{8}$$

This allows us to define yet another energy

$$E'_{silver}(\theta, U) = \sum_{n=1}^{N_{fg}} \|S(u_n;\theta) - x_n\|^2 \tag{9}$$

defined both on the block of parameters θ and a block of surface coordinates $U = \{u_1, ..., u_{N_{fg}}\} \subseteq \Omega$. Importantly $E_{silver}(\theta) = \min_U E'_{silver}(\theta, U) \leq E'_{silver}(\theta, U)$ for any U, and thus we can approach minimizing (7) by minimizing (9). When $S(u;\theta)$ is differentiable and there is a procedure available for calculating $u^*(x;\theta) = \arg\min_{u \in \Omega} \|S(u;\theta) - x\|^2$, one can perform coordinate descent on the θ and U to obtain a classical iterative closest point method [13].

In our case, we use the M vertices of our mesh to provide a discretization $\Omega' = \{u'_1, ..., u'_M\}$ of Ω where $S(u'_m;\theta) = p_m$ is well defined from (4). We then desire to minimize (9) but use a set of surface coordinates U' restricted to this discretization (i.e. $U' \subseteq \Omega'$). Importantly,

$$E_{silver}(\theta) = \min_{U \subseteq \Omega} E'_{silver}(\theta, U) \leq \min_{U' \subseteq \Omega'} E'_{silver}(\theta, U') \tag{10}$$

where the first bound gets tighter as we optimize for U and the second bound can be made very tight using a dense enough mesh. This is the case for our discretization where $M = 11211$, a number close to the typical number of pixels on the face.

To minimize $E'_{silver}(\theta, U')$ we observe that our restriction allows us to satisfy both properties needed to craft a classical iterative closest point algorithm. The function $S(u';\theta)$ for fixed $u' \in U'$ defined by (4) and (3) has well defined derivatives that are straightforward to compute. The function $u'^*(x;\theta) = \arg\min_{u' \in \Omega'} \|S(u';\theta) - x\|^2$ is now approachable by, for example, iterating over the M possible values in Ω'. We show in the next subsection, how we obtain a good initial guess for U' and now provide more details about how we efficiently perform coordinate descent on $E'_{silver}(\theta, U')$.

Our procedure for optimizing over θ while holding U' fixed exploits the availability of derivatives and the squared error terms in the energy. This allows us to exploit the Gauss-Newton approximation $J(\theta)^t J(\theta)$ of the Hessian $H(\theta)$, where $J(\theta)$ is the Jacobian, and perform powerful second order Gauss-Newton step $\theta_{k+1} = (J(\theta_k)^t J(\theta_k))^{-1} J(\theta_k)^t r(\theta_k)$ where $r(\theta_k)$ is the vector of residuals at step k. We use the publicly available Ceres implementation [1] of the popular Levenberg-Marquardt variant [33] that simply damps the $J(\theta_k)^t J(\theta_k)$ matrix when the quadratic approximation fails to yield a good step. This variant combines the advantage of quadratic convergence when the quadratic approximation is valid (e.g. provably so near local minima) with a graceful degradation to first order gradient descent when the approximation fails to allow progress to be made.

Our procedure for optimizing over U' while holding θ' fixed is carefully designed to maintain real-time speeds. Indeed, the naive method of calculating $u^*(x;\theta) = \arg\min_{u' \in \Omega'} \|S(u';\theta) - x\|^2$ by iterating linearly over the elements of Ω' results in an algorithm with a $O(NM)$ complexity. It has been suggested [35] to use a KD-Tree to reduce the complexity to $O(NlogM)$, but it is not obvious how to implement the tree construction at real-time speeds. In fact, as we are searching over model points dependent on θ, and not data points which are independent of θ, one has to construct a new KD-tree at every iteration. In order to obtain real-time speeds, we instead perform a simple but effective local approximation that is easily parallelizable. We rely on the GPU to quickly render our model into 3D and only search for rendered vertices in a small local neighbourhood that back projects from a rectangular patch surrounding each depth pixel.

