Online Generative Model Personalization for Hand Tracking

ANASTASIA TKACH*, École Polytechnique Fédérale de Lausanne
ANDREA TAGLIASACCHI*, University of Victoria
EDOARDO REMELLI, École Polytechnique Fédérale de Lausanne
MARK PAULY, École Polytechnique Fédérale de Lausanne
ANDREW FITZGIBBON, Microsoft Hololens Research (Cambridge)

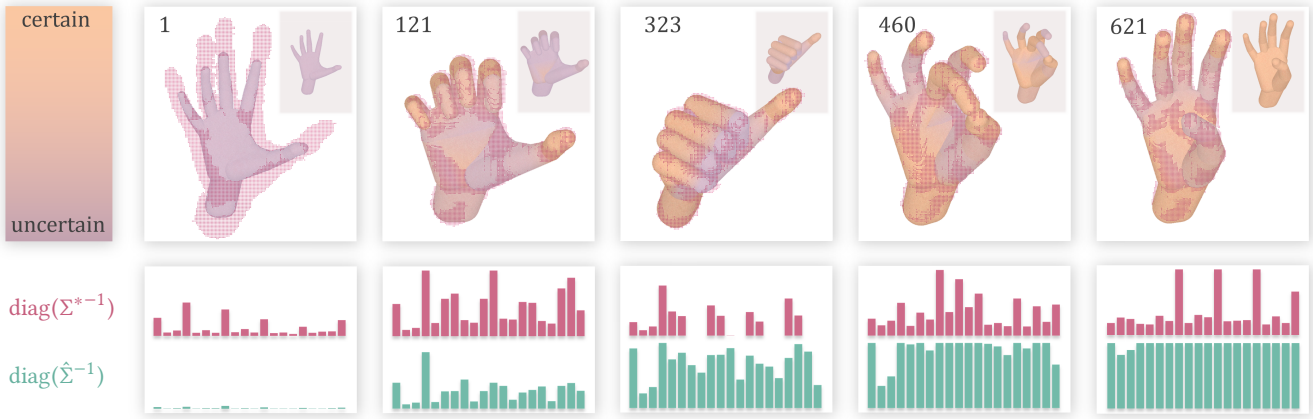

Fig. 1. Our adaptive hand tracking algorithm optimizes for a tracking model on the fly, leading to progressive improvements in tracking accuracy over time. **Above**: hand surface color-coded to visualize the spatially-varying confidence of the estimated geometry. Insets: color-coded *cumulative* certainty. Notice how in the last frame all parameters are certain. **Below**: histograms visualize the certainty of each degree of freedom, that is, the diagonal entries of the inverse of the covariance estimate from: (a) data in the current frame Σ^*, or (b) the information $\hat{\Sigma}$ accumulated through time by our system.

We present a new algorithm for real-time hand tracking on commodity depth-sensing devices. Our method does not require a user-specific calibration session, but rather learns the geometry as the user performs live in front of the camera, thus enabling seamless virtual interaction at the consumer level. The key novelty in our approach is an online optimization algorithm that jointly estimates pose and shape in each frame, and determines the uncertainty in such estimates. This knowledge allows the algorithm to integrate per-frame estimates over time, and build a personalized geometric model of the captured user. Our approach can easily be integrated in state-of-the-art continuous generative motion tracking software. We provide a detailed evaluation that shows how our approach achieves accurate motion tracking for real-time applications, while significantly simplifying the workflow of accurate hand performance capture. We also provide quantitative evaluation datasets at http://gfx.uvic.ca/datasets/handy

CCS Concepts: • **Computing methodologies** → **Tracking**;

ACM Reference Format:
Anastasia Tkach, Andrea Tagliasacchi, Edoardo Remelli, Mark Pauly, and Andrew Fitzgibbon. 2017. Online Generative Model Personalization for Hand Tracking. *ACM Trans. Graph.* 36, 6, Article 243 (November 2017), 11 pages. https://doi.org/10.1145/3130800.3130830

*Both authors contributed equally to the paper

1 INTRODUCTION

In our everyday life we interact with the surrounding environment using our hands. A main focus of recent research has been to bring such interaction to virtual objects, such as the ones projected in virtual reality devices, or super-imposed as holograms in AR/MR headsets. Performance capture is also essential in film and game production for pre-visualization, where motion can be transferred in real-time to a virtual avatar. This allows directors to plan shots more effectively, reduce turn-around times and hence costs. For these applications, it is desirable for the tracking technology to be robust, accurate, and have a *seamless deployment*, since performance capture can happen at an animator's desk, on a movie set, or even "in the wild", where the user might not be aware of its operative requirements (e.g. advertising or security).

Hand tracking from monocular depth. Recent developments in hand motion capture technology have brought us a step closer to achieving effective tracking, where hardware solutions such as data-gloves, reflective markers and multi-camera setups have been shelved due to their invasiveness and cumbersome setup. Hence, a single camera has become the standard acquisition device, where depth cameras have taken a solid lead over color imagery to overcome the many challenges of hand-tracking [Supancic et al. 2015]. Modern techniques (e.g. Taylor et al. [2016]) often rely on *discriminative* techniques (e.g. Valentin et al. [2016]) to identify a coarse

pose, followed by a *generative* stage (e.g. Tkach et al. [2016]) to refine the alignment and obtain a precise pose estimate.

Tracking templates and personalization. Since depth imagery provides incomplete 3D data of the tracked object, generative trackers attempt to register a geometric *template*, also referred to as a *tracking model*, to 3D data so to minimize alignement residuals. The more accurately a model fits the observed user, the better tracking accuracy can be achieved [Taylor et al. 2016; Tkach et al. 2016]. The process of accurately generating a *user-specific tracking model* from input data is referred to in the literature as *calibration* or *personalization*. Calibrating a template from a set of static poses is a standard component in the workflow of facial performance capture [Cao et al. 2015; Weise et al. 2011], and the work of Taylor et al. [2014] pioneered it within the realm of hand tracking. However, current methods such as [Taylor et al. 2016] and [Tkach et al. 2016] suffer a major drawback: the template must be created during a controlled calibration stage, where the hand is scanned in several static poses (i.e. *offline*). While appropriate for professional use, a calibration session is a severe drawback for seamless deployment in consumer-level applications. Therefore, inspired by recent efforts in facial performance capture that calibrate templates while tracking [Bouaziz et al. 2013; Li et al. 2013], in this paper we propose a pipeline for *online* model calibration. The approach we present has been tailored to monocular acquisition, where we tackle the significant technical challenges created by missing data due to self-occlusions.

Contributions. Our core contribution is a principled way to integrate per-frame information into an online real-time pose/shape tracking algorithm: one that estimates the hand's *pose*, while simultaneously refining its *shape*. That is, as more of the user's hand and articulation is observed during tracking, the more the tracking template is progressively adapted to match the performer, which in turns results in more accurate motion tracking. From a single frame only a subset of the shape degrees of freedom can be estimated, for example, it is difficult to estimate the length of a phalanx when observing a straight finger. Our technique automatically estimates the *confidence* in per-frame parameter computations, and leverages this information to build a tracking model that selectively *accumulates* confident parameter estimates over time. Assuming a reasonable performance by the user, our system typically constructs a fully calibrated model within a few seconds, while simultaneously tracking the user in real time. Perhaps more importantly, however, if the user is "unreasonable", holding his/her hand in an ambiguous pose (e.g. fingers unbent), the system maintains its shape uncertainty until a constraining pose is adopted. The key technical component of our solution is a recent tool from control theory – the Levenberg-Marquardt Kalman Filter (LMKF) of [Skoglund et al. 2015]. Although it has long been known [Bell and Cathey 1993; Bellaire et al. 1995] that there are strong links between Levenberg-style algorithms and the Kalman filter, and that Kalman filters are useful to maintain uncertainty in visual tracking and SLAM [Strasdat et al. 2012], only recently have the advantages of both views been combined. This paper shows, in both qualitative and quantitative performance evaluations, that the LMKF enables practically useful online calibration. Overall, our solution yields a fully automatic, real-time hand tracking system that is well-suited for consumer applications.

2 RELATED WORKS

Due to the vast amount of work in the area of body, face and hand tracking we respectively refer the reader to the recent works of Bogo et al. [2015], Cao et al. [2016] and Taylor et al. [2016] for a complete overview. In this section we focus our attention on generative hand tracking and model calibration. Model personalization is a core ingredient in generative motion tracking [Pons-Moll and Rosenhahn 2011]. Due to the large number of hand self-occlusions, the low signal-to-noise ratio in current depth sensors, a globally unconstrained pose, and the similar appearance of fingers make the personalization of a hand model a harder problem than face or body model calibration; see [Supancic et al. 2015].

