Vision, Modeling, and Visualization (2014)
J. Bender, A. Kuijper, T. von Landesberger, H. Theisel and P. Urban (Eds.)

Online Adaptive PCA for Inverse Kinematics Hand Tracking

Matthias Schröder Mario Botsch

Computer Graphics & Geometry Processing Group, Bielefeld University

Abstract

Recent approaches to real-time bare hand tracking estimate the hand's pose and posture by fitting a virtual hand model to RGBD sensor data using inverse kinematics. It has been shown that exploiting natural hand synergies can improve the efficiency and quality of the tracking, by performing the optimization in a reduced parameter space consisting of realistic hand postures [SMRB14]. The downside, however, is that only postures within this subspace can be tracked reliably, thereby trading off flexibility and accuracy for performance and robustness.
In this paper we extend the previous method by introducing an adaptive *synergistic model that is automatically adjusted to observed hand articulations that are not covered by the initial subspace. Our adaptive model combines the robustness of tracking in a reduced parameter space with the flexibility of optimizing for the full articulation of the hand, which we demonstrate in several synthetic and real-world experiments.*

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking

1. Introduction

Visual tracking of human hand movements is a problem that has applications in many different areas, such as human-computer interaction, character animation and robotics. While there are several existing hand tracking solutions, many of these are expensive, inconvenient, or lacking in efficiency. Other methods provide means for rough gestural interaction based on consumer-level depth cameras, but the recovery of the user's hand articulation with full degrees of freedom (DoFs) remains an ongoing research topic.

In a recent paper, Schröder et al. [SMRB14] presented a method for real-time bare hand tracking using an RGBD sensor, where the hand posture estimation was formulated as an inverse kinematics (IK) problem based on iterative closest point correspondences between the sensor point cloud and a virtual hand model. They perform the IK optimization in a reduced parameter space, which was obtained from a motion capture database containing human hand movements using principal component analysis (PCA). Solving the IK problem in this subspace reduces computational complexity and constrains the tracking to realistic hand postures even in cases of incomplete or ambiguous sensor data.

However, this subspace tracking is a trade-off between robustness and flexibility of the hand posture estimation. Hand articulations that are not contained in the database cannot be reconstructed during tracking. While a large amount of natural hand postures can be represented in a synergistic subspace, some hand articulation details are not captured.

In this paper we present an extension to the above method that adds flexibility to the posture estimation while maintaining the robustness of tracking in a synergistic subspace. To this end we define an adaptive PCA model that is adjusted during real-time tracking to account for observed hand articulations that are not covered by the initial parameter subspace. We closely follow and extend the approach outlined in [LYYB13], where such an adaptive model was used for real-time facial performance capture. By extending the hand tracking method described in [SMRB14] by an adaptive PCA model we robustly combine the natural constraints provided by subspace optimization with the flexibility of optimizing in the full parameter space.

2. Related work

Approaches to the hand posture estimation problem are commonly classified as appearance-based [WP09, RKK10,

WPP11] or model-based methods. In this paper we focus on the latter only and refer the reader to [SMRB14] for more details.

The model-based approach of Oikonomidis et al. [OKA11, OKA12] produces solid results using particle swarm optimization (PSO) and a Kinect camera, but had to be optimized to run on a GPU due to its high computational complexity. The PSO hand tracking method has since been extended and combined with other tracking approaches, such as marker-based mocap [ZCX12] and gradient-based optimization [QSW*14]. Other model-based approaches propose sophisticated optimization frameworks to accurately reconstruct the articulation of hands [dLGFP11] or hands interacting with objects [BTG*12, WMZ*13], but these methods are thus far not suitable for real-time tracking.

Schröder et al. [SMRB14] exploit hand synergies by performing the inverse kinematics in a PCA subspace of motion captured hand movements. Similar concepts were previously employed for synthesis of realistic human motion [GMHP04, CH05, ZZMC13, LWH00]. Using synergistic concepts to reduce the DoFs of a virtual hand model, the high computational complexity associated with model-based tracking is reduced, while at the same time facilitating robust real-time bare hand tracking. However, hand postures not contained in the database or not covered by the global PCA subspace cannot be tracked reliably.

In this paper, we define an *adaptive* PCA model that can adjust the synergistic subspace to previously unknown hand articulations in real-time. This concept was used in a similar fashion in [LYYB13] for real-time facial performance capture in order to accurately adapt a blend-shape face model to user-specific facial expressions. Using an adaptive PCA model we optimize the hand articulation in a low-dimensional space that both constrains the estimation to realistic postures while still allowing for high flexibility and accuracy. We develop an incremental update of the adaptive PCA subspace based on direct rank-one updates, which allows for a highly efficient adaptation process.

Adaptive PCA extends the method of [SMRB14] in a simple and efficient way by complementing the initial subspace model with a continuously updated local linear model. While there are various other methods for linear or non-linear local embeddings [UD08, RSH02], usually their performance depends on the quality of model parameters or they do not meet real-time requirements. Data-driven local linear models were previously used for full-body motion capture in [CH05, LZWM06]. Our method differs from these approaches in that we aim to specifically model articulations that are not present in the ground truth database.