3. Discriminative Model

Our discriminative model consists of a random forest of binary decisions trees. For each pixel in the input depth image, the forest predicts its corresponding position $u' \in \Omega'$ on the canonical face model. This approach is similar to [37] that has been applied to body pose estimation task. To train their forest, Taylor *et al.* use a surrogate classification objective based on body parts, which has been shown to achieve higher accuracies than a pure regression objective in [20]. In this paper, we show that a hybrid objective yields better results than both a pure classification and a pure regression objective for our application.

The decisions that each split node of the trees make are based on the simple depth-invariant depth comparison features ($\mathbf{f}_\phi$) proposed in [36]. Although extremely lightweight to compute, these features have been shown to be powerful

for a variety of tasks [36, 37, 20]. At each node, our training algorithm processes a sample set Q as follows:

1. A pool of features $\phi = \{\phi_i\}_{i=1}^{|\phi|}$ is randomly selected.
2. For each feature ϕ_i, a set of candidate thresholds $\{\tau_{ij}\}_{j=1}^{|\tau|}$ is selected.
3. For each set of split parameters $z = (\phi, \tau)$, samples Q are divided into left and right partitions: $Q_l(z) = \{Q : \mathbf{f}_\phi < \tau\}$ and $Q_r(z) = Q \setminus Q_l(z)$.
4. The optimal parameter z^* is chosen to maximize the information gain ($G(z)$)

$$\mathcal{G}(z) = \mathcal{H}(Q) - \sum_{s \in (l,r)} \frac{|Q_s(z)|}{|Q|} \mathcal{H}(Q_s(z)) \quad (11)$$

$$\mathcal{H}(Q) = \alpha \mathcal{H}^* + (1 - \alpha)\mathcal{H}^\dagger \quad (12)$$

with $\alpha = \mathbb{1}(depth \leq L)$. In the above, L indicates the depth at which we switch the objective from that of classification to regression. The classification objective is based on the Shannon entropy defined over part classes whereas the regression objective is simply a measure of correspondence variation

$$\mathcal{H}^*(Q) = - \sum_{c \in classes} P(c) \log P(c) \quad (13)$$

$$\mathcal{H}^\dagger(Q) = \mathrm{Tr}(\Lambda(Q)) \, . \quad (14)$$

In the above, $P(c)$ is the proportion of samples in Q with class label c, and $\Lambda(Q)$ is the covariance of the regression target labels in Q.

5. If the appropriate information gain $\mathcal{G}(z^*)$ can be made sufficiently large, the split is accepted. The algorithm then continues recursively down the right and left branches until a maximum depth is achieved or the sample set in a node becomes small.

Lastly, we use the mean-shift algorithm [11] on the empirical distribution that ends up in each leaf to find the modes of the distribution. At test time, each decision tree is traversed based on its selected features until a leaf node is reached and the set of all modes found by mean shift are aggregated. The final output of the forest is the correspondence u' closest to the strongest mode in this aggregation.

4. Hybrid Method

We now return to our original task of minimizing the golden energy (5) to recover both the model parameters and a good set of correspondences. Although this energy is difficult to optimize directly, we have now developed the necessary tools in the previous section to develop our hybrid method that can rapidly obtain a good minimum.

We start by detecting the head using a standard skeleton tracker [36] and removing the outliers by simple distance thresholding. Then, the algorithm, described in Algorithm 1, leverages our discriminative correspondence to obtain a good initial guess for the correspondences U' used in our proxy generative model described by the silver energy. We set all of the blend shape weights to zero and solve simultaneously for an initial optimal global scale, translation and rotation [23]. We then perform N_{ICP} iterations of an iterative closest point algorithm by alternating between a continuous optimization of θ and a discrete update of U' as described in Section 2 to get close to a local minimum of (7). Note that, in practice, we add a small regularizer $\lambda \sum_{i=1}^{N_{\text{expression}}} \rho(\beta_i)$ to (7) [1] which we find helps condition the optimization. We then sample near the current solution to construct a population of PSO particles, which rapidly refines the solution locally to drive down (5).