Offline model calibration. Albrecht et al. [2003] pioneered the construction of realistic (skin, bone and muscles) personalized models. They proposed a pipeline for the registration of a 3D mesh model to RGB data manually pre-processed by the user. Reducing the amount of manual interaction required from the user, Rhee et al. [2006] showed how skin creases and silhouette images can also be used to guide the registration of a model to color imagery. Taylor et al. [2014] introduced a more automatic pipeline, generating personalized hand models from input depth sequences where the user rotates his hand while articulating fingers. More closely related to ours is the work by Tan et al. [2016]. They show how to robustly personalize a hand model to an individual user from a set of depth measurements using a trained shape basis such the one proposed by Khamis et al. [2015]. The calibration pipeline, although robust and efficient, is not fully automated as the user needs to *manually* pick the set of frames over which the calibration optimization is performed. In facial calibration, Weise et al. [2011] asked users to assume a set of standard facial expressions to match standard poses in the Facial Action Coding System (FACS) of Ekman and Friesen [1977]. Inspired by these approaches, Taylor et al. [2016] recently proposed an analogous offline hand calibration method, but the question "which is the set of *optimal* hand poses that allows to properly capture the hand's geometry?" has yet to be addressed. Hence, none of the above methods is suitable or easily adaptable to the kind of consumer-level applications that we target.

Online model calibration. In [de La Gorce et al. 2011], the authors introduced a (non real-time) model-based approach for hand tracking from a monocular RGB video sequence. Hand pose, texture and lighting are dynamically estimated, while shape is determined by optimizing over the first frame only. Recently Makris and Argyros [2015] proposed a model-based approach to jointly solve the pose tracking and shape estimation problem from depth measurements in an online framework. They solve for the cylindrical geometry of a hand through render-and-compare evaluations over a set of frames with particle swarm optimization (PSO). Their pipeline runs in real-time (30fps), but lacks the degree of robustness and accuracy desirable for consumer level applications, and does not address uncertainty. More sophisticated approaches to information agglomeration such as the ones for face tracking/modeling by Bouaziz et al.

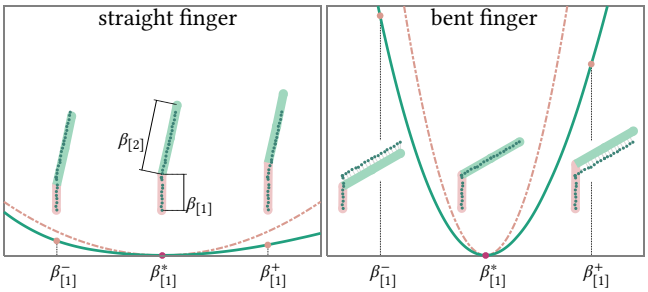

Fig. 2. **(Per-frame regression)** We abstract the hand shape/pose estimation problem from a single frame into the one of a simpler 2D stick-figure. Note, however, that this illustration is not hand-crafted, but is derived from numerical optimization executed on these simplified datasets. When the finger is straight (left), it is difficult to estimate the length of individual phalanges as the optimization problem is ill-posed. With a bent finger (right) the problem is better conditioned. We analyze the landscape of the registration energy $E(\beta_{[1]})$, and observe how estimation uncertainty relates to the width of the local minima valley. This uncertainty, the posterior distribution of shape parameters after computing their estimate from the data in the current frame, can be estimated through a quadratic approximation $\tilde{E}(\beta_{[1]})$, derived from the Hessians of the registration energies.

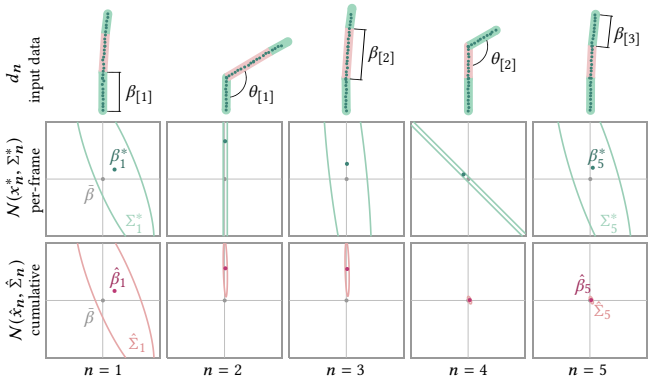

Fig. 3. **(Cumulative regression)** We visualize several temporally sorted frames of input data d_n, the uncertainty ellipsoid Σ^*_n estimated by per-frame regression, and the *online* uncertainty estimate $\hat{\Sigma}_n$. For illustration purposes, we only display the two-dimensional ellipsoids representing the covariance of $\beta_{[1]}$ and $\beta_{[2]}$. Although $\Sigma^*_1 = \Sigma^*_5$, observe how $\hat{\Sigma}_1 \succ \hat{\Sigma}_5$: in the last frame we have a confident estimate as the information from frames $2:4$ has been integrated. Further, notice how even though the parameter $\beta_{[2]}$ was not observed directly in any of the presented frames, its value was inferred from the highly-certain measurements $(\beta_{[1]})_{n=2}$ and $(\beta_{[1]} + \beta_{[2]})_{n=4}$.

[2013], Li et al. [2013] and Thies et al. [2015], where shape estimation is performed over the whole set of frames, allow to obtain more accurate results, while guaranteeing real-time performances. Thies et al. [2015] jointly optimize face identity and expression during calibration stage and keep identity fixed during tracking. The work of Zou and Tan [2013], although in a different applicative domain, is also related to ours, as they solve for SLAM by considering uncertainties when aggregating information through time. Gu et al. [2017] propose a holistic approach for aggregating per-frame measurements. They demonstrate how an LSTM layer in a CNN allows to maintain an online estimate that surpasses the performance of a more standard kalman filter approach.

Online algorithms offer other key advantages compared to offline methods: (1) the ability to offer immediate *feedback* to the user on the quality of the result [Izadi et al. 2011], (2) the potential to dynamically adapt to transparently *hot-swap* users [Bouaziz et al. 2013], and (3) reduced storage and computational resources, as information is integrated frame-by-frame, in a *streaming* fashion.

3 ONLINE MODEL CALIBRATION

We now describe our *joint* calibration and tracking algorithm, which combines the Levenberg-style optimization of previous hand trackers with the uncertainty maintenance framework of Kalman filtering. Previous hand tracking work has made use of temporal smoothing priors to propagate *pose* information from previous frames, without the use of filtering. However this approach cannot be used for *shape* because it is so weakly constrained in any given frame, and because its temporal prior is so strong, as shape parameters are *persistent* over time: we observe the same user performing in front of the camera for thousands of frames. However, sufficient information to estimate certain shape parameters is simply not available in certain frames. For example, by observing a straight finger like the one in Figure 2-(left), it is difficult to estimate the length of a phalanx. Therefore, knowledge must be gathered from a *collection* of frames capturing the user in different poses.

As we illustrate in Figure 2 and Figure 4, the confidence in regressed *shape* parameters is conditional on the *pose* of the current frame. Rather than manually picking a few frames in different poses as in [Taylor et al. 2016], we show how propagation of not just the shape estimate, but also its uncertainty allows reliable calibration even if the initial poses fail entirely to constrain some shape dimensions. Additionally, the temporal priors of previous work are easily incorporated in the LMKF formulation.

Input data and shape model. The input data are a sequence of depth frames $\mathcal{D}_n$, which are segmented via a wristband [Tagliasacchi et al. 2015] to produce a point cloud $d_n \subset \mathbb{R}^3$. The pose vector in frame n is θ_n, and our shape model $\mathcal{M}(\theta;\beta)$ is the sphere mesh of Tkach et al. [2016]. Shape is encoded via scalar length parameters β instead of sphere positions; see Figure 10 and [Remelli et al. 2017].

Estimation. Let $x_n = [\theta_n; \beta_n]$ denote the model *state*: the vector of coalesced pose and shape parameters at frame n. Our goal in tracking is to produce the best estimate $\hat{x}_n$, at frame n, of the state x_n,

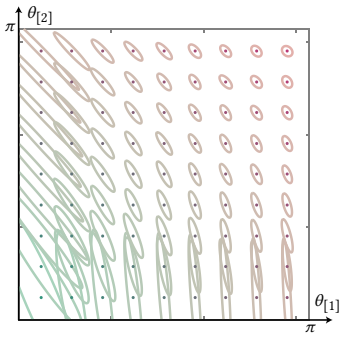

Fig. 4. A visualization of the covariance estimate for phalanx lengths $\{\beta_{[1]}, \beta_{[2]}\}$ as we vary phalanx bend angles $\{\theta_{[1]}, \theta_{[2]}\}$. A confident measurement of $\beta_{[1]}$ is only available when $\theta_{[1]}$ is bent, while a confident measurement of $\beta_{[1]} + \beta_{[2]}$ is available when $\theta_{[2]}$ is bent. The covariance ellipsoids are centered at the corresponding $\{\theta_{[1]}, \theta_{[2]}\}$ location.

