3D Hand Tracking for Human Computer Interaction

Victor Adrian Prisacariu
University of Oxford
victor@robots.ox.ac.uk

Ian Reid
University of Oxford
ian@robots.ox.ac.uk

January 17, 2012

Abstract

We propose a real-time model-based 3D hand tracker that combines image regions and the signal from an off-the-shelf 3-axis accelerometer placed on the user's hand. The visual regions allow the tracker to cope with occlusions, motion blur and background clutter, while the latter aids with the inherent silhouette-pose ambiguities. The accelerometer and tracker are synchronised by casting the calibration problem as one of principal component analysis. Based on the assumption that, often, the number of possible hand configurations is limited by the activity the hand is engaging in, we use a multiclass pose classifier to distinguish between a number of activity dependent articulated hand configurations. We demonstrate the benefits of our method, both qualitatively and quantitatively, on a variety of video sequences and hand configurations and show a proof-of-concept human computer interface based on our system.

1 Introduction

As computers have metamorphosed from pure computational machines into domestic gaming and media hubs, at the centre of our social lives, so the search for friendlier, more natural ways of interacting with them has gained pace. One goal is to replace the standard mouse and keyboard-type interaction, which limits the user to a constrained work flow, with gesture and touch-based interfaces. In this paper we describe a natural and intuitive way of interacting with a virtual environment, by combining a vision system with an inertial sensor. Our system, while still being cost effective, is able to recover the rigid pose of a human hand, in real time. Furthermore, we do not limit ourselves to a single, fixed, hand configuration. Instead, we are able to distinguish between a small, discrete, subset of such configurations (such as pointing or grasping).

Our method combines a *vision based 3D tracker*, similar to the one presented in [1], with *a single off-the-shelf accelerometer* and a *multiclass visual classifier*.

We use the visual 3D tracker to recover the 6 degree of freedom rigid pose of the hand, with a known (fixed) 3D model. This tracker is model and region based: we use simple 3D meshes as models (so no texture or lighting information) and a region based energy function. We represent the region statistics by colour histograms with a variable bin size, and adapt these online. We use the implicit representation of the contour of the projection of the known 3D model, embedding it inside a level set function, such as a signed distance function. Rigid pose recovery is done by maximising the posterior per-pixel probability of foreground and background membership as a function of pose, directly, bypassing any pre-segmentation phase. To help deal with the ambiguities in the pose–silhouette mapping, we place an accelerometer on the hand.

Tracking the full articulated pose from a single, natural, image of the hand remains an open problem. In this work we choose a compromise between rigid and full articulated tracking, by using an SVM one-against-all multiclass classifier to distinguish between a small subset of articulated hand poses and changing the 3D model as a function of the articulated hand configuration. As input to the classifier, we use a Histogram of Oriented Gradients (HOG) descriptor of 2D hand segmentations.

Our method has the following significant advantages over previous hand tracking work:

- unlike most previous work, such as [2, 3, 4], it works in real time (for fixed hand configurations) and close to real time (for variable hand configurations);
- unlike many edge based works, such as [3], it does not require a prior hand segmentation and is region based, allowing it to work in real-world environments (cluttered and with large amounts of motion blur and occlusions);
- it is much less intrusive, compared to glove-based approaches (such as [5, 6, 7]);
- compared to our previous work of [8], it is not limited to a single, fixed articulated hand pose, but rather is able to choose between different 3D models, based on the configuration of the hand, effectively giving the appearance of articulated tracking.

An early version of this work was presented as the conference paper [8]. In the present paper we elaborate this work and extend it to limited articulated hand tracking (by using a classifier to separate between a small, discrete subset of possible hand configurations). We also provide more detailed experiments, by increasing the number of qualitative experiments and adding several quantitative ones.

The remainder of the article is structured as follows: we begin in Section 2 with an overview of the related works and continue in Section 3 with an overview of our system. In Section 4 we present the visual 3D tracker, in Section 5 we detail the way in which we extract the orientation from the accelerometer and correlate it with the 3D tracker. In Section 6 we show how we switch between different 3D models, based on the articulated pose of the hand. Section 7 shows various results. We conclude in Section 8.

2 Related Work

The standard method of doing fast and reliable hand tracking is to use marker or glove-based approaches. One commercial example is the the ShapeHand data glove [5], which uses a combination of accelerometers, gyroscopes and flex sensors. Two non-commercial examples are [6] and [7]. In [6] the authors use a specially coloured glove (but no sensors) to obtain the full 3D articulated pose of a hand. The SixthSense system of [7] tracks individual fingers (rather than the whole hand) using specially coloured markers on the tips of the fingers. The system is able to track the markers and recognise several hand gestures in real time and with very high accuracy. Tracking is only 2D as the system provides the position of the fingers in *image space* rather than *full 3D space*.

Vision based 3D hand tracking can be split into two categories: *model-based* and *appearance-based* [9]. In *model based* works, an alignment is sought between a predefined model and the image data. Typically this is achieved by defining some error function and optimising it with respect to the pose parameters. *Appearance-based* techniques, in contrast, try to learn a mapping directly from some image-based features to the pose. There have also been attempts at *hybrid methods*. Usually these involve using the 3D model to generate multiple views for training the classifier.

2.1 Model Based Tracking

Model-based methods encompass a fairly broad range of techniques, which mostly all share some common elements: a 3D model, a set of image features, an error function (that characterises the agreement between the model and the image evidence) and a non-linear optimisation algorithm. In this section we will briefly discuss these components, with reference to prior work in model-based hand tracking. Please note that the feature set subsection also applies to appearance-based tracking.

2.1.1 Feature Set

A common approach is first to reduce the image to its constituent *edges*, in, among others, [10, 11, 12, 13, 2]. This is attractive because the edges exhibit invariance to lighting changes and can be rapidly extracted. However they are easily affected by motion blur and/or occlusion. Trackers based on edges are also often prone to being distracted by background "clutter" edges.

In more generic 3D tracking tasks, recently region-based tracking has been shown to be effective [1, 14]. For the case of hands one route to extract a region would be the use of *skin colour*, as used in, among others, [3, 15, 16, 12, 13, 2, 4, 17].