Algorithm 1 Pseudo-code of our hybrid single frame model fitting and correspondence finding procedure. The same algorithm is used for both identity and expression fitting by setting $N_{\text{expression}}$ or N_{identity} to zero respectively. In the latter case, the base mesh is assumed to have been morphed to incorporate the identity.

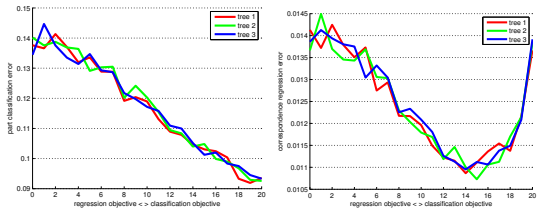
Initialize scalars $\{\alpha_i\}_{i=1}^{N_{\text{identity}}}$ and $\{\beta_i\}_{i=1}^{N_{\text{expression}}}$ to zero.
Evaluate forest on depth image to obtain initial U'.
Solve for optimal R, t, s holding everything else fixed.
for $i = 1$ to N_{ICP} **do**
  Optimize $\{\alpha_i\}_{i=1}^{N_{\text{identity}}}$ and $\{\beta_i\}_{i=1}^{N_{\text{expression}}}$, R, s and t using LM.
  Update U' using closest point approximation.
end for
Initialize PSO by sampling near current solution.
for $i = 1$ to N_{PSO} **do**
  Evolve PSO Swarm.
  Update U' using closest point approximation.
end for

Our algorithm can be used in two different modes of operation: identity fitting and expression fitting. In identity fitting, only identity blend shapes are used (i.e. $N_{\text{expression}} = 0$) to fit the shape of the user in a neutral pose. These identity blend shapes can then be incorporated into the base mesh by setting $v_m^b \leftarrow v_m^b + \sum_{i=1}^{N_{\text{identity}}} \alpha_i v_m^s$ for $m \in \{1, ..., M\}$. The algorithm, can then be switched to expression fitting mode where only expression blend shapes are used (i.e. $N_{\text{identity}} = 0$) as the base mesh has been fit to the identity of the user.

5. Experiments

This section details a set of experiments that we have performed to evaluate different components of our system individually and as a whole. We use synthetic data to train and test our system, and also provide qualitative generalization results on real depth images 5.

[1] Here ρ is the Huber error functional [25]



(a) Part classification error (b) Correspondence error

Figure 2: Effect on classification (a) and correspondence regression error (b) resulting from varying the switching depth L. Note that naturally using a pure part classification objective ($L = 20$) does lead to better classification but as we desire to minimize correspondence error, we should switch objectives for the last few levels ($L = 15$).

Evaluation of correspondence prediction: To train our discriminative model, we use third-party software to generate synthetic images. For each image, we randomize the model parameters between reasonable limits and render a synthetic depth image, an image with part annotations and an image that encodes ground truth correspondences. We synthesize $10,000$ images of size 320×240 and sample 2000 pixels from each to train the random forest. Features at each split node are selected from a pool of 5000 random features. Our final forest consists of 3 trees of depth 20.

As described in Section 3, we train our random forest with a combined objective (12), which includes a classification term for the coarser part labels and a regression term for the finer correspondence labels. Intuitively, the signal that the classification objective provides, helps regularize the tree in the initial levels, which implicitly isolates the multiple modes of the distribution of the regression labels. This in turn makes the regression objective more effective in the deeper levels of the tree. Figure 2 shows that the hybrid objective function performs better than using a pure classification or pure regression objective.