$$\hat{x}_n = \arg\max_{x_n} \underbrace{\log\left(p(x_n^*|x_n)\, p(x_n|\hat{x}_{n-1})\right)}_{L(x_n)}$$

$$p(x_n^*|x_n) = \exp\left(-\tfrac{1}{2}(x_n^* - x_n)^T {\Sigma_n^*}^{-1}(x_n^* - x_n)\right)$$

$$p(x_n|\hat{x}_{n-1}) = \exp\left(-\tfrac{1}{2}(x_n - \hat{x}_{n-1})^T \hat{\Sigma}_{n-1}^{-1}(x_n - \hat{x}_{n-1})\right)$$

$$\hat{\Sigma}_n^{-1} = \left.\frac{\partial^2 L}{\partial x_n^2}\right|_{\hat{x}_n} \approx \begin{bmatrix} \sqrt{{\Sigma_n^*}^{-1}} \\ \sqrt{\hat{\Sigma}_{n-1}^{-1}} \end{bmatrix}^T \begin{bmatrix} \sqrt{{\Sigma_n^*}^{-1}} \\ \sqrt{\hat{\Sigma}_{n-1}^{-1}} \end{bmatrix} = {\Sigma_n^*}^{-1} + \hat{\Sigma}_{n-1}^{-1}$$

Table 1. *Split* cumulative regression – Kalman Filter (KF)

$$\hat{x}_n = \arg\max_{x_n} \underbrace{\log\left(p(d_n|x_n)\, p(x_n|\hat{x}_{n-1})\right)}_{L(x_n)}$$

$$p(d_n|x_n) = \exp\left(-\tfrac{1}{2}(d_n - F(x_n))^T (d_n - F(x_n))\right)$$

$$p(x_n|\hat{x}_{n-1}) = \exp\left(-\tfrac{1}{2}(x_n - \hat{x}_{n-1})^T \hat{\Sigma}_{n-1}^{-1}(x_n - \hat{x}_{n-1})\right)$$

$$\hat{\Sigma}_n^{-1} = \left.\frac{\partial^2 L}{\partial x_n^2}\right|_{\hat{x}_n} \approx \begin{bmatrix} -\frac{\partial F(\hat{x}_n)}{\partial x_n} \\ \sqrt{\hat{\Sigma}_{n-1}^{-1}} \end{bmatrix}^T \begin{bmatrix} -\frac{\partial F(\hat{x}_n)}{\partial x_n} \\ \sqrt{\hat{\Sigma}_{n-1}^{-1}} \end{bmatrix} = \Sigma_n^{-1} + \hat{\Sigma}_{n-1}^{-1}$$

Table 2. *Joint* cumulative regression – Iterated Extended KF (IEKF)

given all the data seen previously, $d_1, ..., d_n$. Additionally, we want to estimate not just the state, but the parameters of the *probability distribution* over the state $p(x_n|d_{1..n})$. Thus, if we write

$$p(x_n|d_{1..n}) \approx \mathcal{N}(x_n \mid \hat{x}_n, \hat{\Sigma}_n), \tag{1}$$

we are saying that x_n approximately follows a normal distribution with mean $\hat{x}_n$ and covariance $\hat{\Sigma}_n$. When we display a tracked hand to the user, we will most likely just draw the hand with pose and shape parameters $\hat{x}_n$, which sometimes leads to $\hat{x}_n$ being called "the estimate of x_n", but it is more correctly "the estimate of the mean of the distribution $p(x_n)$", and similarly with $\hat{\Sigma}_n$.

It is generally computationally intractable to estimate the parameters conditioned on all the previous history $d_{1..n}$ at every frame (although in Section 3.4 we compute some related quantities as a baseline), so the estimation is typically expressed in terms of an *per-frame* term $p(x_n|d_n)$, which describes the components due only to information in frame d_n and *cumulative* term $p(x_n|d_1, ..., d_{n-1})$. Different approximations for this term lead to different methods, denoted *split cumulative* and *joint cumulative* below.

3.1 Per-frame estimate – $p(x_n|d_n)$

The distribution $p(x_n|d_n)$ is, by Bayes' rule, proportional to the product of a data term and a prior $p(d_n|x_n)p(x_n)$, which is naturally related to the traditional energy formulations by identifying the negative log likelihood with the energy. Consider the energy:

$$E(x_n) = \sum_{\tau \in \mathcal{T}} E_\tau(d_n, x_n) \tag{2}$$

Where the terms $\mathcal{T}$ ensure that:

d2m	data points are explained by the model
m2d	model lies in the sensor visual-hull
smooth	recorded sequence is smooth
pose-prior	calibrated hand pose is likely
shape-prior	calibrated hand shape is likely
pose-valid	semantics: collisions and joint limits
shape-valid	semantics: finger order and connectivity

The energy terms in the objective function are detailed in [Tkach et al. 2016] and [Tagliasacchi et al. 2015], with the exception of *shape-prior* and *shape-valid* that are discussed in Section 3.5. Given E as above, we can write

$$p(x_n|d_n) \propto \exp(-E(x_n)), \tag{3}$$

but to perform propagation, we will need a more compact form, for example a Gaussian approximation. A natural choice is the *Laplace approximation*: a Gaussian with its mean at the mode of (3) (see Appendix A.1) and covariance chosen to match a second-order expansion of E about that mode. The mode computation is the standard energy minimization

$$x_n^* = \arg\min_{x_n} E(x_n) \tag{4}$$

which can be solved by nonlinear optimization given an initialization x_n^0 (obtained from a discriminative method or from the solution of the previous time-step), and indeed this is the same minimization performed by current state-of-the-art hand trackers. The covariance matrix Σ_n^* of the Laplace approximation is the inverse of the Hessian of E, and as we are using a Gauss-Newton solver, $E(x)$ is of the form $\|d_n - F(x_n)\|^2$, so we may make the G-N approximation of the Hessian in terms of the Jacobian of $\bar{F}(x_n) = d_n - F(x_n)$, yielding

$$\Sigma_n^* = \left(\frac{\partial \bar{F}(x^*)}{\partial x}^\top \frac{\partial \bar{F}(x^*)}{\partial x}\right)^{-1}. \tag{5}$$

Thus, after processing the information in a frame d_n, the sought-after quadratic approximation of posterior distribution of model parameters is

$$\tilde{p}(x_n|d_n) \approx \mathcal{N}\left(x_n^*, \Sigma_n^*\right), \tag{6}$$

so the "per-frame" posterior parameters are $\hat{x}_n = x_n^*, \hat{\Sigma}_n = \Sigma_n^*$.

3.2 Split cumulative estimate – $p(x_n|d_{1..n})$

The per-frame distribution in Section 3.1 encodes the uncertainty in pose and shape solely due to the data in frame n. To aggregate information from previous frames, we would like a simple form of the distribution $p(x_n|d_{1..n})$, for example a Gaussian:

$$p(x_n|d_{1..n}) \approx \mathcal{N}(\hat{x}_n, \hat{\Sigma}_n) \tag{7}$$

Then, given values of the parameters $\hat{x}_{n-1}, \hat{\Sigma}_{n-1}$ at the previous timestep, we must update them to incorporate the new information in frame n. This leads to the following pair of inductive update equations:

$$\mathcal{N}(x_n|\hat{x}_1, \hat{\Sigma}_1) = \mathcal{N}(x_n|x_1^*, \Sigma_1^*) \tag{8}$$

$$\mathcal{N}(x_n|\hat{x}_n, \hat{\Sigma}_n) = \mathcal{N}(x_n|\hat{x}_{n-1}, \hat{\Sigma}_{n-1})\mathcal{N}(x_n|x_n^*, \Sigma_n^*) \tag{9}$$

Fig. 5. We evaluate our real-time calibration framework on twelve different subjects. For each user we show a frame in a (more-or-less) rest pose configuration, as well as a different pose selected from the recorded sequence. These results are better appreciated by watching our Video [04:03].

By applying the product of Gaussians rule [Petersen et al. 2008], we obtain update equations for $\hat{x}_n$ and $\hat{\Sigma}_n$:

$$\begin{aligned}\hat{x}_n &= \Sigma_n^*(\hat{\Sigma}_{n-1} + \Sigma_n^*)^{-1}\hat{x}_{n-1} + \hat{\Sigma}_{n-1}(\hat{\Sigma}_{n-1} + \Sigma_n^*)^{-1}x_n^* \\ \hat{\Sigma}_n &= \hat{\Sigma}_{n-1}(\hat{\Sigma}_{n-1} + \Sigma_n^*)^{-1}\Sigma_n^* = \left(\Sigma_n^{*-1} + \hat{\Sigma}_{n-1}^{-1}\right)^{-1}\end{aligned} \tag{10}$$

In Appendix B.1, we shown how Equation 10 is equivalent to the Kalman Filter (KF) update equations in Table 1, with measurement x_n^*, and measurement noise covariance Σ_n^*. This optimization, which we refer to as *split cumulative* is arguably the simplest way of achieving an online parameter regression: by treating the results of the per-frame solve $\mathcal{N}(x_n^*, \Sigma_n^*)$ as the measurements in a KF.