Several types of features can be combined. In [18] *edges* are combined with *optical flow* and *shading information*, in order to refine the model during tracking. The authors test their solution only in controlled environments (so with *known lighting conditions*) but report accurate results. The technique does not scale well to real environments or to variable lighting conditions.

2.1.2 3D Model

In [15] and [4] the authors obtain accurate tracking results via the creation of very accurate 3D hand models, by using both texture and lighting information. Lighting is adapted online in both cases while in [4] the texture is adapted online as well. However this comes at the cost of significant computational effort.

More commonly, the 3D models used for hand tracking are coarse approximations of the shape of the hand. Most are built using cuboids, cylinders or cones for the phalanges and spheres or hemispheres for the joints. For example, in [19] the 3D model is made of truncated cones for the phalanges and spheres for the joints. These methods have the advantage of low processing time, but can easily lead to tracking errors when the model is very different from the tracked object.

We take a somewhat intermediate route, as explained in Section 4.

2.1.3 Error Function and Optimisation

The error function defines the goodness of fit of a model pose hypothesis with the image evidence. Different methods formulate this either in image space (they project down into the image) or in 3D space, by back projecting up. Only rarely does such a formulation have a unique, easily found global minimum, so the ability to deal with the multimodality can be advantageous.

A convenient formulation for the error function is to base it on least squares, by relating the distances between the projection of the 3D model and the silhouette/edges, or between the back-projected silhouette/edges and the 3D model. More complex factors, such as shading or texture information can also be included. When only one hypothesis is desired, the first order (and sometimes second order) derivatives of the energy function with respect to the pose parameters are computed.

Local minima are obtained using the Gauss-Newton algorithm in [20, 21] or using a quasi-Newton method in [4]. In [15] an optimisation technique called *Stochastic-Meta-Descent*, or SMD, is introduced. It is based on classical first-order gradient descent, but with adaptive local step sizes. The SMD method converges three times faster than normal first-order gradient descent.

When multiple hypotheses are pursued a common solution is to use particle filters, for example in [16, 17, 10]. Here, instead of obtaining a single local minimum, a probabilistic distribution over poses is approximated by a number of particles. One problem with particle filters is that they require a large number of particles to produce accurate results. One solution is to combine particle filters with local optimisation. In [16] particle filters are combined with Stochastic-Meta-Descent: after propagating the particles, SMD is applied on each particle. Only a small number of particles (eight) are needed to improve upon the performance of SMD from [15].

2.2 Appearance-based Tracking

Appearance-based algorithms aim to learn a direct mapping between image features and pose. Therefore only two steps are required when a new frame is processed: extract the image features and obtain the pose from the feature–pose mapping. These methods have the advantage (over model-based methods) that no initialisation is need. These approaches are also (theoretically) faster, because most of the processing effort is put into training, which is done offline, rather than in the online processing. To our knowledge there are only a few papers in the 3D articulated hand tracking literature where appearance-based tracking is used, although them among the most successful. The mapping between feature and pose can either be learned inside a database in [22, 23], using a tree-based filter in [2, 13, 12] or via a Relevance Vector Machine in [3].

In [23] a database is created containing uniform sampling of all possible views of the hand shapes the authors want to recognise. Each view is labelled with the hand pose that generated it. A similarity measure based on Chamfer distance (called *approximate directed Chamfer distance*) is introduced and for each frame the most similar image from the database is retrieved. Their system requires a bounding box around the hand, but can deal with clutter inside the bounding box. Unfortunately accuracy is low and the system is very slow (about 15 seconds per frame). Furthermore their system can handle (include in the database) only a small number ($\sim$100) hand shapes.

In [2, 13, 12] the authors try to combine probabilistic tracking with a hierarchical Bayesian filter. The hierarchy is constructed offline from templates generated from a 3D model, by partitioning the pose space. At each frame a posterior distribution $p(pose|imagefeature)$ is computed over this hierarchy. In the articulated motion case they build the filtering tree using PCA on the training data set and than split the subspace into a number of basis configurations, corresponding to each finger being fully extended and fully bent. The likelihood $p(imagefeature|pose)$ is based on edges and pixel colours. They also model the hand dynamics. Their method provides good tracking results, but at the expense of very large processing times (two to three seconds per frame for a 320$\times$240 image).

In [3] the authors train a Relevance Vector Machine [24] (a probabilistic formulation of Support Vector Machines) classifier to relate image features to 3D poses, a technique initially used for human pose recovery in [25]. As image features they use shape contexts [26]. To disambiguate between hand poses, they employ the use of multiple views: descriptors are generated from each camera individually and are concatenated into a higher dimensional descriptor. The RVM classifier is trained using these descriptors. They also use histogram-based skin colour classifier applied on the Cr and Cb channels of the YCbCr chromatic space. The average error for the three camera system is around 2.4°, with the maximum at around 9°. Tracking is not real time, most of the processing time being taken up by the computation of the descriptor ($\sim$170ms per image).

3 Overview

An outline of our algorithm is shown in Figure 1. When an image is received from the camera it is piped through a 2D segmentation step, which uses a 2D level set based segmenter to give us an approximate hand segmentation. This is then refined using graph cuts and rescaled to a standard size. A HOG descriptor is built

from the normalised segmentation and a multiclass SVM classifier is used to determine the configuration of the hand (out of a fixed, small subset of possible configurations). The detected configuration determines the 3D model used in the visual 3D tracker, which is then run on the original image from the camera. An accelerometer (placed on the hand) is used to aid the visual 3D tracker.

We begin by describing the core of our algorithm, the visual 3D tracker, in Section 4. The accelerometer integration follows, in Section 5. Finally, the pose classifier is described in Section 6.