In the following we briefly review the IK hand tracking of [SMRB14], first using all DoFs (Section 3) and then the reduced parameter subspace (Section 4), before presenting our adaptive synergistic model in Section 5.

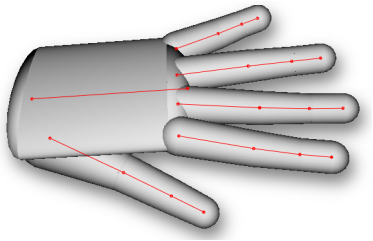

Figure 1: *Virtual hand model with its control skeleton.*

3. Inverse kinematics hand tracking

Our kinematic hand model consists of 16 joints, three for each finger and one for the wrist (Figure 1). Its posture is controlled by 20 joint angles, where each finger joint has a flexion-extension angle and the fingers' base joints each have an additional abduction-adduction angle. In addition to the 20 joint angles controlling the posture, the pose of the hand is represented by 6 DoFs for the global translation and rotation. In total we use a 26-dimensional kinematic parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{26})^T$ to control the pose and posture of the hand. These parameters and the kinematic chain of the joint hierarchy define the forward kinematics of the hand model, which can be expressed in terms of a product of affine transformations for each joint.

The geometry of the virtual hand model is composed of capsule-shaped segments, which are rigidly transformed according to the articulation of their corresponding joints, as shown in Figure 1. A major advantage of simple capsule-shaped segments is the efficient computation of point-to-segment correspondences, which has to be performed many times during the IK-based hand tracking.

The general tracking process is depicted in Figure 2. The pose and posture of the user's hand are estimated by fitting the virtual hand model to the point cloud obtained from an RGBD Kinect camera. In a preprocessing step, the hand is segmented in the input point cloud by detecting skin-colored pixels, omitting points whose coordinates are outside of a predefined working volume, and uniformly sub-sampling the detected hand pixels. The remaining points define the target constraints $\{\mathbf{t}_1, \ldots, \mathbf{t}_k\}$ for the fitting process.

These target points are matched to their spatially closest points $\mathbf{p}_i$ on the surface of the hand model. Estimating the hand articulation is then an inverse kinematics problem, in which the k points $\mathbf{p}_i = \mathbf{p}_i(\boldsymbol{\theta})$ are regarded as *effector positions* (relative to the joint hierarchy and kinematic parameters), which are subject to move towards their corresponding *target positions* $\mathbf{t}_i$ in the sensor point cloud.

To this end the pose is initialized by finding the best rigid transformation between the target and effector points using the well known rigid iterative closest points (ICP) technique [BM92], based on the posture parameters from the previous frame. Then the correspondences $(\mathbf{t}_i, \mathbf{p}_i)$ are up-

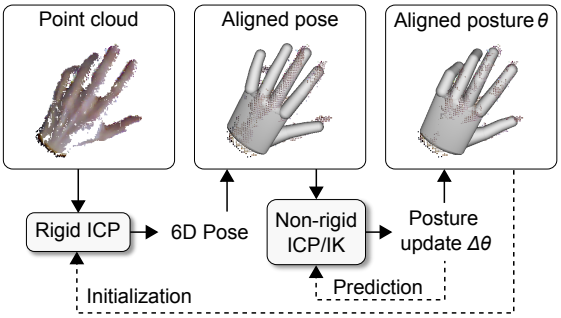

Figure 2: *Schematic overview of the hand tracking process.*

dated and used as input for the iterative IK-based pose and posture estimation. During this process, the model parameters $\boldsymbol{\theta}$ are updated and the effector points $\mathbf{p}_i(\boldsymbol{\theta})$ are moved according to the updated skeleton. This process is iterated until the target-effector error converges, which usually takes less than 10 iterations. The process of recomputing the correspondences and solving the IK optimization is iterated several times in a non-rigid ICP manner. As a result the virtual hand model is aligned with the observation point cloud, which yields the hand pose and posture estimation.

The core component of this hand tracking process is the iterative optimization of model parameters $\boldsymbol{\theta}$ given effector positions $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_k)^T$ and their target positions by $\mathbf{t} = (\mathbf{t}_1, \ldots, \mathbf{t}_k)^T$. The IK problem, $\mathbf{t} = \mathbf{p}(\boldsymbol{\theta})$, is solved by finding a parameter update $\Delta\boldsymbol{\theta}$ to the current state $\boldsymbol{\theta}$ in order to minimize the objective function

$$E(\Delta\boldsymbol{\theta}) = \frac{1}{2}\left\|\mathbf{p}(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) - \mathbf{t}\right\|^2 + \frac{1}{2}\left\|\mathbf{D}\Delta\boldsymbol{\theta}\right\|^2. \qquad (1)$$

The first term penalizes the least squares error between the effectors $\mathbf{p}_i$ and their target positions $\mathbf{t}_i$. The second term regularizes the under-determined problem by damping the parameter update $\Delta\boldsymbol{\theta}$ with a diagonal matrix $\mathbf{D}$. This damping is also used for joint limit avoidance (see [SMRB14]).