3.3 Joint cumulative estimate – $p(x_n|d_{1..n})$

The optimization in Tab. 1 does not provide any information about the current estimate of the parameters $\hat{x}_n$ to the independent solve described in Section 3.1. This could be problematic, as in this case Eq. 2 does not leverage any temporal information aside from initialization, while relying on a sufficiently good initialization to compute $\mathcal{N}(x_n^*, \Sigma_n^*)$. We propose to coalesce the cumulative and per-frame optimization resulting in the *joint* cumulative regression scheme in Table 2. The optimization in Table 2 can be expressed in least-squares form, and embedded in Equation 2 through the term:

$$E_{\text{iekf}} = \|\hat{\Sigma}_{n-1}^{-1/2}(x_n - \hat{x}_{n-1})\|_2^2 \tag{11}$$

In Appendix B.2 we link this update to LM [Skoglund et al. 2015], demonstrating that optimizing the objective in Table 2 with a Levenberg Marquardt method is equivalent to an Iterated Extended Kalman Filter (IEKF) with measurement update d_n. In a practical setting, this observation creates a very simple way to encode IEKF-like behavior within existing LM optimization codebases.

3.4 Joint multiframe (batch/offline) estimate

While we focus on an online/streaming algorithm, we also describe an offline baseline calibration procedure – inspired by the work of [Taylor et al. 2014] – where multiple frames in the input sequence are simultaneously considered.

Offline-Hard. To achieve this, Equation 2 is modified to consider N frames, each with its own pose parameters θ_n, but with the same underlying shape β, resulting in what we refer to as *Offline-Hard* calibration:

$$\underset{\beta,\{\theta_n\}}{\arg\min} \sum_{n=1}^{N} \sum_{\tau\in\mathcal{T}} E_\tau(d_n, [\theta_n, \beta]) \tag{12}$$

Such optimization is initialized with a single β^0, and in our experiments, we noticed how this resulted in reduced convergence performance and a propensity for the optimization to fall into local minima.

Offline-Soft. Therefore, we introduce the *Offline-Soft* calibration, where the constraint that a single β should be optimized is enforced through a soft penalty:

$$\underset{\beta,\{\theta_n\}}{\arg\min} \sum_{n=1}^{N} \sum_{\tau\in\mathcal{T}} E_\tau(d_n, [\theta_n, \beta_n]) + \omega_\beta \sum_{n=1}^{N} \|\beta_n - \beta\|^2 \tag{13}$$

The initializations β_n^0 are derived from the per-frame optimization of Equation 2, while the penalty weight is set to a large value ($\omega_\beta = 10e4$). The advantage of Offline-Soft over Offline-Hard can be clearly observed in Figure 8, where the former achieves a performance comparable to the one of the (overfitting) per-frame optimization. Finally, note that in practice we do not consider every frame as this large problem would not fit into memory, but instead we subsample at a $\approx 1/20$ rate, the same subsampling is used for Kalman *measurement* updates in our online solution to avoid a bias in the comparisons.

3.5 Shape regularizers

Hand shape variation can be explained in a low dimensional space whose fundamental degrees of freedom include variation like uniform scale, palm width, and finger thickness [Khamis et al. 2015].

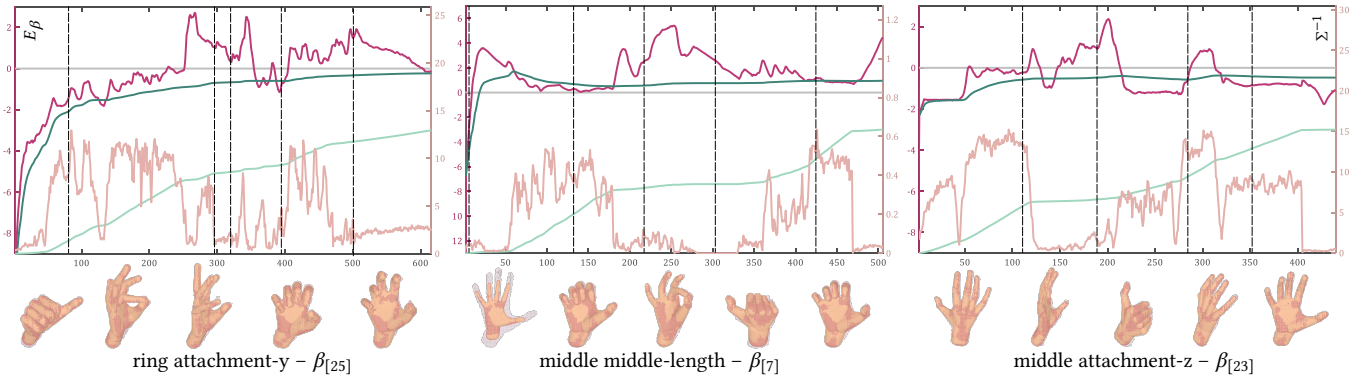

Fig. 6. We illustrate the formulation in Section 3 on the calibration of four different degrees-of-freedom on the Handy/Teaser dataset. (top) Frames are indexed by n and we display: ground-truth value $\bar{\beta}$, per-frame estimate β_n^* and cumulative estimate $\hat{\beta}_n$; the scale for these quantities is relative to $\bar{\beta}$, and shown on left-hand side of each axis. In the same plot, we also visualize the *inverse* of the diagonal entry of the covariance matrices (i.e. certainty) for per-frame Σ_n^* and cumulative $\hat{\Sigma}_n$ estimates; the scale for these quantities is on the right-hand side of each plot. (bottom) We also display the pose corresponding to a selection of frames in the sequence (dashed); please note the correlation between pose and uncertainty.

In our paper we follow the ideas presented in [Remelli et al. 2017], and build a latent-space encoding hand shape variability through *anisotropic scaling*. By setting $\omega_{\text{shape-space}} \ll 1$, this prior acts as a soft regularizer and does not prevent the algorithm from computing a tight fit:

$$E_{\text{shape-space}} = \|\beta - (\bar{\beta} \circ \mathcal{I}\tilde{\beta})\|^2 \tag{14}$$

where $\tilde{\beta} \in \mathbb{R}^3$ is a latent vector encoding relative changes in hand *height*, *width* and sphere *thickness* with respect to the default template $\bar{\beta}$, while $\mathcal{I}$ is a matrix mapping latent DOFs to the corresponding full-dimensional DOFs $\beta_{[i]}$; see Figure 10. Since our shape-prior has a small weight, unfeasible hand-shape configurations are still possible, such as a finger floating in mid-air, or when the natural order of fingers {index, middle, ring, pinky} has been compromised. We overcome this problem by a set of quadratic *barrier* constraints that are conditionally enabled in the optimization when unfeasible configurations are detected (encoded via $\chi_c(\beta) \in \{0, 1\}$):

$$E_{\text{shape-valid}} = \sum_{c=1}^{C} \chi_c(\beta) \| \langle \beta, \kappa \rangle \|_2^2 \tag{15}$$

For example to avoid middle and index fingers from permuting, one such constraint is written in the following form, and $\chi_0(\beta) = 1$ only when an invalid configuration is detected:

$$\chi_0(\beta) \| \beta_{\text{idx-base-x}} - \beta_{\text{idx-base-rad}} - \beta_{\text{mid-base-x}} + \beta_{\text{mid-base-rad}} \|_2^2$$

4 EVALUATION

To evaluate the technical validity of our approach we verify its effectiveness by applying it to a new dataset acquired through commodity depth cameras (Sec. 4.1); corroborate the formulation of our optimization on a synthetic 3D dataset (Sec. 4.2); analyze its robustness through randomly perturbing the algorithm initialization (Sec. 4.3); and attest how our method achieves state-of-the-art performance on publicly available datasets (Sec. 4.4 and Sec. 4.5).

4.1 Calibration dataset: Handy/GuessWho? – Fig. 5

We stress-tested our system by qualitatively evaluating our calibration technique with data acquired from *twelve* different users performing in front of an Intel RealSense SR300 camera (a consumer-level time-of-flight depth sensor). Snapshots of the twelve calibration sequences are reported in Figure 5. While ground truth information is not available, these datasets will enable comparisons to our method through the use of empirical metrics; e.g. E_{d2m} and E_{m2d} [Tkach et al. 2016], or the golden-energy [Taylor et al. 2016].

4.2 Synthetic dataset: formulation analysis – Fig. 6

For synthetic data the ground truth shape parameters $\bar{\beta}$ are readily available, and the sphere-mesh model $\mathcal{M}(\theta, \beta)$ is animated in time with the $\bar{\theta}_n$ parameters of the complex motions in the Handy dataset [Tkach et al. 2016].