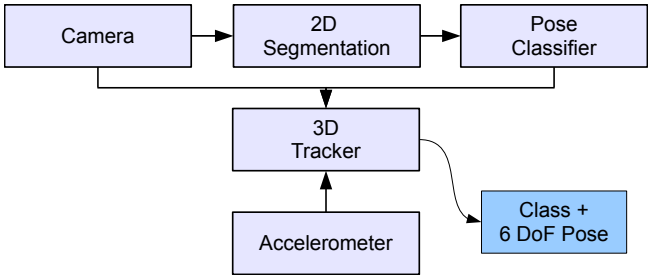

Figure 1: System Overview

4 Visual 3D Tracker

We use the PWP3D tracker of [1] to track rigid hands. This section presents the details of the 3D tracker. We begin by defining notation. We then discuss the 3D model, the tracking algorithm, the region statistics and the implementation.

4.1 Notation

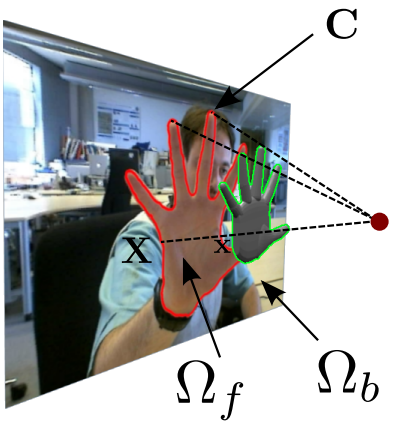

Figure 2: Notation – the contour around the visible part of the 3D model (green) and its corresponding projection $\mathbf{C}$ (red), the foreground region Ω_f and the background region Ω_b, a 2D point $\mathbf{x}$ on the contour and its corresponding 3D point in the object coordinate frame, $\mathbf{X}$

Let the image be denoted by $\mathbf{I}$, and the image domain by $\Omega \subset \mathbf{R}^2$ with the area element $d\Omega$.

Any 3D model can be represented by a collection of 3D points $\mathbf{X_0} = [X_0, Y_0, Z_0]^T \in \mathbf{R}^3$ represented with respect to some convenient (arbitrary) model coordinate system. In camera coordinates each such point is then $\mathbf{X} = [X, Y, Z]^T = \mathbf{R}\mathbf{X_0} + \mathbf{T} \in \mathbf{R}^3$. $\mathbf{R}$ and $\mathbf{T}$ represent the unknown pose and are the rotation matrix and translation vector. In practice we use quaternions to represent rotation, so we optimise over a total of 7 pose parameters (4 for rotation and 3 for translation), denoted by $\boldsymbol{\lambda} = \{\lambda_i | i \in [1, \ldots, 7]\}$.

We assume a calibrated camera, with (f_u, f_v) being the focal distance expressed in horizontal and vertical pixels and (u_o, v_o) the principal point of the camera.

The contour $\mathbf{C}$ in the image (marked with red in Figure 2) is the projection of the contour around the visible part of the object in 3D (marked with green). In our work it is implicitly defined as the zero-level set of an embedding function $\Phi(\mathbf{x})$. It segments the image into two disjoint regions: a foreground region denoted by Ω_f, with an appearance model $P(\mathbf{y}|M_f)$ and a background region Ω_b with an appearance model $P(\mathbf{y}|M_b)$.

Finally, we use $H_e(x)$ to denote the smoothed Heaviside step function and $\delta_e(x)$ to denote the smoothed Dirac delta function.

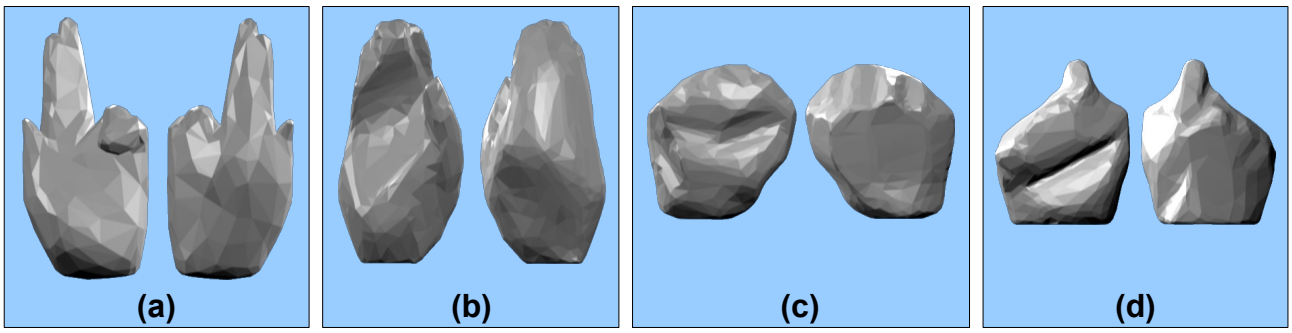

Figure 3: 3D Models. (a) is used in Section 7.2; (b),(c),(d) are used in Section 6.

4.2 3D Models

We use 3D triangle meshes as models for the tracker. In previous work we tracked a single, fixed 3D model [1]. In the present work we dynamically select between various 3D models. Figure 3 shows the four models, each with front and back views, that we employed throughout this paper. Each model was obtained by taking several pictures of the hand (using a normal digital camera) from different angles and combining them using iModeller 3D [27], into a coherent 3D model. One source of error in the recovered pose is the difference between the model and the tracked object. Small differences between the real object and the model (in the millimetre range) can lead to errors less than 5-10 degrees for rotation and 5-10 pixels for translation. Naturally, large differences, such as not modelling the fingers, leads to complete loss of tracking.

4.3 Tracking

The visual 3D tracker used in this work is the PWP3D algorithm of [1].