The objective function (1) is minimized using a Gauss-Newton approach, where in each iteration a linear system is solved for the parameter update:

$$\left(\mathbf{J}^T\mathbf{J} + \mathbf{D}\right)\Delta\boldsymbol{\theta} = \mathbf{J}^T\left(\mathbf{t} - \mathbf{p}(\boldsymbol{\theta})\right), \qquad (2)$$

where $\mathbf{J} = \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}}$ is the $(3k \times 26)$ Jacobian matrix of the effector positions [Bus09]. This update $\Delta\boldsymbol{\theta}$ has to be scaled to guarantee convergence. As mentioned above, the Gauss-Newton minimization typically converges after 5–10 iterations.

Performing the IK optimization for all 26 kinematic parameters during the tracking process closely aligns the model with the observed sensor data with high freedom of movement. However, if the sensor data is incomplete, ambiguous or noisy, the computed point correspondences can be unreliable and the optimization can therefore result in inaccurate or unnatural hand posture reconstructions.

4. Optimization in PCA subspace

To overcome the above problem of full-DoF tracking, the space of possible hand postures can be reduced in a meaningful way by using hand synergies obtained from a PCA of real human hand posture data. Schröder and colleagues obtained such a dataset by capturing a high variety of human hand movements using a Vicon motion tracking system [SMRB14].

Performing PCA on the set of 20-dimensional posture data (the 6D pose is not considered for dimensionality reduction) and subsequently choosing the eigenvectors corresponding to the l largest eigenvalues yields a $20 \times l$ matrix of principal components, $\mathbf{P}$. The conversion matrix $\mathbf{M}$ that maps from the reduced $(6+l)$-dimensional principal component-space (PC-space) to the $(6+20)$-dimensional parameter space has to consider both pose and posture:

$$\mathbf{M} = \begin{pmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{P} \end{pmatrix}, \qquad (3)$$

where $\mathbf{I}$ is the 6×6 identity matrix passing through the global pose parameters $(\theta_1, \ldots, \theta_6)$. The full parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{26}$ can then be computed from the reduced PC-space parameters $\boldsymbol{\alpha} \in \mathbb{R}^{6+l}$ as

$$\boldsymbol{\theta} = \mathbf{M}\boldsymbol{\alpha} + \boldsymbol{\mu}, \qquad (4)$$

where $\boldsymbol{\mu} \in \mathbb{R}^{26}$ is the mean of the database postures. Given this mapping, the forward kinematics of an effector point $\mathbf{p}_i$ can be written as a function of the PC-space parameters: $\mathbf{p}_i = \mathbf{p}_i(\boldsymbol{\alpha}) = \mathbf{p}_i(\boldsymbol{\theta}(\boldsymbol{\alpha}))$. Optimizing the posture by minimizing the objective function (1) *in PC-space* requires the $3k \times (6+l)$ Jacobian matrix of the effector positions w.r.t. the PC-parameters $\boldsymbol{\alpha}$, which due to the chain rule is

$$\mathbf{J}_{PC} := \frac{\partial \mathbf{p}}{\partial \boldsymbol{\alpha}} = \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\alpha}} = \mathbf{J} \cdot \mathbf{M}.$$

The PC-space parameter update $\Delta\boldsymbol{\alpha}$ is obtained by replacing $\mathbf{J}$ by $\mathbf{J}_{PC}$ and $\mathbf{D}$ by an analogous $(6+l) \times (6+l)$ damping matrix in the Gauss-Newton iterations (2).

This facilitates hand tracking as described in Section 3 in the reduced PC-space, which naturally constrains the estimated hand postures to those represented by linear combinations of the PCs of the posture database. While this improves robustness and performance, it inherently restricts tracking flexibility to a subset of the movements contained in the database. In [SMRB14] a hierarchical optimization was proposed, in which the optimization in a low-dimensional parameter space was followed by local refinements in higher dimensional spaces. While giving good results, this method is not suitable for real-time tracking, firstly due to high computational cost and secondly because the method did not take advantage of temporal coherence, as the local refinements outside of the low-dimensional synergistic subspace were lost across frames, requiring a high number of ICP iterations.

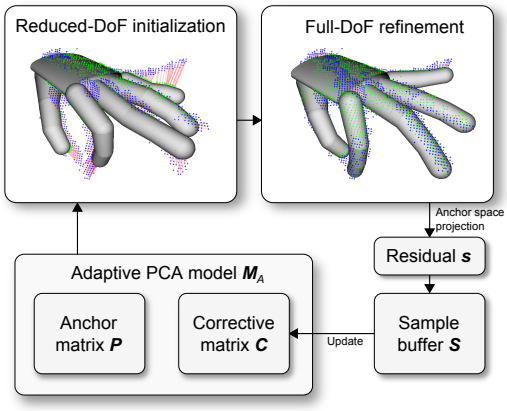

Figure 3: *Online adaptation of the adaptive PCA model.*

5. Online adaptive PCA model

To overcome the limitations related to optimizing the hand posture in a reduced parameter space, we define an adaptive PCA model. This model can be automatically modified to account for newly observed postures which cannot be represented within the initial PCA subspace. To this end, the PC space conversion matrix $\mathbf{M}$ defined in (3) is extended by d columns corresponding to adaptive PC basis vectors, resulting in the $26 \times (6+l+d)$ subspace matrix

$$\mathbf{M}_A = \left(\mathbf{M} \;\middle|\; \begin{matrix} 0 \\ \mathbf{C} \end{matrix} \right) = \left(\begin{matrix} \mathbf{I} & 0 \\ 0 & \mathbf{P} \end{matrix} \;\middle|\; \begin{matrix} 0 \\ \mathbf{C} \end{matrix} \right), \tag{5}$$

where $\mathbf{I}$ is the 6×6 identity matrix, $\mathbf{P}$ is the $20 \times l$ matrix containing the original principal components and $\mathbf{C}$ is a $20 \times d$ matrix containing the "adaptive columns", which by construction will lie in the null space of $\mathbf{P}$.