The following metric measures average ground-truth residuals (in millimeters):

$$E_\beta = \frac{1}{|M|} \sum_{m \in M} \left| \beta_{[m]} - \bar{\beta}_{[m]} \right| \tag{16}$$

For ground-truth comparisons, analogously to [Taylor et al. 2016], we selected a subset of shape parameters in Figure 10: $M = \{0{:}16, 19, 22, 25, 28, 45{:}74\}$. This is necessary as sphere-centres on the palm can move without affecting the tracking energy – a null-space of our optimization. The tracking algorithm is initialized in the first frame by $\bar{\theta}_0$. In Figure 6, we report an experiment analogous to that of Figure 3 but on a full 3D sequence. Consider Figure 6b, where we report the runtime estimates for the length of the middle-finger's middle-phalanx; the subscript [7] will henceforth be implied. Congruously to the discussion in Section 3, the phalanx length estimates computed in frames where the finger is bent are given a large weight. Per-frame estimates β_n^* can often oscillate away from the ground truth, but these incorrect estimates are associated with a small weight. Our algorithm estimates these per-frame uncertainties Σ_n^*, and accordingly updates the online cumulative estimates $\hat{\beta}_n$ and $\hat{\Sigma}_n$.

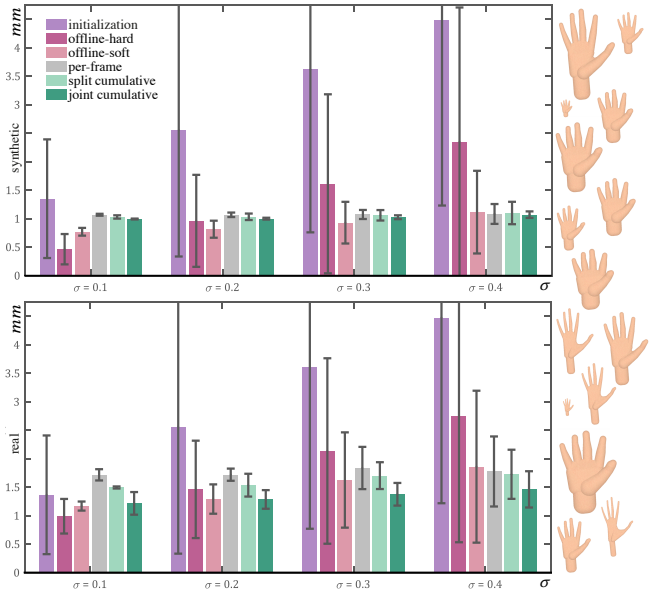

Fig. 7. Mean and standard deviation for ground-truth calibration residuals as we vary the algorithm's initialization with a random perturbation of standard deviation σ. We evaluate the residuals on (top) synthetic depth maps, as well as (bottom) on the raw depth maps. (right) the exemplars drawn from the $\sigma = 0.4$ perturbation used in the evaluation.

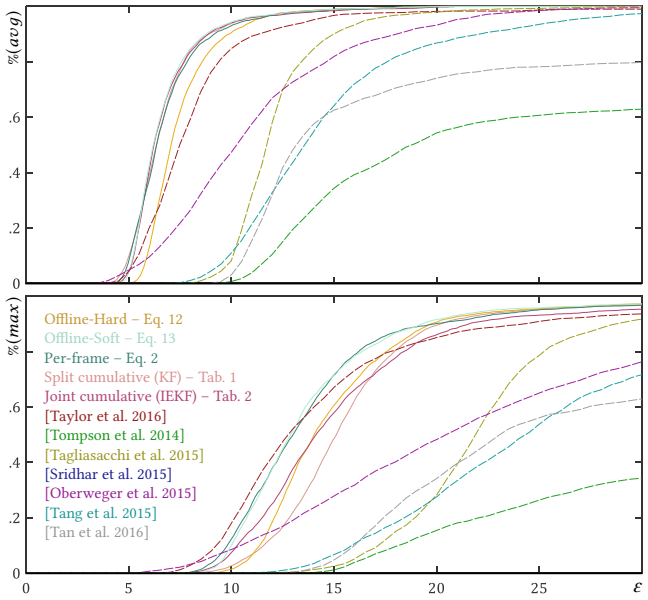

Fig. 8. Evaluation on the NYU dataset from [Tompson et al. 2014] reporting the percentage of frames with average (top) and max (bottom) ground-truth marker-error less than ε.

4.3 Synthetic dataset: robustness – Fig. 7

We evaluate the *robustness* of the algorithm by analyzing its convergence properties as we vary the magnitude of perturbations. We provide two experiments: *synthetic* and *real*. The real dataset consists of depth maps $\{\mathcal{D}_n\}$ measured by an Intel Realsense SR300 sensor, where we track motion with the multi-view stereo (MVS) calibrated model from [Tkach et al. 2016] to estimate a sequence of pose parameters $\{\theta_n\}$; the shape parameters $\bar{\beta}$ of this user are known with good confidence thanks to the MVS data. In the synthetic dataset, depth images $\{\mathcal{D}_n\}$ are generated by animating the sphere-mesh model $\mathcal{M}(\theta_n, \bar{\beta})$ and then rasterizing the model as in Section 4.2. To achieve this, random initializations for the user-personalized models are drawn from the Gaussian distribution $\mathcal{N}(\bar{\beta}, \sigma)$. A few examples of such perturbations for $\sigma = .4$ are visualized in Figure 7. In our experiments, we draw fifteen random samples per each value of σ, and compute mean and standard deviation of the measured ground truth residuals E_β. As each sample requires the re-tracking of the entire sequence ($\approx$ 20 seconds) with a new initialization, the two plots in Figure 7 amount to roughly four hours of footage. For this reason, in Video [03:16] we only display a few examples of calibration and apply a random perturbation every few seconds. Notice that although we still have a non-zero average residual of $\approx$ 1mm, the video shows how the model is an excellent fit to the synthetic data. In both experiments, Offline-Hard performs worse than Offline-Soft for the reasons discussed in Section 3.4. With the large ($\sigma = .4$) perturbations Offline-Soft still had some troubles converging to the correct results, as per-frame pose initializations were too significantly wrong; in this regard, we believe discriminative pose re-initializers such as [Oberweger et al. 2015] could be helpful to increase the performance of both offline calibration algorithms.

Technically we could not display the per-frame algorithm performance in Figure 7, since it does not provide a single final estimate of shape parameters. To do this, we employ the model parameters it estimated in the last frame of each sequence. In the last frame the error is low, as each frame is initialized with the values from the previous frame; see Video [03:41]. Note how per-frame calibration performs excellently, even outperforming Offline-Hard. This is because, thanks to our carefully designed shape priors, per-frame calibration is quite robust; see Video [03:41]. This is essential in cumulative calibration, as the true value of a parameter can be recovered only if accurate measurements are available in at least some poses. The per-frame algorithm should also not be mistaken for tracking algorithm (where shape parameters are fixed) which is twice more efficient (calibration executes at 30 Hz, while tracking executes at 60 Hz) and, in general, much more robust.

It is difficult to differentiate the split vs. joint cumulative variants in the synthetic dataset, as calibration converges very effectively when it can rely on precise measurements. Overall, on the sensed dataset our *joint cumulative* calibration performs the best. Our split variant performs very well when per-frame consistently provides an accurate solution (e.g. on the synthetic sequences). Nonetheless, we noticed that how with more challenging motions, the joint-cumulative can aid the per-frame solver by providing a temporal regularization. This is beneficial when dealing with an uncooperative user, or to perform calibrations in sequences that were not specifically designed for this task (e.g. fast motion and long-term occlusions).

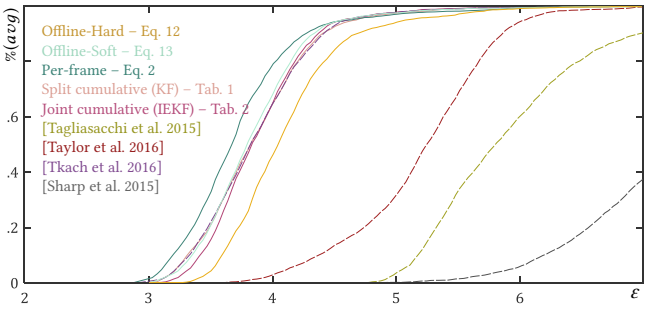

Fig. 9. Evaluation on the Handy/Teaser dataset, reporting the percentage of frames with an average E_{d2m} data-to-model energy below ε.

4.4 Marker-based evaluation on NYU dataset – Fig. 8

Although several marker-based datasets are available, such as [Qian et al. 2014], [Sharp et al. 2015] and [Yuan et al. 2017], state-of-the-art *generative* methods have focused on the NYU [Tompson et al. 2014] and Handy [Tkach et al. 2016] datasets for quantitative evaluation. On the NYU dataset, to properly compare to Taylor et al. [2016], we evaluate the metrics on the first 2440 frames (user #1), and consider only markers on finger joints. This dataset allow us to compare our method (and its variants) to a number of other algorithms including: the PSO tracker by Sharp et al. [2015], the calibration methods by Khamis et al. [2015] and Tan et al. [2016], the subdivision tracker of Taylor et al. [2016], the cylindroid tracker by Tagliasacchi et al. [2015], the sphere-mesh tracker by Tkach et al. [2016], the Gaussian tracker of Sridhar et al. [2015], and discriminative methods such as those of Tompson et al. [2014], Tang et al. [2015] and Oberweger et al. [2015]. Our online algorithm achieves very competitive tracking performance while being the *first* capable of calibrating the user-personalized tracking model *online*, rather than in an offline calibration session like Taylor et al. [2016]. Notice how the best performance is achieved by either: (1) the per-frame optimization, where per-frame *overfitting* takes place, or (2) by offline calibration techniques such as Offline-Soft or [Taylor et al. 2016]. This is expected, as offline algorithms jointly consider all available information, while online/streaming algorithms can only integrate information one frame at a time.