The contour of the projection of the known 3D model is embedded as the level of a level set function Φ. Tracking is achieved by maximising the log (posterior) probability of this embedding function, given the image data:

$$P(\Phi|\Omega) = \prod_{\mathbf{x}\in\Omega} \Big(H_e(\Phi)P_f + \big(1 - H_e(\Phi)\big)P_b \Big) \Rightarrow \tag{1}$$

$$E(\Phi) = -\sum_{\mathbf{x}\in\Omega} \log \Big(H_e(\Phi)P_f + \big(1 - H_e(\Phi)P_b\big) \Big) \tag{2}$$

where P_f and P_b are defined as:

$$P_f = \frac{P(\mathbf{y}|M_f)}{\eta_f P(\mathbf{y}|M_f) + \eta_b P(\mathbf{y}|M_b)} \quad P_b = \frac{P(\mathbf{y}|M_b)}{\eta_f P(\mathbf{y}|M_f) + \eta_b P(\mathbf{y}|M_b)} \tag{3}$$

where η_f and η_b are the areas of the foreground and background regions respectively:

$$\eta_f = \sum_{\mathbf{x}\in\Omega} H_e\big(\Phi(\mathbf{x})\big) \quad \eta_b = \sum_{\mathbf{x}\in\Omega} 1 - H_e\big(\Phi(\mathbf{x})\big) \tag{4}$$

This energy function is differentiated with respect to the pose parameters λ_i:

$$\frac{\partial E}{\partial \lambda_i} = P(\Phi|\Omega) \sum_{\mathbf{x}\in\Omega} \frac{P_f - P_b}{H_e(\Phi)P_f + \big(1 - H_e(\Phi)\big)P_b} \frac{\partial H_e(\Phi)}{\partial \lambda_i} \tag{5}$$

$$\frac{\partial H_e\big(\Phi(x,y)\big)}{\partial \lambda_i} = \frac{\partial H_e}{\partial \Phi} \left(\frac{\partial \Phi}{\partial x}\frac{\partial x}{\partial \lambda_i} + \frac{\partial \Phi}{\partial y}\frac{\partial y}{\partial \lambda_i} \right) \tag{6}$$

Every contour 2D point is the projection of at least one 3D point $\mathbf{X}$, so:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -f_u \dfrac{X}{Z} - u_0 \\ -f_v \dfrac{Y}{Z} - v_0 \end{bmatrix} \tag{7}$$

Therefore:

$$\frac{\partial x}{\partial \lambda_i} = -f_u \frac{\partial}{\partial \lambda_i}\frac{X}{Z} = -f_u \frac{1}{Z^2}\left(Z\frac{\partial X}{\partial \lambda_i} - X\frac{\partial Z}{\partial \lambda_i}\right)$$
$$\frac{\partial y}{\partial \lambda_i} = -f_v \frac{\partial}{\partial \lambda_i}\frac{Y}{Z} = -f_v \frac{1}{Z^2}\left(Z\frac{\partial Y}{\partial \lambda_i} - Y\frac{\partial Z}{\partial \lambda_i}\right) \quad (8)$$

The differentials $\frac{\partial X}{\partial \lambda_i}$, $\frac{\partial Y}{\partial \lambda_i}$ and $\frac{\partial Z}{\partial \lambda_i}$ follow trivially by differentiating $\mathbf{X} = \mathbf{R}\mathbf{X_0} + \mathbf{T}$ with respect to λ_i.

For more details the reader is referred to [1].

The tracker needs an initialisation pose. In this work we use a predefined pose, which the user must take before tracking begins. While the hand does not explicitly need to be motionless for the tracker to initialise, in our experiments we did start motionless, to avoid motion blur corrupting the histograms.

4.4 Region Statistics

Equation 3 leads straightforwardly to expressions for the posterior foreground/background membership probabilities $P(M_j|\mathbf{Y})$ where M_j, $j \in \{f, b\}$ is the foreground/background model, in which we are implicitly modelling the priors as ratios of areas. This takes no account of temporal considerations. To address this, (inspired by [28]) we introduce temporal consistency by computing $P(M_j^t|\mathbf{Y}^t)$ via a recursive Bayesian estimate:

$$P(M_j^t|\mathbf{Y}^t) = \frac{P(\mathbf{y}^t|M_j^t, \mathbf{Y}^{t-1})P(M_j^t|\mathbf{Y}^{t-1})}{P(\mathbf{y}^t|\mathbf{Y}^{t-1})} \quad (9)$$

where $\mathbf{y}^t$ the value of pixel $\mathbf{y}$ at time t and $\mathbf{Y}_t = [\mathbf{y}^t, \mathbf{y}^{t-1}, \ldots]$ the values of pixel $\mathbf{y}$ up to time t.

Fixed colour likelihoods $P(\mathbf{y}|M_j)$ are computed with respect to colour histograms. We make use of a non-uniform bin size for our histograms. This allows us to take into account the empirical observation that the brightness variation in response to changing illumination is not uniform across the brightness spectrum. One solution would be to use a non-linear correction but this would result in extra processing time. Our solution is expedient, easily tuned, and does not add any extra processing time.

4.5 Implementation and Timings

In order to achieve real-time performance, we use the highly-parallel GPU implementation detailed in our previous papers, [1, 8]. We recapitulate this briefly in this section.

We use gradient descent to minimize Equation 2 and each iteration proceeds as follows:

- The 3D model is rendered using the current estimate of the pose.
- The contour of the rendering and its exact signed distance transform are computed.
- The partial derivatives of the energy function with respect to the pose parameters are computed.
- The pose parameter are adjusted by a step change in the direction of the gradient. The step size is fixed.

All the processing steps are implemented on the GPU, using the NVIDIA CUDA framework. Iterations are run until convergence or until an upper bound (capped to 15 fps) is reached. In general, our average per frame processing time is 50ms.

5 Accelerometer

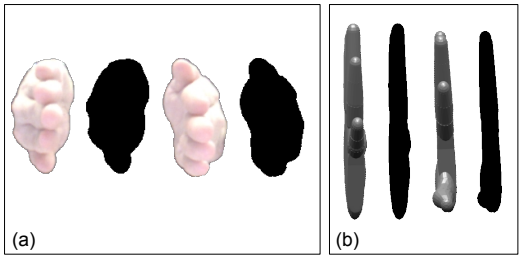

Figure 4: Silhouette - pose ambiguities: (a) real example; (b) artificial example. In both examples poses that are 180 degrees apart generate very similar silhouettes.

Multiple different hand poses can project to the same silhouette, making the mapping from silhouette to pose multimodal. Figure 4 shows four such examples, two real and two artificial. The common solution to this problem is to use multiple hypotheses in the minimisation process. As discussed in the introduction, a relatively high number of particles is necessary to properly explore the search space. Methods for reducing the number of particles do exist, but even if we used a relatively small number of eight particles, such as is done in [16], we would still need 8+ times the processing time of our current visual 3D tracker. The solution we employ in this article is to augment the 3D tracker with orientation information from a single off-the-shelf accelerometer, mounted on the hand. This section discusses the three aspects regarding our usage of the accelerometer: the extraction of orientation data, the calibration and the accelerometer-tracker integration.