Following the terminology of [LYYB13], we refer to $\mathbf{P}$ as the *anchor* matrix and $\mathbf{C}$ as the *corrective* matrix. The anchor matrix remains fixed and prevents gradual drift of the PCA model, whereas the corrective matrix is adaptive and represents the observed hand articulations that cannot be represented in the *anchor space* spanned by $\mathbf{M}$. The number of corrective dimensions d depends on the desired flexibility of the adaptive model (see Sections 5.2 and 5.3).

The inverse kinematics posture estimation can be performed in the extended adaptive PC space by using the Jacobian matrix $\mathbf{J}_A = \mathbf{J} \cdot \mathbf{M}_A$ and an analogously extended damping matrix in the Gauss-Newton process (2). The online adaptation of the corrective matrix $\mathbf{C}$ takes place after the non-rigid ICP process, during which the hand posture is estimated using the current subspace matrix $\mathbf{M}_A$ in the inverse kinematics optimization. Figure 3 gives a schematic overview of the adaptation process.

At the beginning of the adaptation procedure, the posture estimate from the current subspace $\mathbf{M}_A$ is refined by an additional IK optimization in the *full* 26-dimensional parameter space. This aligns the model more closely with the observed point cloud and thereby captures details of the user's hand articulation that cannot yet be represented in the adaptive PC space. Since the full-DoF IK optimization starts from a good initial guess (the subspace IK result), it robustly improves the posture low-DoF estimate. The result of this refinement is an updated parameter vector $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{26}$.

Based on this refined posture we compute the anchor space residual $\hat{\mathbf{s}}$ as the orthogonal projection of the refined posture $\hat{\boldsymbol{\theta}}$ onto the complement of the anchor space:

$$\hat{\mathbf{s}} = \left(\mathbf{I} - \mathbf{M}\mathbf{M}^T \right) \left(\hat{\boldsymbol{\theta}} - \boldsymbol{\mu} \right). \tag{6}$$

As the leading six pose DoF entries of $\hat{\mathbf{s}} = (s_1, \ldots, s_{26})^T$ are zero by construction, we only consider the vector of joint angle residuals $\mathbf{s} = (s_7, \ldots, s_{26})^T$ in the following. Intuitively, the residual vector $\mathbf{s}$ represents those aspects of the refined posture that lie outside of the initial PC subspace $\mathbf{P}$. A new residual sample $\mathbf{s}$ is considered valid if $\|\mathbf{s}\|$ (the distance of the refined posture from the anchor space) is above a threshold $s_{\min}$ (significant improvement) and below a threshold $s_{\max}$ (no outlier). In this case, it is stored in a FIFO ring buffer matrix $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_N)$. We use validity bounds $s_{min} = 0.1$ and $s_{max} = 3$ and a buffer size of $N = 250$.

Once the buffer is full, i.e., N frames contributed significant residuals, the corrective matrix $\mathbf{C}$ can be computed by performing PCA on the sample matrix $\mathbf{S}$. Standard methods for PCA compute an eigenvector decomposition of the data covariance, e.g. by singular value decomposition (SVD) of the data matrix, or eigenvalue decomposition (EVD) of the covariance matrix itself. While the former is numerically more accurate, we found that the latter generally has better run-time performance for the rather tall $N \times 20$ matrices in our context. However, neither method scales well with the size of the data matrix (see Table 1).

In the following, we focus on PCA computed by EVD of the sample covariance. The covariance matrix $\mathbf{K}$ of the sample matrix $\mathbf{S}$ is defined as

$$\mathbf{K} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{s}_i - \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}})^T, \tag{7}$$

where $\bar{\mathbf{s}} = \sum_{i=1}^{N} \mathbf{s}_i / N$ is the mean of the sample points. This sum of outer products involves many numerical calculations and can impact performance significantly for large N. Since we need to update the corrective matrix for every new incoming sample $\mathbf{s}_i$, this way of performing PCA can cause a performance bottleneck. In the following, we explore alternative methods for performing PCA in order to compute the corrective matrix $\mathbf{C}$ efficiently.