4.5 Dense evaluation on the Handy dataset – Fig. 9

Another way to evaluate the quality of tracking/calibration is to compare the depth map $\mathcal{D}_n$ (i.e. sensor point cloud) to the tracking model depth map $\bar{\mathcal{D}}(\theta, \beta)$ (i.e. model point cloud); see [Tkach et al. 2016]. The model depth map is obtained by rendering $\mathcal{M}(\theta, \beta)$ with the same intrinsic/extrinsic parameters of the sensor. The following metric measures the average magnitude of data-to-model ICP correspondences:

$$E^n_{\text{d2m}} = \frac{1}{|d_n|} \sum_{\mathbf{p}_j \in d_n} \|\mathbf{p}_j - \Pi_{\bar{\mathcal{D}}(\theta,\beta)}(\mathbf{p}_j)\|_2 \qquad (17)$$

where Π is an operator computing the closest-point projection of points in the sensor's point cloud, onto the point-cloud associated with the synthetic depth-map $\mathcal{D}(\theta, \beta)$. This metric is *dense*, as it computes residual of an entire geometry model rather than just a sparse set of markers. If $E_{\text{d2m}} \approx 0$ (up to sensor noise) in every frame, then the personalized model is a seemingly perfect *dynamic replica* of the user's hand. The Handy dataset from [Tkach et al. 2016] enables these type of comparisons and includes rendered depth maps for [Tagliasacchi et al. 2015], [Sharp et al. 2015], as well as the state-of-the-art method of [Taylor et al. 2016]. Further, note how this dataset considers a range of motion substantially more complex than the one in the NYU dataset. Like in earlier comparisons, the per-frame technique performs best as it overfits to the data, by generating a collection of β_n instead of a single tuple β. Our techniques calibrate a model with performance comparable to that of [Tkach et al. 2016], where a high-quality MVS point cloud with manual annotations was used for calibration.

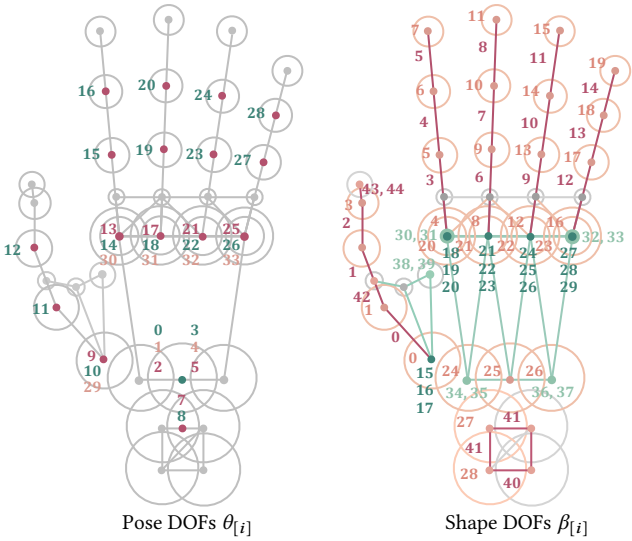

Fig. 10. The degrees of freedom of our optimization, where use a cartesian right-handed coordinate frame for translational DOFs. For pose parameters, global translation is represented by $\theta_i||i \in [0, 1, 2]$ and rotation by $\theta_i||i \in [3, 4, 5]$. We then color code DOFs according to whether they represent flexion, twist, and abduction. For shape parameters, we color code DOFs for lengths, radii, 3DOFs vertices (x,y,z), 2DOFs vertices (x,y), and passive DOFs (linearly dependent).

5 CONCLUSIONS

From an application point of view, our approach significantly improves on the usability of real-time hand tracking, as it requires neither controlled calibration scans nor offline processing prior to tracking. This allows easy deployment in consumer-level applications. From a technical point of view, we introduce a principled approach to online integration of shape information of user-specific hand geometry. By leveraging uncertainty estimates derived from the optimization objective function, we automatically determine how informative each input frame is for improving the estimates of the different unknown model parameters. Our approach is general and can be applied to different types of calibration, e.g., for full body tracking. More broadly, we envisage applications to other difficult types of model estimation problems, where unreliable data needs to be accumulated and integrated into a consistent representation.

Limitations and future works. The intrinsic limitation of our online approach as well its offline counterparts is reliance on reasonable tracking quality during calibration. If tracking fails, the model quality is compromised as shown in the Video [07:18]. Currently, our optimization relies on heavy parallelization and high-end GPU hardware – we use a 4GHz i7 equipped with an NVIDIA GTX 1080Ti. In future work we want to reduce computational overhead to facilitate deployment on mobile devices. To obtain a complete personalized tracking model, the user needs to perform a suitable series of hand poses. As discussed above, if a finger is never bent, the estimate of phalanx lengths will be unreliable. Currently, the system provides limited visual feedback to the user to guide the calibration. In the future, we aim to design a feedback system that provides visual indication of the most informative hand poses given the current model estimate. For example one could create a dictionary containing a suitable pose for estimating each parameter with high certainty. During calibration the user is prompted to show hand pose corresponding to the lowest certainty parameter. Other interesting avenues for future work include extending our approach to handle hand-object or hand-hand interactions, adapting the method to other tracking scenarios such as full body tracking, and studying the perceptual relevance of tracking accuracy to further optimize the performance of our approach.

ACKNOWLEDGMENTS

We would like to thank Tom Cashman for his help executing quantitative comparisons, as well as Marco Tarini, Baptiste Angles, and Daniel Rebain for their help proofreading the paper. This work is supported by the SNF grant #200021-153567, the National Science and Engineering Research Council of Canada (NSERC) Discovery grant #2016-05786, and the Industrial Research Chair in 3D Sensing.

REFERENCES

Irene Albrecht, Jörg Haber, and Hans-Peter Seidel. 2003. Construction and animation of anatomically based human hand models. In *Proc. Symp. on Computer Animation (SCA)*.

Brian Anderson and John Moore. 1979. *Optimal filtering*. Englewood Cliffs.

Bradley M Bell and Frederick W Cathey. 1993. The iterated Kalman filter update as a Gauss-Newton method. In *IEEE Trans. on Automatic Control*.

R Louis Bellaire, Edward W Kamen, and Serena M Zabin. 1995. New nonlinear iterated filter with applications to target tracking. In *SPIE Intl. Symposium on Optical Science, Engineering, and Instrumentation*.

Federica Bogo, Michael J Black, Matthew Loper, and Javier Romero. 2015. Detailed full-body reconstructions of moving people from monocular RGB-D sequences. In *Proc. Intl. Conf. on Comp. Vision (ICCV)*.

Sofien Bouaziz, Yangang Wang, and Mark Pauly. 2013. Online modeling for realtime facial animation. In *ACM Trans. on Graphics (Proc. SIGGRAPH)*.

Chen Cao, Derek Bradley, Kun Zhou, and Thabo Beeler. 2015. Real-time high-fidelity facial performance capture. In *ACM Trans. on Graphics (Proc. SIGGRAPH)*.

Chen Cao, Hongzhi Wu, Yanlin Weng, Tianjia Shao, and Kun Zhou. 2016. Real-time facial animation with image-based dynamic avatars. In *ACM Trans. on Graphics (Proc. SIGGRAPH)*.

Martin de La Gorce, David J Fleet, and Nikos Paragios. 2011. Model-based 3D hand pose estimation from monocular video. In *Pattern Analysis and Machine Intelligence (PAMI)*.

Paul Ekman and Wallace V Friesen. 1977. *Facial Action Coding System*. Consulting Psychologists Press, Stanford University, Palo Alto.

Jinwei Gu, Xiaodong Yang, Shalini De Mello, and Jan Kautz. 2017. Dynamic Facial Analysis: From Bayesian Filtering to Recurrent Neural Network. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Jindřich Havlík and Ondřej Straka. 2015. Performance evaluation of iterated extended Kalman filter with variable step-length. In *Journal of Physics: Conf. Series*.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and others. 2011. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. ACM User Interface Software and Technology*.

Sameh Khamis, Jonathan Taylor, Jamie Shotton, Cem Keskin, Shahram Izadi, and Andrew Fitzgibbon. 2015. Learning an efficient model of hand shape variation from depth images. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Hao Li, Jihun Yu, Yuting Ye, and Chris Bregler. 2013. Realtime Facial Animation with On-the-fly Correctives. In *ACM Trans. on Graphics (Proc. SIGGRAPH)*.

Alexandros Makris and A Argyros. 2015. Model-based 3D hand tracking with online hand shape adaptation. In *Proc. British Machine Vision Conf. (BMVC)*.