5.1 Extracting Hand Orientation from the Accelerometer

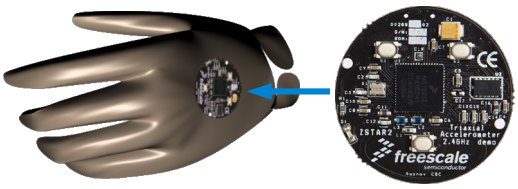

Figure 5: Hand with accelerometer sensor board

In this work we use the Freescale ZSTAR3 dev kit. It consists of a sensor board and an USB dongle. The transmission between sensor and dongle is wireless (through the ZigBee / IEEE 802.15.4 [29] protocol). The sensor board can be mounted either on top of the hand, as shown in Figure 5, or in the palm of the hand. Although the wrist would have been a more convenient point of placement (from the point of view of the user), this would measure arm rotation instead of hand rotation.

This accelerometer measures the direction of gravity summed with the instantaneous linear acceleration. Thus, in the steady state (zero or constant velocity) it returns the "down" (world z) direction.

The absolute orientation on two axes (pitch and roll) can be computed from the gravity vector using the standard Euler angle formulas [30]. Note, however, that rotation around the z axis, in world coordinates (yaw) is not measurable. To avoid issues that arise with angle wrap-around and singularities in the Euler angle representation, we use quaterions and convert the current and previous gravity vector measurements into a relative angle change. Note that this is an approximation, since we are in effect ignoring any linear acceleration the hand undergoes. In our experiments this approximation did not influence the final results in any meaningful way, as the visual tracker minimisation was always able to account for small deviations in the pose given by the accelerometer.

5.2 Accelerometer – Tracker Calibration

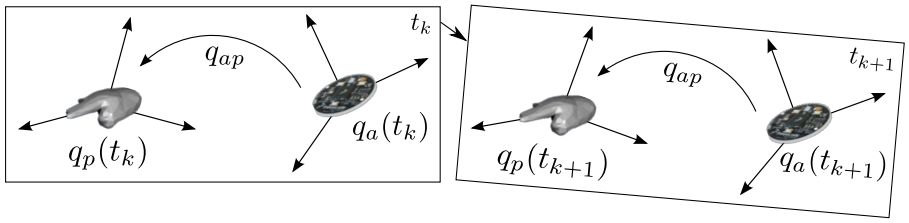

Figure 6: Rotation quaternions and coordinate systems for the visual 3D tracker and the accelerometer

A fundamental problem when combining data from inertial and visual sensors is calibration. Each sensor outputs a value for rotation, under some representation, in its own coordinate system. In order to combine the two, the values need to be converted to the same coordinate system meaning that the (constant) offset between the various rotation sources needs to be obtained. In our case, this is the quaternion offset q_{ap} between the quaternion q_a given by the accelerometer and the quaternion q_p given by the visual 3D tracker (as shown in Figure 6). We can therefore write:

$$q_a q_{ap} = q_p \tag{10}$$

and t_k and t_{k+1} are the frames at the current and next time step.

To collect q_{ap} values, we measure q_a and q_p as the user moves his/her hand trough the entire range of orientations. Figure 7 shows an example such motion.

Figure 7: Example calibration motion

A standard way to solve Equation 10 for a set of q_{ap} values is to use the (linear or nonlinear) least squares framework [31, 32]. For example in [32] the authors maximise:

$$q_{pa} = \arg\max_{q} \{q^T \Big(\sum_{i=1}^{t} \overline{Q}^T_{\Delta q_a}(t) Q_{\Delta q_p}(t) \Big) q\} \tag{11}$$

where $q_{pa} = q_{ap}^{-1}$, Q and $\overline{Q}$ are the matrix representations of Δq_a and Δq_p, and:

$$\Delta q_p = q_p^{-1}(t_1) q_p(t_2) \quad \Delta q_a = q_a^{-1}(t_1) q_a(t_2) \tag{12}$$

Our solution assumes these values lie close together on the rotation sphere and we choose the minimum principal component of the q_{ap} data set as our calibration quaternion. The first three principal components will capture the variance inside the data set (hopefully small), while the fourth (and minimum) principal component will effectively be a noise-corrected mean quaternion.

Ideally the user would calibrate the system before every use, where, typically, the calibration process would take around 30-40 seconds. In our experiments we have noted however that a single, one time, calibration is enough. As long as the user places the accelerometer in roughly the same position and orientation on the hand (within 1 centimetre and 5 degrees, on each axis, from the calibrated pose), the visual tracker is able to compensate for any error in angle.

Figure 8 shows a comparison between our calibration method and the least squares one of [32]. We use two rotations, over 500 frames, to generate two sets of 500 quaterions, $q_1(t)$ and $q_2(t)$. The calibration quaternion between the two rotations is constant and known. We apply uniformly distributed random noise, at each frame and on each axis, within the [-5°, 5°] range. Next we recover q_c, where $q_2(t) = q_1(t) q_c$ is the calibration quaterion, with both methods. At this noise level, both methods yield similar results. We then add a secondary source of noise, a uniformly distributed random noise, at every 60th frame and on each axis, within the [-30°, 30°] range. At this noise level, our method still successfully obtains q_c, while [32] fails.

5.3 Accelerometer – Tracker Integration

To smooth the accelerometer output, we use a Kalman Filter, with a constant acceleration motion model on the raw accelerometer data. Even so, the actual output of the accelerometer is not especially reliable, for three reasons: (i) it remains noisy; (ii) it cannot measure one of the three axis of rotation; (iii) it measures the gravity vector alone only in the steady state, but, when subjected to acceleration, measures the sum of this acceleration and the gravity vector. Therefore we trust the accelerometer data only as a starting point for the visual 3D tracker iterations. We begin with the rotation estimate from the previous frame, update it with the differential motion obtained from the accelerometer, and use this as the starting point for the visual 3D tracker's iterations. This increases the speed and reliability of convergence of the visual 3D tracker as well as overcoming many of the visually ambiguous poses, but places little faith on the actual output of the accelerometer.