Evaluation of model fitting strategy: Our generative model uses a total of 50 blend shapes to represent the face. These include $N_{\text{identity}} = 40$ and $N_{\text{expression}} = 10$. The silver and golden energy described in Section 2 allows us to evaluate the fit of our generative model to the observed depth data. In addition, we are specifically interested in the inference of the expression weights $\{\beta_i\}_{i=1}^{N_{\text{expression}}}$ and data model correspondences $U' = \{u'_i\}_{n=1}^{N_{fg}}$. In a single image, we measure the expression error as

$$e_{expression} = \sum_{i=1}^{N_{\text{expression}}} (\beta_i - \beta_i^{\text{gt}})^2 \,. \tag{15}$$

For a single foreground pixel, we measure the correspondence error as

$$e_{correspondence} = \|S(u;\theta_0) - S(u^{gt};\theta_0)\| \tag{16}$$

where θ_0 is simply the parameter setting that yields the undeformed base mesh model.

Evaluation of silver energy optimization: We begin by demonstrating how optimizing the proxy objective (9) allows us to rapidly reduce the error in (7), the silver energy. This is demonstrated in panel (a) of Figure 3, where we can see a significant decrease as the first ICP step corrects the errors in the forest predictions (see panel (c)). As further ICP steps are taken, the model parameters slowly adjust in tandem so that more accurate correspondences can be acquired (see panel (c)). As expected, there is high correlation between the silver and golden energies, and we manage to greatly decrease the golden energy by minimizing the proxy silver energy, which can be seen in panel (b). Not surprisingly, a better fit of our model also allows us to acquire more accurate expressions as shown in panel (d). Again, the key result here is that by optimizing (9) we are able to simultaneously drive down all relevant energies and errors.

Evaluation of golden energy optimization: We now analyze the ability of our complete hybrid method that refines the result of the previous section by directly optimizing the golden energy with PSO. This is summarized in Figure 4, where it can be seen that 10 iterations of PSO brings us substantially further down in energy. In panel (c) it can be seen that the expression error actually increases after 10 iterations of PSO. In panel (d), we see that the expression error lowers again if we continue to 100 iterations. This is not unexpected behavior, as the blend shape regularization contained in the silver energy is not contained in the golden energy. Although it is helpful to quickly get to a stable and robust result, PSO takes a bit of time to undo this overfitting. Interestingly, a better fit to the data only translates into a moderately better correspondence error (panel (b)) but closer inspection shows that the error actually does drop significantly in key regions of the face. This is demonstrated in Table 1 shows that the correspondence error in the important mouth and nose regions is reduced considerably.

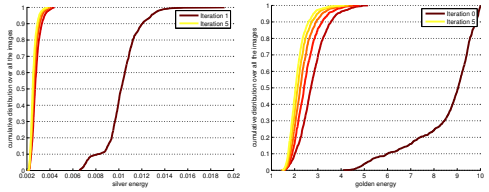
Qualitative results on real data: We demonstrate qualitative results on real data in Figure 5 and supplementary material. In panel (e) we can see the reconstructed face model and in panel (d) the final set of correspondences. To demonstrate the accuracy of the correspondence field, we use our method to extract textures from users' faces and also paint on these to create visual effects. This can be done in real-time ($>25Hz$), as shown in the performance section of the supplementary material and accompanying video.

6. Conclusion

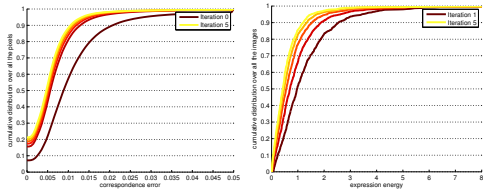
We presented a real-time algorithm for fitting a complex but generic face model to a single depth image of an arbitrary face. In addition to the fit model, we also are able to infer the expression weights and a dense data-model correspondence field. Our system is real-time allowing 3D facial shape and expressions to be used for interactive scenarios such as retargeting and retexturing. In addition, we demonstrate em-

	forehead	eye	temple	cheek	ear	nose	mouth	jaw	average
RF	14.453	9.274	12.871	9.575	14.687	9.634	11.141	9.551	10.554
ICP 1	5.086	4.637	4.934	5.037	4.546	5.413	6.017	5.227	5.079
ICP 5	3.824	3.761	4.268	4.215	3.747	4.676	5.261	4.613	4.299
PSO 1	3.729	3.673	4.252	4.120	3.803	4.476	5.145	4.705	4.260
PSO 5	3.600	3.560	4.266	3.965	3.829	4.190	4.828	4.704	4.151
PSO 10	3.580	3.528	4.308	3.926	3.825	4.134	4.732	4.667	4.115

Table 1: Average correspondence error over facial parts at different stages of the pipeline. The error is given in millimeters.