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer.

Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. 2015. Hands deep in deep learning for hand pose estimation. In *Proc. Computer Vision Winter Workshop*.

Kaare Brandt Petersen, Michael Syskind Pedersen, and others. 2008. *The matrix cookbook*. Technical University of Denmark.

Gerard Pons-Moll and Bodo Rosenhahn. 2011. Model-based pose estimation. In *Visual analysis of humans: Looking at People*. Springer.

Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. 2014. Realtime and robust hand tracking from depth. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Edoardo Remelli, Anastasia Tkach, Andrea Tagliassachi, and Mark Pauly. 2017. Low-Dimensionality Calibration through Local Anisotropic for Scaling for Robust Hand Model Personalization. In *Proc. Intl. Conf. on Comp. Vision (ICCV)*.

Taehyun Rhee, Ulrich Neumann, and John P Lewis. 2006. Human hand modeling from surface anatomy. In *Proc. Symposium on Interactive 3D graphics and games*.

Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, and others. 2015. Accurate, robust, and flexible real-time hand tracking. In *Proc. ACM Special Interest Group on Computer-Human Interaction (CHI)*.

Martin A Skoglund, Gustaf Hendeby, and Daniel Axehill. 2015. Extended Kalman filter modifications based on an optimization view point. In *Intl. Conf. on Inf. Fusion*.

Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. 2015. Fast and Robust Hand Tracking Using Detection-Guided Optimization. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Hauke Strasdat, José MM Montiel, and Andrew J Davison. 2012. Visual SLAM: why filter?. In *Image and Vision Computing*.

James S Supancic, Grégory Rogez, Yi Yang, Jamie Shotton, and Deva Ramanan. 2015. Depth-based hand pose estimation: data, methods, and challenges. In *Proc. Intl. Conf. on Comp. Vision (ICCV)*.

Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. 2015. Robust Articulated-ICP for Real-Time Hand Tracking. In *Computer Graphics Forum (Proc. Symposium on Geometry Processing)*.

David J. Tan, Thomas Cashman, Jonathan Taylor, Andrew Fitzgibbon, Daniel Tarlow, Sameh Khamis, Shahram Izadi, and Jamie Shotton. 2016. Fits like a glove: Rapid and reliable hand shape personalization. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Danhang Tang, Jonathan Taylor, Pushmeet Kohli, Cem Keskin, Tae-Kyun Kim, and Jamie Shotton. 2015. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *Proc. Intl. Conf. on Comp. Vision (ICCV)*.

Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, and others. 2016. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. In *ACM Trans. on Graphics (Proc. SIGGRAPH)*.

Jonathan Taylor, Richard Stebbing, Varun Ramakrishna, Cem Keskin, Jamie Shotton, Shahram Izadi, Aaron Hertzmann, and Andrew Fitzgibbon. 2014. User-specific hand modeling from monocular depth sequences. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. 2015. Real-time expression transfer for facial reenactment. *ACM Trans. Graph.* 34, 6 (2015), 183–1.

Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. 2016. Sphere-meshes for real-time hand modeling and tracking. In *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*.

Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. 2014. Real-time continuous pose recovery of human hands using convolutional networks. In *ACM Trans. on Graphics (TOG)*.

Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. 2016. Learning to navigate the energy landscape. In *Int. Conf. on 3D Vision (3DV)*.

Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. 2011. Realtime performance-based facial animation. In *ACM Trans. on Graphics (Proc. SIGGRAPH)*.

Greg Welch and Gary Bishop. 1995. *An introduction to the Kalman filter*.

Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhand Jain, and Tae-Kyun Kim. 2017. BigHand2M Benchmark: Hand Pose Dataset and State of the Art Analysis. In *arXiv preprint arXiv:1704.02612*.

Danping Zou and Ping Tan. 2013. COSLAM: Collaborative visual slam in dynamic environments. In *Pattern Analysis and Machine Intelligence (PAMI)*.

Time	Measurement
$\hat{x}_n^0 = \hat{x}_{n-1}$	$K_n = P_n^0 J^T (J P_n^0 J^T + R)^{-1}$
$P_n^0 = P_{n-1} + Q$	$\hat{x}_n = \hat{x}_n^0 + K_n(z_n - J\hat{x}_n^0)$
	$P_n = (I - K_n J) P_n^0$

Table 3. Kalman Filter update equations (with $A = I$).

Time	Measurement
$\hat{x}_n^0 = \hat{x}_{n-1}$	$K_n = P_n^0 J_n^T (J_n P_n^0 J_n^T + R)^{-1}$
$P_n^0 = P_{n-1} + Q$	$\hat{x}_n = \hat{x}_n^0 + K_n(z_n - F_n)$
	$P_n = (I - K_n J_n) P_n^0$

Table 4. Extended Kalman Filter update equations (with linear $\tilde{F}$).

A OVERVIEW ON KALMAN FILTERS

Kalman Filter (KF). Following the notation in [Welch and Bishop 1995], let us denote the latent state of a discrete-time controlled process as $x_n \in \mathbb{R}^N$, a generic measurement as $z_n \in \mathbb{R}^M$ and let us consider the following linear stochastic difference equations

$$x_n = Ax_{n-1} + w_{n-1} \tag{18}$$
$$z_n = Jx_n + v_n \tag{19}$$

where w is a normally distributed process noise $p(w) \sim \mathcal{N}(0, Q)$, and v is a normally distributed measurement noise $p(v) \sim \mathcal{N}(0, R)$. The matrix A provides a linear estimate for state updates, while J maps the state x_n to the measurement z_n. Given a generic frame n, let us define an initial (*a priori*) state estimate $\hat{x}_n^0$, together with an improved (*a posteriori*) state estimate $\hat{x}_n$ accounting for the measurement z_n. We can then define *a priori* and *a posteriori* estimate error covariances as

$$P_n^0 = \mathbb{E}[(x_n - \hat{x}_n^0)^T (x_n - \hat{x}_n^0)] \tag{20}$$
$$P_n = \mathbb{E}[(x_n - \hat{x}_n)^T (x_n - \hat{x}_n)]. \tag{21}$$

The Kalman Filter (KF) estimates the latent state x_n of a discrete control linear process by minimizing the *a posteriori* error covariance. In particular it estimates the process through a *predictor-corrector* approach: given a generic time n the filter first estimates the process state (*time update equation*) and then obtains feedback in the form of noisy measurements (*measurement update equation*). Let us now particularize the system above to our framework, where the latent state of our system corresponds to hand parameters and the measurement corresponds to the solution of Equation 2. An estimate of the current hand parameters is given by the one of the previous time-step up to Gaussian noise, that is $x_n = x_{n-1} + w_{n-1}$, while the noisy measurement corresponds to the state itself, meaning that $J = I$ (note that in order to highlight the similarities to other Kalman filter formulations we will maintain the notation J). Our discrete-time process can simply be written in the following form, resulting in the update equations of Table 3; see [Welch and Bishop 1995]:

$$x_n = x_{n-1} + w_{n-1} \tag{22}$$
$$z_n = Jx_n + v_n \tag{23}$$

Extended Kalman Filter (EKF). The Extended Kalman Filter (EKF) extends the KF to the case in which the process to be estimated and/or the measurement relationship to the process are not linear:

$$x_n = \tilde{F}(x_{n-1}, w_{n-1}) \tag{24}$$
$$z_n = F(x_n, v_n) \tag{25}$$

where $\tilde{F}$ relates the current latent state x_n to the previous time step one x_{n-1} and F relates the current latent state x_n to measurement z_n. The EKF simply estimates the latent state of such system by means of linearization of process and measurement equations around the current estimate; see [Welch and Bishop 1995] for a detailed overview. We can apply this framework to ours and, differently from the linear case, consider now the input depth map d_n as system measurement. The function $F(\cdot)$ therefore maps state x_n to measurement z_n by *applying* shape and pose parameters to the template hand model and computing the closest model points to sensor data points, while as discussed in the previous section $\tilde{F}(\cdot)$ is a simple identity mapping. We can write the non-linear process and measurement equations associated to our framework as:

$$x_n = x_{n-1} + w_{n-1} \tag{26}$$
$$z_n = F(x_n) + v_n \tag{27}$$

By defining $F_n = F(\hat{x}_n^0)$ and $J_{n[i,j]} = \partial F_{[i]} / \partial x_{[j]}(\hat{x}_n^0)$, the EKF update equations can be written as reported in Table 4; see [Welch and Bishop 1995].