6 Pose Classifier

The full articulated pose of the hand has 27 degrees of freedom, split between the 6 necessary to describe the pose of the palm (the rigid pose) and the 21 needed to describe the articulation of the phalanges.

There are several problems when dealing with articulated hands and high dimensional spaces: (i) local minima: searching in a 27 dimensional space is considerably more likely to run into local minima than searching in a 6 dimensional space; (ii) multimodality: as explained above, multiple poses can generate the same silhouette; (iii) speed: tracking speed is inverse proportional to the number of objects tracked i.e. tracking all 19 bones in the hand will be much slower than tracking a single object.

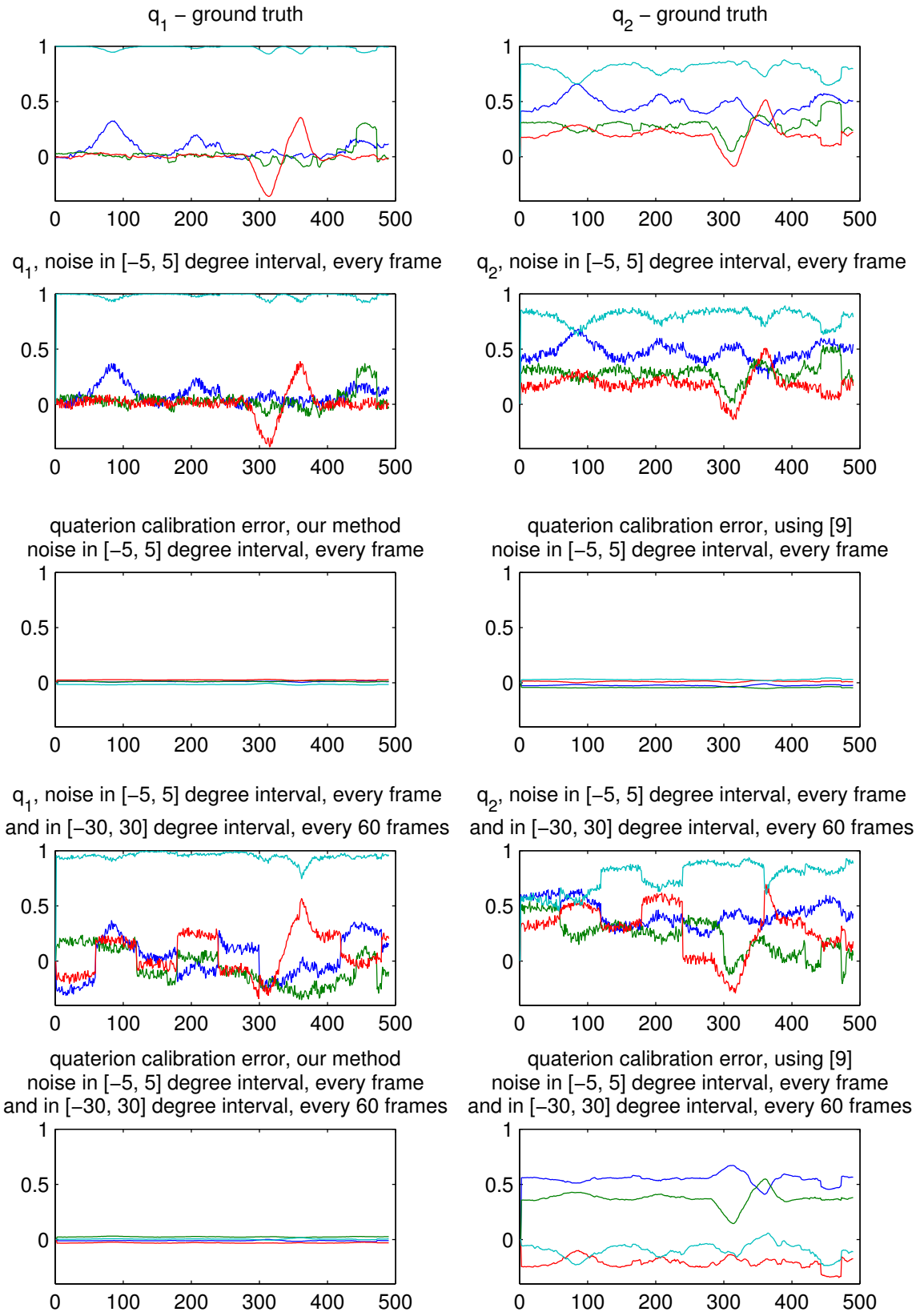

Figure 8: Charts comparing our calibration method to [32]. We use two known 500 frame motions, q_1 and q_2, with a known calibration quaternion. We first add a small amount of noise at each frame (random angle in [-5°, 5°]) and compute the calibration quaternion, using the two methods. Both recover the calibration quaternion successfully. Next, we keep this source of noise and add another, bigger, one every 60 frames (random angle in [-30°, 30°]). Our method again recovers the correct calibration quaternion, while [32] fails.

Often however, the number of possible hand configurations is limited by the activity the hand is engaging in. For example, when picking up a cup, the hand pose will be limited so a small, discrete subset, rather than the full, high dimensional, space. Based on this idea, in this work, we propose a mixture between appearance based and model based methods, which solves some of the above mentioned problems: (i) we segment the hand in 2D and use a multiclass classifier, combined with local features within the boundary of the segmentation, to infer one of several possible hand configurations; (ii) we use separate 3D models for each configuration and the visual 3D tracker (and accelerometer) to track the rigid hand pose parameters. Compared to the full 27 dimensional hand tracking, our solution is a compromise, but is far less likely to run into local minima and can operate at close to real time speeds. Note that, when using the pose classifier, speed does suffer, resulting in an average speed of 5 to 8 fps. A similar methodology, though using a hierarchy of models and a different classifier and not coupling with a 3D rigid pose tracker, was used in [12].