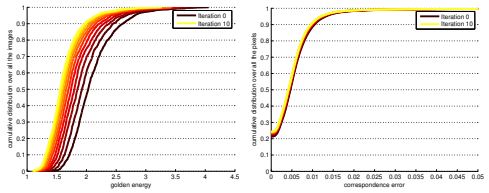


(a) Silver energy (b) Golden energy

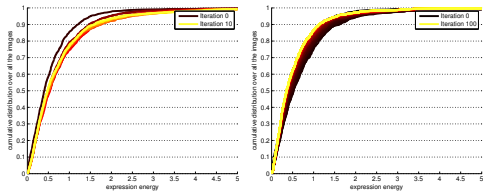


(c) Corr. error (d) Expression error

Figure 3: Cumulative distribution of (a) silver energy, (b) golden energy, (c) correspondence error, and (d) expression error at different iterations of ICP. ICP procedure reduces all the error measures while optimizing over the silver energy.



(a) Golden energy (b) Corr. error



(c) Expression error (d) Exp. error (100)

Figure 4: Cumulative distribution of (a) golden energy, (b) correspondence error, (c) expression error at different iterations of PSO, and (d) expression error up to 100 iterations of PSO.

pirically how the various components of our algorithm drive down the "golden energy", which we argue is the natural energy to minimize. We show that this energy is highly correlated with the other two quantities that we are interested in estimating, namely the correspondence error and expression error. Unlike other related methods, our method relies only on per-frame depth, avoiding tracking failures due to fast motions, working in low-lighting conditions, and removing the need for per-user calibration. Moreover, our discriminative pipeline estimates a dense correspondence field, making it more robust than methods that rely on a small number of

Figure 5: Qualitative results on real data captured using Kinect camera. From left to right, we show the input depth data, the depth data overlaid on the reconstructed model, the reconstructed model and the parts overlaid on the rgb image (which was only used for visualization). Some of these examples are from [9].

landmarks which can easily be occluded.

Naturally, our method is only robust to moderate occlusions. This is due to our use of a truncated loss function in the final optimization of the golden energy and due to the locality of forest features. The latter helps the forest provide a good enough initialization to the (largely local) optimization of the former. Larger occlusions are not currently handled, but could be alleviated by synthesizing occluded faces for training. Additionally, as with other machine learning techniques, we are of course limited by the expressiveness of our model and the variety of our training data. Improving the richness of our model and handling occlusions remains interesting areas of future work.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. https://code.google.com/p/ceres-solver/. 4

[2] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R. W. Sumner, and M. Gross. High-quality passive facial performance capture using anchor frames. In *TOG*, 2011. 2

[3] M. J. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *ICCV*, 1995. 2

[4] V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. In *Computer graphics forum*, 2003. 2

[5] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of SIGGRAPH*, 1999. 2

[6] S. Bouaziz, Y. Wang, and M. Pauly. Online modeling for realtime facial animation. *TOG*, 2013. 2

[7] D. Bradley, W. Heidrich, T. Popa, and A. Sheffer. High resolution passive facial performance capture. *TOG*, 2010. 2

[8] C. Cao, Y. Weng, S. Lin, and K. Zhou. 3d shape regression for real-time facial animation. *TOG*, 2013. 2

[9] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: a 3d facial expression database for visual computing. 2013. 7

[10] J. Chai, J. Xiao, and J. Hodgins. Vision-based control of 3d facial animation. In *Computer Animation*, 2003. 2