5.1. Computation of the corrective matrix

In [LYYB13] the iterative expectation maximization (EM) algorithm for PCA presented in [Row98] was used to compute the corrective matrix more efficiently. This algorithm

progressively updates an approximation of a dataset's PC subspace given only a limited number of sample points at a time. A single EM iteration for updating the corrective matrix involves the following calculations:

$$\begin{aligned} &\text{1. E-step:} && \mathbf{Y} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{S} \\ &\text{2. M-step:} && \mathbf{C} = \mathbf{S}\mathbf{Y}^T(\mathbf{Y}\mathbf{Y}^T)^{-1} \end{aligned}$$

These steps are iterated several times before orthonormalizing the result $\mathbf{C}$. While this method can outperform standard PCA methods in some cases, it still involves many numerical operations and calculations, including a matrix orthonormalization, and needs to be iterated 3–4 times to converge [Row98], which altogether diminishes the run-time benefits in our application context, making it perform slightly worse than EVD PCA for large N (see Table 1).

It can be observed that for larger N the cost of the EVD PCA method is dominated *not* by the 20×20 eigenvector decomposition of $\mathbf{K}$, but instead by the computation of the matrix $\mathbf{K}$ itself as in (7), which scales linearly with N.

In contrast, we exploit the incremental nature of the adaptation process by revising the computation of the covariance matrix $\mathbf{K}$ in such a way that allows us to efficiently update the covariance in an incremental way, given a single new corrective sample at a time. The method we propose below results in *constant* costs for computing $\mathbf{K}$ and its eigenvector decomposition—independent of the buffer size N.

We achieve this goal by rewriting the definition of the covariance matrix $\mathbf{K}$ in a way that allows for an incremental adaptation based on rank-one updates. Expanding the outer products in (7) yields

$$\begin{aligned} \mathbf{K} &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{s}_i-\bar{\mathbf{s}})(\mathbf{s}_i-\bar{\mathbf{s}})^T \\ &= \frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{s}_i\mathbf{s}_i^T - \mathbf{s}_i\bar{\mathbf{s}}^T - \bar{\mathbf{s}}\mathbf{s}_i^T + \bar{\mathbf{s}}\bar{\mathbf{s}}^T\right] \\ &= \frac{1}{N}\left[\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T - \left(\sum_{i=1}^{N}\mathbf{s}_i\right)\bar{\mathbf{s}}^T - \bar{\mathbf{s}}\sum_{i=1}^{N}\mathbf{s}_i^T + N\,\bar{\mathbf{s}}\bar{\mathbf{s}}^T\right] \\ &= \frac{1}{N}\left[\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T - N\,\bar{\mathbf{s}}\bar{\mathbf{s}}^T - N\,\bar{\mathbf{s}}\bar{\mathbf{s}}^T + N\,\bar{\mathbf{s}}\bar{\mathbf{s}}^T\right] \\ &= \frac{1}{N}\left[\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T\right] - \bar{\mathbf{s}}\bar{\mathbf{s}}^T. \end{aligned} \tag{8}$$

Based on this expression, the mean and covariance of the sample points are decoupled, allowing us to directly and separately update them given a single new sample point at a time. Once the sample buffer $\mathbf{S}$ is full, the mean $\bar{\mathbf{s}}$ and the covariance $\mathbf{K}$ are explicitly initialized by computing (7). After this, each subsequent incoming sample $\mathbf{s}_{in}$ replaces an old sample $\mathbf{s}_{out}$ in the FIFO ring buffer matrix $\mathbf{S}$. The two samples are then used to directly compute the updated mean $\bar{\mathbf{s}}'$ and covariance $\mathbf{K}'$ in an incremental way:

PCA method	$N = 100$	$N = 500$	$N = 1000$
Jacobi SVD	125μs	272μs	477μs
EVD	60μs	119μs	198μs
Exp. Max.	54μs	137μs	254μs
Inc. cov. (ours)	35μs	35μs	35μs

Table 1: *Run-times for the corrective matrix update using different PCA methods for varying sample buffer sizes. PCA using SVD, EVD and EM scale poorly with increasing buffer sizes, whereas our method runs in constant time.*

$$\bar{\mathbf{s}}' = \bar{\mathbf{s}} + \frac{\mathbf{s}_{in}}{N} - \frac{\mathbf{s}_{out}}{N} \tag{9}$$

$$\mathbf{K}' = \mathbf{K} + \frac{\mathbf{s}_{in}\,\mathbf{s}_{in}^T}{N} - \frac{\mathbf{s}_{out}\,\mathbf{s}_{out}^T}{N} + \bar{\mathbf{s}}\bar{\mathbf{s}}^T - \bar{\mathbf{s}}'\bar{\mathbf{s}}'^T. \tag{10}$$

Update (9) shifts the mean according to the incoming and outgoing samples and update (10) is a series of rank-one updates to the covariance matrix derived from (8), which can be computed efficiently in a single loop. Finally, the new corrective matrix $\mathbf{C}$ is obtained by performing eigenvalue decomposition of the updated covariance matrix $\mathbf{K}'$.

This computation of the corrective matrix is independent from the size N of the buffer matrix $\mathbf{S}$ and therefore allows for an efficient update of the adaptive model in *constant time* given a single new sample. The computational cost of the update is dominated by the eigenvalue decomposition. Table 1 lists average run-times for PCA using SVD, EVD, expectation maximization, and our incremental covariance method and shows the latter to outperform the previous methods.

We note that it is also be possible to directly update the EVD of $\mathbf{K}$ or the SVD of $\mathbf{S}$ after rank-one modifications [BNS78, Bra06], but our approach is more straightforward to implement and provides a significant improvement over non-incremental methods with only minor algorithmic modifications.