Figure 9: Pose classification overview

The remainder of this section details each step of our pose classifier, as depicted in Figure 9.

6.1 2D Hand Segmentation

The first step of our proposed solution is to segment the hand. For this we use the segmentation phase of the PWP tracker of [33] (upon which our PWP3D tracker is based) thus performing an unconstrained evolution of a pixel-wise posteriors level set energy function similar to that of Equation 2:

$$E(\Phi) = -\sum_{\mathbf{x}\in\Omega}\left(\log\Big(H_e(\Phi)P_f + \big(1-H_e(\Phi)P_b\big)\Big) - \frac{(|\nabla\Phi(\mathbf{x})|-1)^2}{2\sigma^2}\right) \tag{13}$$

The difference between this energy function and the PWP3D one of Equation 2 is the added geometric prior on Φ, which rewards a signed distance function, as proposed in [34]. All the other terms have the same meaning (see Subsection 4.1).

The unconstrained evolution is done by differentiating Equation 13 with respect to the level set function:

$$\frac{\partial P(\Phi|\Omega)}{\partial\Phi} = \frac{\delta_e(P_f - P_b)}{H_e(\Phi)P_f + \big(1-H_e(\Phi)\big)P_b} - \frac{1}{\sigma^2}\left[\nabla^2\Phi - \mathrm{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right)\right] \tag{14}$$

For more details, the reader is referred to [33].

Figure 10: Example coarse hand segmentations obtained by the PWP algorithm.

The PWP algorithm combines a tracking and a segmentation phase. The tracking phase is very fast (running at over 100Hz), but the segmentation phase can be slow as it can require many iterations for full convergence. Since we use the 2D tracker to provide segmentation, we would need to iterate the level set minimisation to complete convergence, making the full PWP segmentation too slow for our needs. Our solution is to obtain coarse segmentations (Figure 10) using the PWP tracker, by running a fixed, small, number of iterations for the level set evolution, and refine the segmentation using graph cuts.

Using the output from the level set tracker and Equation 3, we define a per pixel trimap. Foreground pixels (those that have $P_f >> P_b$) are connected to the source with a very high, preset weight and to the sink with weight 0, background pixels (those with $P_b >> P_f$) the opposite, and the remaining pixels are connected to the source with weight P_f and to the sink with weight P_b. The binary potentials between neighbouring pixels is the L2 distance between their RGB colour values. We then use the Boykov-Kolmogorov Min-Cut/Max-Flow algorithm [35] to obtain the final segmentation.

The final segmentation windows are normalised to a fixed size (112×128 pixels, in this work). Examples of final, normalised segmentations are shown in Figure 11.

Figure 11: Example final segmentation results.

6.2 Descriptor

The next step is to build Histogram of Oriented Gradients (HOG) [36] descriptors for the normalised segmentations.

HOG divides each descriptor window into groups of pixels, called cells. For each cell, a histogram of gradient orientations is computed. We use 4×4 pixel cells and 5 histogram bins, evenly spaced between 0 and 180 degrees. Multiple cells are grouped into blocks, with each cell being part of multiple blocks. We use 2×2 blocks. Finally, each block histogram (consisting of 2×2 cell histograms) is normalized using the L2-Hys norm [37, 36].

6.3 Multiclass Classifier

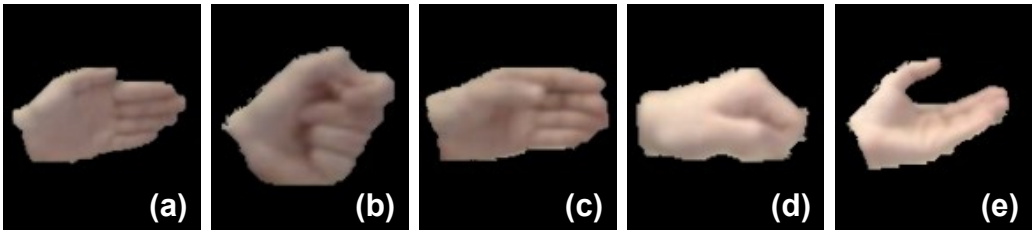

Figure 12: Example training data: (a) – class 1; (b) – class 2; (c) – class 3; (d), (e) – negative class instances. The models for (a), (b) and (c) are shown in Figure 3.

The final step in our algorithm is to use the one-against-all multiclass SVM algorithm of [38]. In this example we use 4 classes (Figure 12): three for the various hand configurations (Figure 3 (b), (c) and (d)) and one for unmodeled configurations. The classes are trained to recognise a fixed hand shape in a specific pose. Based on the output from the classifier, we change the 3D model for the visual 3D tracker: when one of the positive classes is detected we change accordingly, when the negative class is detected (the hand with an unfamiliar shape and/or in an unfamiliar pose) we leave the 3D model unchanged.

For both the descriptor computation and the classification, we use the NVIDIA CUDA GPU implementation, of [39], which we extended to one-against-all multiclass classification. The processing time for both stages (summed together) is usually less than *2ms*.

7 Results

In this section we show the results obtained by applying our algorithm to various video sequences and hand configurations. We showcase the benefits of using a region based method and the accelerometer, and then show a typical failure case. This is followed by an example of our algorithm successfully tracking different articulated hand poses. Next we show a quantitative comparison between the poses produced by our tracker (with and without the accelerometer) and the ground truth. We finish by detailing a proof-of-concept user interface, based on our 3D tracker.

We have based our system on the region-based tracker we have previously described in [1]. The virtue of regions is that unlike high frequency features such as edges and corners, regions exhibit robustness to motion blur and this was demonstrated in our previous work. Our experience has shown that the target motion between frames is limited by a requirement for a 5% overlap between the predicted and actual image positions, and when this is satisfied, the amount of motion blur is irrelevant. Similarly, the tracker is able to recover the pose with a mean error of 20-30mm in the position of the tip of the index finger with 20-30% of hand visible.