Iterated Extended Kalman Filter (IEKF). The EKF performs well for systems with mildly nonlinear measurement functions, but if the measurement equation is strongly nonlinear the performance of the filter deteriorates; see [Havlík and Straka 2015]. To address this problem, we can perform measurement updates in several steps, where in each one we linearize the measurement function F around the updated value iteration $\hat{x}_n^i$, leading to the Iterated Extended Kalman Filter (IEKF) formulation [Havlík and Straka 2015]. The

for $i = 1 \ldots i_{\max}$
$F_n^i = F(\hat{x}_n^i) \Rightarrow J_{n[u,v]}^i = \partial F_{[u]}^i(\hat{x}_n^i) / \partial x_{[v]}$
$K_n^i = P_n^0 J_n^{iT} (J_n^i P_n^0 J_n^{iT} + R)^{-1}$
$\hat{x}_n^{i+1} = \hat{x}_n^0 + K_n^i (z_n - F_n^i - J_n^i (\hat{x}_n^0 - \hat{x}_n^i))$
end
$\hat{x}_n = \hat{x}_n^i$
$P_n = (I - K_n J_n^i) P_n^0$

Table 5. Iterated EKF measurement update equations.

Ex. Kalman Filter	Ex. Information Filter
	$H_n = (P_n)^{-1}$
$K_n = P_n^0 J_n^T (J_n P_n^0 J_n^T + R)^{-1}$	$H_n = H_n^0 + J_n^T R^{-1} J_n$
$\hat{x}_n = \hat{x}_n^0 + K_n(z_n - F_n)$	$K_n = {H_n}^{-1} J_n^T R^{-1}$
$P_n = (I - K_n J_n) P_n^0$	$\hat{x}_n = \hat{x}_n^0 + K_n(z_n - F_n)$

Table 6. Analogy of EKF and EIF.

time update equation for IEKF is analogous to the one in Table 4, while the measurement update is reported in Table 5.

Extended Information Filters (EIF). In order to ease the derivations of the upcoming section let us observe that the EKF measurement updates can also be rewritten in the equivalent *Extended Information Filter* form [Anderson and Moore 1979]; see Table 6. We introduce this formulation in order to ease the upcoming derivations. Note that in order to do that we need to assume the measurement noise to be independent and identically distributed (i.i.d.) across samples, therefore $R = rI$ where $r \in \mathbb{R}^+$ and I is the identity matrix. Further, similarly to EKF, we can write the iterated version of an EIF, as reported in Table 7.

A.1 Laplace Approximation

To derive our uncertainties, we start by converting the data terms *d2m* and *m2d* of Equation 2 into probabilistic form:

$$p(d_n|x_n) = \exp\left(-\tfrac{1}{2}(d_n - F(x_n))^T (d_n - F(x_n))\right) \tag{28}$$

By temporarily omitting the frame index n for conveniency, our problem is rewritten as a maximum likelihood optimization:

$$x^* = \arg\max_x \log p(d|x) = \arg\max_x L(x) \tag{29}$$

We now perform a second-order Taylor expansion of the log-likelihood of the data $L(x)$ around the *optimal* solution x^*:

$$L(x) \approx \tilde{L}(x) = L(x^*) + \frac{\partial L(x^*)}{\partial x}\Delta x + \tfrac{1}{2}\Delta x^T \frac{\partial^2 L(x^*)}{\partial x^2}\Delta x + \text{h.o.t.} \tag{30}$$

where $\Delta x = x - x^*$, and let $\partial f(x^*)/\partial x$ indicate the partial derivative of $f(x)$ evaluated at x^*. We rewrite $\bar{F}(x_n) = d_n - F(x_n)$ for brevity. Note how the Jacobian and the Hessian are respectively zero and positive definite at our optimal point x^* (see [Nocedal and Wright 2006, Sec. 10.2]):

$$\frac{\partial L(x^*)}{\partial x} = -\bar{F}(x^*)^T \frac{\partial \bar{F}(x^*)}{\partial x} = 0 \tag{31}$$

$$\frac{\partial^2 L(x^*)}{\partial x^2} \approx -\frac{\partial \bar{F}(x^*)}{\partial x}^T \frac{\partial \bar{F}(x^*)}{\partial x} \triangleq -\Sigma^{*-1} \prec 0 \tag{32}$$

From Equation 30, using $\tilde{p}(d|x) = \exp(\tilde{L}(x))$, we can then derive the *approximated* posterior distribution:

$$\tilde{p}(d|x) = \exp\left(-\tfrac{1}{2}(x - x^*)^T \Sigma^{*-1}(x - x^*)\right) = \mathcal{N}(x^*, \Sigma^*) \tag{33}$$

B DERIVATIONS

B.1 Derivation for Section 3.2

Let us consider the Kalman Filter measurement update equations introduced in Table 4, recalling to the reader that we are considering the case in which the measurement $z_n = x_n^*$ is in the same space of the estimated state $\hat{x}_n$, thus when J is the identity matrix.

$$\begin{aligned}
\hat{x}_n &= \hat{x}_n^0 + P_n^0(P_n^0 + R)^{-1}(z_n - \hat{x}_n^0) \\
&= (P_n^0 + R)(P_n^0 + R)^{-1}\hat{x}_n^0 + P_n^0(P_n^0 + R)^{-1}(z_n - \hat{x}_n^0) \\
&= R(P_n^0 + R)^{-1}\hat{x}_n^0 + P_n^0(P_n^0 + R)^{-1} z_n \\
P_n &= ((P_n^0 + R)(P_n^0 + R)^{-1} - P_n^0(P_n^0 + R)^{-1})P_n^0 = R(P_n^0 + R)^{-1}P_n^0
\end{aligned}$$

Note how setting $z_n = x_n^*$, $P_n^0 = \hat{\Sigma}_{n-1}$ and $R = \Sigma_n^*$ the measurement update equations coincide with Equation 10 for product of two Gaussians, showing how the inter-frame regression algorithm is indeed equivalent to a KF.

for $i = 1 \ldots i_{\max}$
$H_n^i = \frac{1}{r}(rH_n^0 + J_n^{iT} J_n^i)$
$K_n^i = \frac{r}{r} {H_n^i}^{-1} J_n^{iT}$
$\hat{x}_n^{i+1} = \hat{x}_n^0 + K_n^i(z_n - F_n^i - J_n^i(\hat{x}_n^0 - \hat{x}_n^i))$
end
$\hat{x}_n = \hat{x}_n^i$

Table 7. Iterated EIF update equations.

B.2 Derivation for Section 3.3

Focusing on the optimization associated to Table 2, let us consider a generic Gauss-Newton iterative update which reads:

$$x_n^{i+1} = x_n^i - (\bar{J}_n^T \bar{J}_n)^{-1} \bar{J}_n^T \bar{F}_n \tag{34}$$

observing that

$$\bar{J}_n = \begin{bmatrix} -J_n^i \\ \hat{\Sigma}_{n-1}^{-1/2} \end{bmatrix} \quad \bar{F}_n = \begin{bmatrix} d_n - F_n^i \\ \hat{\Sigma}_{n-1}^{-1/2}(x_n^i - \hat{x}_{n-1}) \end{bmatrix} \tag{35}$$

we obtain what follows:

$$\begin{aligned}
\bar{J}_n^T \bar{J}_n &= {J_n^i}^T J_n^i + \hat{\Sigma}_{n-1}^{-1} = A^{-1} \\
\bar{J}_n^T \bar{F}_n &= -{J_n^i}^T (z_n - F_n^i) + \hat{\Sigma}_{n-1}^{-1}(x_n^i - \hat{x}_{n-1})
\end{aligned}$$

where $F_n^i = F(x_n^i)$ and $J_n^i = \frac{\partial F}{\partial x_n}(x_n^i)$. Hence, expanding the matrix products in (34), we can write:

$$\begin{aligned}
x_n^{i+1} &= x_n^i + \overbrace{A{J_n^i}^T(d_n - F_n^i)}^{B} - A\hat{\Sigma}_{n-1}^{-1}(x_n^i - \hat{x}_{n-1}) = \\
&= \hat{x}_{n-1} + B - A\hat{\Sigma}_{n-1}^{-1}(x_n^i - \hat{x}_{n-1}) + AA^{-1}(x_n^i - \hat{x}_{n-1}) = \\
&= \hat{x}_{n-1} + B + A(-\hat{\Sigma}_{n-1}^{-1} + {J_n^i}^T J_n^i + \hat{\Sigma}_{n-1}^{-1})(x_n^i - \hat{x}_{n-1}) = \\
&= \hat{x}_{n-1} + B + A{J_n^i}^T J_n^i (x_n^i - \hat{x}_{n-1}) = \\
&= \hat{x}_{n-1} + \underbrace{\left({J_n^i}^T J_n^i + \hat{\Sigma}_{n-1}^{-1}\right)^{-1} {J_n^i}^T}_{K_n^i} \left(d_n - F_n^i - J_n^i(\hat{x}_{n-1} - x_n^i)\right).
\end{aligned}$$

Recalling the definition of the *a priori* estimate $x_n^0 = \hat{x}_{n-1}$, setting $H_n^0 = H_{n-1}$ and denoting $\hat{\Sigma}_{n-1}^{-1} = rH_{n-1}$ we can now see that such an iterative update is equivalent to the update of the IEIF from Table 7 for measurement $z_n = d_n$. Finally, under the assumption of the measurement noise to be i.i.d. across samples, we can conclude that the optimization from 7 is indeed equivalent to an IEKF.