5.2. Continuous tracking in adaptive space

The adaptive PCA model allows us to perform local posture refinements after fitting in a reduced PC space without losing temporal coherence. Since increasing the buffer size does not negatively impact run-time performance, we can choose a large buffer size containing a long history of samples. However, while using a long history captures more details missing from the anchor space, it can cause the adaptive PCA model to drift from the initial synergistic model, which can compromise the plausibility of the reconstructed postures. As our tracking system runs at approximately 25 fps, a sample buffer size of $N = 250$ captures a history of new postures for approximately ten seconds, which can fully account for

the local refinements and additionally captures hand articulation details that are not present in the initial synergistic subspace in a robust way.

The analysis of hand postures in [SMRB14] shows that 90% of the variance of a dataset of highly varying hand postures can be represented by six PCs and 90% of the variance of specific hand movements, such as grasping, can be covered by as little as three PCs. Based on these findings we use $l = 3$ dimensions for the initial PC subspace and $d = 3$ corrective dimensions for continuous tracking with the adaptive PCA model. Using more corrective dimensions increases flexibility but comes at the price of losing robustness (see Section 6). Less than three corrective dimensions cause the estimation to be driven mostly by the initial PCs.

5.3. Learning a synergistic model by demonstration

Beyond capturing local posture refinements, the adaptive PCA model can be used to generate a synergistic model from scratch, as an alternative to relying on a pre-recorded database of human hand postures. To this end, the user demonstrates individual hand movements in a training phase, during which the adaptive model learns the corrective DoFs that represent these movements. Then, the anchor space is incrementally expanded to include the trained corrective dimensions.

In the beginning of the learning process, the anchor matrix **P** should be initialized with a single manually defined hand posture. After training the adaptive model by demonstrating a certain new hand movement, the anchor space can be expanded by joining the anchor matrix **P** with the corrective matrix **C** and re-initializing the adaptive model with this new anchor space. As only isolated movements are demonstrated during the training phase, it is sufficient to use $d = 1$ new corrective dimension at a time. Alternatively, more corrective dimensions can be used during training to capture more involved movements and learn multiple synergistic DoFs simultaneously.

6. Results

In the following we show results of our hand tracking system. We first compare the accuracy achieved by the adaptive PCA model to that of a non-adaptive method based on synthetic input data. Then, we provide experimental results of our real-time tracking system using a Kinect camera.

6.1. Evaluation with synthetic data

We evaluate the accuracy of our posture reconstruction by using synthetically generated input point clouds based on known ground truth posture data. The virtual hand model was animated using this posture data and synthetic depth images were generated from a rendering of the virtual hand. These depth images were then used as input for our tracking method. The synthetic images were particularly designed to include a high amount of self-occlusions during complex finger movements.

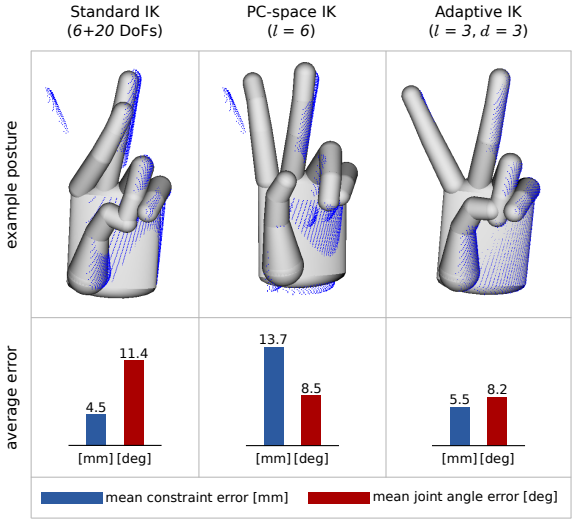

Figure 4: *Posture reconstruction accuracy using the full 26-dimensional parameter space (left), reduced space with 6 PCs (center), and adaptive subspace with 3 anchor DoFs and 3 corrective DoFs (right). The blue points show the input point cloud for one specific frame. The error values are averages over the entire synthetic sequence.*

The generated image sequences were tracked in multiple runs, varying the optimization method (full-DoF, reduced PC-space, adaptive PC-space). For all methods, we measured the difference between the postures generated by our system and the known ground truth postures by computing the average joint angle error. Additionally, we report the average distance between the sensor point cloud and their corresponding points on the hand model surface.

Figure 4 shows exemplary results of this evaluation. For the depicted posture the standard full-DoF optimization produces an inaccurate posture reconstruction due to highly occluded data, which cause bad correspondences. The reduced-DoF IK approximates the ground truth posture more accurately, but due to the inflexibility of the PC-space the hand model is not closely aligned with the point cloud. Our adaptive PCA model produces a result that is closer to the ground truth posture. The average error values over the whole synthetic sequence (Figure 4, bottom) reflect these properties of the different optimization methods. The full-DoF estimation is the least accurate in terms of posture recovery, although producing low constraint errors (partly due to wrong correspondences). The adaptive model combines flexible and accurate reconstruction of the hand animation with the robustness of PC-space optimization.