[11] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 2002. 5

[12] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *ECCV*, 1998. 2

[13] S. Corazza, L. Mündermann, E. Gambaretto, G. Ferrigno, and T. P. Andriacchi. Markerless motion capture through visual hull, articulated icp and subject specific model generation. *IJCV*, 2010. 4

[14] K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister. Video face replacement. In *TOG*, 2011. 2

[15] M. Dantone, J. Gall, G. Fanelli, and L. J. V. Gool. Real-time facial feature detection using conditional regression forests. In *CVPR*, 2012. 2

[16] A. Delaunoy and E. Prados. Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3d reconstruction problems dealing with visibility. *IJCV*, 2011. 3

[17] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool. Random forests for real time 3d face analysis. *IJCV*, 2013. 2

[18] Y. Furukawa and J. Ponce. Dense 3d motion capture for human faces. In *CVPR*, 2009. 2

[19] P. Garrido, L. Valgaerts, C. Wu, and C. Theobalt. Reconstructing detailed dynamic face geometry from monocular video. *TOG*, 2013. 2

[20] R. B. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. W. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV*, 2011. 4, 5

[21] B. Glocker, O. Pauly, E. Konukoglu, and A. Criminisi. Joint classification-regression forests for spatially structured multi-object segmentation. In *ECCV*. Springer, 2012. 1

[22] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *TOG*, 1998. 2

[23] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 1987. 5

[24] H. Huang, J. Chai, X. Tong, and H.-T. Wu. Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition. *TOG*, 2011. 2

[25] P. J. Huber et al. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 1964. 5

[26] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 2014. 2

[27] I. Kemelmacher-Shlizerman and R. Basri. 3d face reconstruction from a single image using a single reference face shape. *PAMI*, 2011. 2

[28] J. Kennedy, R. Eberhart, et al. Particle swarm optimization. In *IEEE Neural Networks*, 1995. 3

[29] H. Li, B. Adams, L. J. Guibas, and M. Pauly. Robust single-view geometry and motion reconstruction. *TOG*, 2009. 2

[30] H. Li, P. Roivainen, and R. Forchheimer. 3-d motion estimation in model-based facial image coding. *PAMI*, 1993. 2

[31] H. Li, J. Yu, Y. Ye, and C. Bregler. Realtime facial animation with on-the-fly correctives. 2013. 2

[32] K. Li, F. Xu, J. Wang, Q. Dai, and Y. Liu. A data-driven approach for facial expression synthesis in video. In *CVPR*, 2012. 2

[33] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 1963. 4

[34] F. Pighin and J. Lewis. Performance-driven facial animation. In *TOG*, 2006. 2

[35] G. Pons-Moll, J. Taylor, J. Shotton, A. Hertzmann, and A. Fitzgibbon. Metric regression forests for human pose estimation. In *BMVC*, 2013. 1, 2, 4

[36] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*, 2011. 1, 2, 4, 5

[37] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian Manifold: Inferring dense correspondences for one-shot human pose estimation. In *CVPR*, 2012. 1, 2, 4, 5

[38] L. Valgaerts, C. Wu, A. Bruhn, H.-P. Seidel, and C. Theobalt. Lightweight binocular facial performance capture under uncontrolled lighting. *TOG*, 2012. 2

[39] D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. In *TOG*, 2005. 2

[40] Y. Wang, X. Huang, C.-S. Lee, S. Zhang, Z. Li, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang. High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. In *Computer Graphics Forum*, 2004. 2

[41] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Realtime performance-based facial animation. *TOG*, 2011. 2

[42] T. Weise, H. Li, L. Van Gool, and M. Pauly. Face/off: Live facial puppetry. In *Computer Animation*. ACM, 2009. 2

[43] L. Williams. Performance-driven facial animation. 1990. 2

[44] J. Xiao, S. Baker, I. Matthews, and T. Kanade. Real-time combined 2d+3d active appearance models. In *CVPR*, 2004. 2