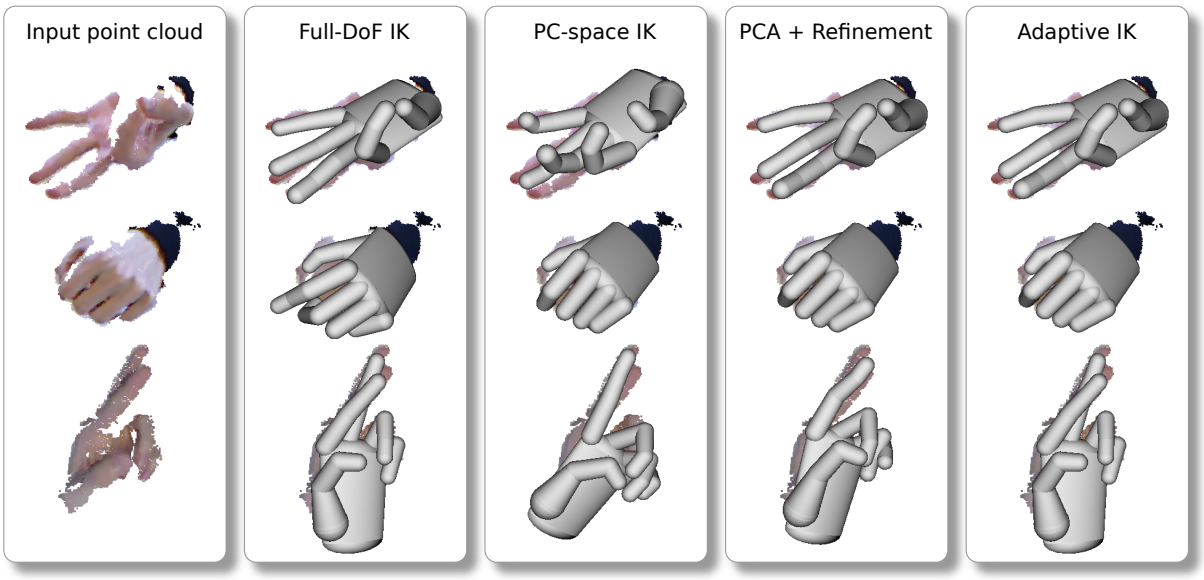

Figure 5: *Tracking results with Kinect input. The full-DoF estimation cannot correctly recover the middle posture due to self-occlusions. Estimation in the reduced PC-space yields a plausible result in spite of these occlusions, but lacks the flexibility to accurately recover the upper and lower postures. Estimation in the fixed PC-space with subsequent refinement cannot recover the lower posture due to the inaccurate PC-space initialization. Our adaptive model successfully recovers all postures.*

When using an adaptive model with $d = 6$ corrective DoFs instead of $d = 3$, the average constraint error slightly decreases from 5.5 mm to 5 mm, but the average posture error increases from 8.2° to 9.4°. This indicates that the flexibility gained by additional corrective DoFs comes at the price of lower estimation quality when incomplete sensor data produces unreliable correspondences.

6.2. PCA and refinement without adaptive model

Our adaptive method continuously updates the PCA subspace to account for deviations from the fixed PCA model, allowing previously unknown observed postures to be incorporated in a temporally coherent way. Performing local posture refinements without subsequently adapting the subspace causes the estimation to be mainly driven by the fixed PCA model, which can produce inaccurate results in cases of unknown hand articulations.

Figure 5, bottom row shows an example where the input hand posture cannot be accurately represented in the fixed PCA subspace (third column). Subsequent full-DoF refinement of the posture based on this initialization without an adaptive model reaches an incorrect local minimum (fourth column). In contrast, estimation with an adaptive model reproduces the input posture well (fifth column), because the adaptive subspace was robustly updated according to the refined observations during the previous frames.

6.3. Experimental results using a Kinect camera

Our tracking system is deployed on an Octa-Core Intel Xeon(R) E5-1620 CPU at 3.60GHz with 8 GB of RAM. Our implementation is heavily parallelized and fully utilizes all eight cores during the correspondence search and the construction of the Jacobian matrix. The tracking system runs at approximately 25 fps. The PCA adaptation procedure involving the full-DoF posture refinement and the computation of the corrective matrix usually takes less than 5 ms to complete and therefore does not negatively impact run-time performance.

Figure 5 shows examples comparing the hand posture reconstruction using the full-DoF, reduced PC-space, reduced PC-space with subsequent refinement and adaptive PC-space optimization for point clouds from a Kinect camera. Our camera setup is arranged with a top-down view of the workspace, which contains minimal clutter to facilitate robust segmentation of the user's hand. A live performance of our tracking system is shown in the accompanying video.

7. Discussion

We presented a hand tracking method that fits a virtual hand model to an RGBD sensor point cloud by performing an inverse kinematics optimization in an adaptive reduced synergistic parameter space. Using an adaptive PCA model constrains the estimation to realistic hand postures while simultaneously allowing for continuous hand articulation refine-

ments. The direct modification of the corrective matrix based on incremental rank-one updates during the online adaptation of the PCA model is efficient and generally useful for applications related to dimension reduction. The overall quality of the results indicates applicability in fields like character animation and robotics.

Future work includes the acceleration of the closest point search and Jacobian construction with an optimized GPU-based implementation. The resulting performance improvements will allow for increased input point cloud density and usage of sensors with higher frame-rate and resolution, which will in turn improve the quality of our refinement and adaptation process. While a higher frame-rate requires a larger buffer size N for our adaptive PCA model in order to cover the same time span, this will not negatively impact run-time performance, since our incremental adaptation method is independent from the buffer size N. The overall tracking quality could be further improved by combining our gradient-based approach with discriminative and probabilistic methods.

Acknowledgments

The authors are grateful to Ulrich Schwanecke for fruitful discussions and to the anonymous reviewers for their helpful suggestions. This work was supported by the DFG Center of Excellence "Cognitive Interaction Technology" (CoE 277: CITEC) and the DFG grant "Real-Time Acquisition and Dynamic Modeling of Human Faces, Upper Bodies, and Hands" (BO 3562/1-1).

References

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-D shapes. *IEEE Trans. on Pattern Anal. Mach. Intell. 14*, 2 (1992), 239–256. 2

[BNS78] BUNCH J., NIELSEN C., SORENSEN D.: Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik 31*, 1 (1978), 31–48. 5

[Bra06] BRAND M.: Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications 415*, 1 (2006), 20 – 30. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems. 5

[BTG*12] BALLAN L., TANEJA A., GALL J., GOOL L. V., POLLEFEYS M.: Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision (ECCV)* (2012), pp. 640–653. 2

[Bus09] BUSS S. R.: Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. Unpublished survey, 2009. 3

[CH05] CHAI J., HODGINS J. K.: Performance animation from low-dimensional control signals. *ACM Transactions on Graphics 24*, 3 (2005), 686–696. 2

[dLGFP11] DE LA GORCE M., FLEET D., PARAGIOS N.: Model-based 3d hand pose estimation from monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 9 (Sept 2011), 1793–1805. 2

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (2004), 522–531. 2

[LWH00] LIN J., WU Y., HUANG T. S.: Modeling the constraints of human hand motion. In *Proceedings of the Workshop on Human Motion (HUMO'00)* (2000), IEEE Computer Society, pp. 121–126. 2

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

[LZWM06] LIU G., ZHANG J., WANG W., MCMILLAN L.: Human motion estimation from a reduced marker set. In *Proceedings of Symposium on Interactive 3D Graphics and Games* (2006), pp. 35–42. 2

[OKA11] OIKONOMIDIS I., KYRIAZIS N., ARGYROS A.: Efficient model-based 3D tracking of hand articulations using Kinect. In *22nd British Machine Vision Conference (BMVC)* (2011), pp. 101.1–101.11. 2

[OKA12] OIKONOMIDIS I., KYRIAZIS N., ARGYROS A.: Tracking the articulated motion of two strongly interacting hands. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012), pp. 1862–1869. 2

[QSW*14] QIAN C., SUN X., WEI Y., TANG X., SUN J.: Real-time and robust hand tracking from depth. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014). 2

[RKK10] ROMERO J., KJELLSTRÖM H., KRAGIC D.: Hands in action: real-time 3D reconstruction of hands in interaction with objects. In *IEEE International Conference on Robotics and Automation (ICRA)* (2010), pp. 458–463. 1

[Row98] ROWEIS S.: EM algorithms for PCA and SPCA. In *Conference on Advances in Neural Information Processing Systems* (1998), pp. 626–632. 4, 5

[RSH02] ROWEIS S., SAUL L. K., HINTON G. E.: Global coordination of local linear models. In *Advances in Neural Information Processing Systems 14* (2002), pp. 889–896. 2

[SMRB14] SCHRÖDER M., MAYCOCK J., RITTER H., BOTSCH M.: Real-time hand tracking using synergistic inverse kinematics. In *IEEE International Conference on Robotics and Automation (ICRA 2014)* (2014), pp. 5447–5454. 1, 2, 3, 6

[UD08] URTASUN R., DARRELL T.: Sparse probabilistic regression for activity-independent human pose inference. In *Computer Vision and Pattern Recognition (CVPR)* (2008), pp. 1–8. 2

[WMZ*13] WANG Y., MIN J., ZHANG J., LIU Y., XU F., DAI Q., CHAI J.: Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics 32*, 4 (2013), 43:1–43:14. 2

[WP09] WANG R. Y., POPOVIĆ J.: Real-time hand-tracking with a color glove. *ACM Transactions on Graphics 28*, 3 (2009), 63:1–63:8. 1

[WPP11] WANG R., PARIS S., POPOVIĆ J.: 6d hands: Markerless hand-tracking for computer aided design. In *Proceedings of ACM Symposium on User Interface Software and Technology* (2011), pp. 549–558. 1

[ZCX12] ZHAO W., CHAI J., XU Y.-Q.: Combining marker-based mocap and RGB-D camera for acquiring high-fidelity hand motion data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), pp. 33–42. 2

[ZZMC13] ZHAO W., ZHANG J., MIN J., CHAI J.: Robust realtime physics-based motion control for human grasping. *ACM Transactions on Graphics 32*, 6 (2013), 207:1–207:12. 2