Figure 13 and Figure 14 contain two filmstrips, showcasing the benefits of using the accelerometer. In Figure 13, for each frame, we show the original camera image and two OpenGL renderings of the hand, with the determined pose, with and without the accelerometer. In Figure 14, for each frame, we show a posterior map (the per pixel difference $P_f - P_b$, where $P_f - P_b > 0$, with P_f and P_b defined according to Equation 3) and two

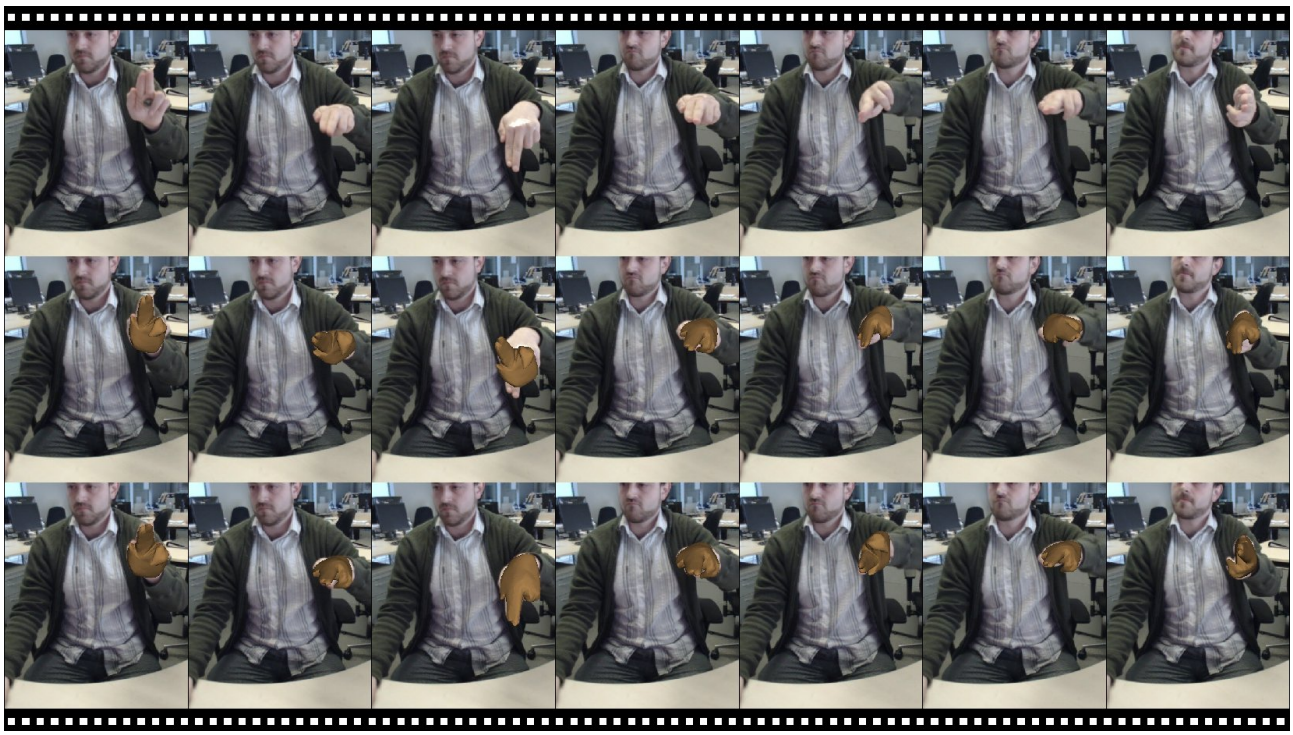

Figure 13: Filmstrip showing tracking results with and without the accelerometer. The top row shows the original camera image. The middle and bottom rows show OpenGL renderings of the hand in the recovered pose, without and with the use of the accelerometer, respectively. The visual 3D tracker alone cannot deal with ambiguities in the silhouette (where a change in pose does not alter the silhouette). The accelerometer provides enough information to discriminate between the ambiguous poses.

wireframe renderings, created by our rendering engine and used by algorithm, also with the determined pose and with and without the accelerometer. The posterior map is a measure of how effective the regions statistics are at separating foreground from background. In the first example the hand was first moving up/down and then left/right, very quickly, in a frontal camera view. In the second example the hand was moving up/down, but at a side camera view. Each motion took up to 5 frames at 30 fps video (so 1/6 seconds), followed by a short (around 60 frames) period of stability. In both examples, initially, when there are no ambiguities, the results are similar, with and without the accelerometer. At the second frame, also in both examples, it can already be seen that the visual tracker alone is showing some error in the recovered pose. This can be explained in both cases by the fact that, when the accelerometer is not used, the visual tracker does not have time to converge fully, whereas, when the accelerometer was used, the starting point is closer to the actual pose, so fewer iterations are required. At the next frame, again in both examples, the pose recovered by the visual tracker is very far from the correct one. In the first example, the frontal view of the hand is an ambiguous pose i.e. a change in hand rotation will not alter the silhouette. In the second example, the visual tracker converges to a local minimum because it was not allowed to converge fully at the previous frame. In the first example the hand moves back up, allowing the visual tracker to recover, albeit not perfectly. Again, when the accelerometer is used, the recovered pose is more accurate. The same behaviour can be noticed in the left/right rotation (next 4 frames of the first example). In the second example the hand does not go back to a pose where the visual tracker can recover, so the results are incorrect for a longer period of time. Quantitative results showing the benefits of using the accelerometer are shown in Section 7.1.

Next we show a typical failure case, in Figure 15. The accelerometer only provides information on two axis, pitch and roll. Its measurements are not affected by changes in yaw. This means that, when the visual tracker encounters ambiguities on the yaw axis, it will not be able to disambiguate between them. The algorithm can also fail in recovering translation, when the hand moves fast enough for the projection not to overlap with the object in the image, at consecutive frames.

In Figure 16 we show our system correctly classifying between different configurations (articulated poses) of the hand, and switching to the correct 3D model. The top row shows the camera image together with the normalised descriptor (bottom left). The contour of the projection is also overlayed, in red. The middle row shows the recovered pose, with the appropriate 3D model. In this example we used four classes, three positive and one negative. When the classifier finds a negative class frame, it uses the class and 3D model from the most recent frame. The bottom rows show a proof-of-concept augmented reality application, where such a technology

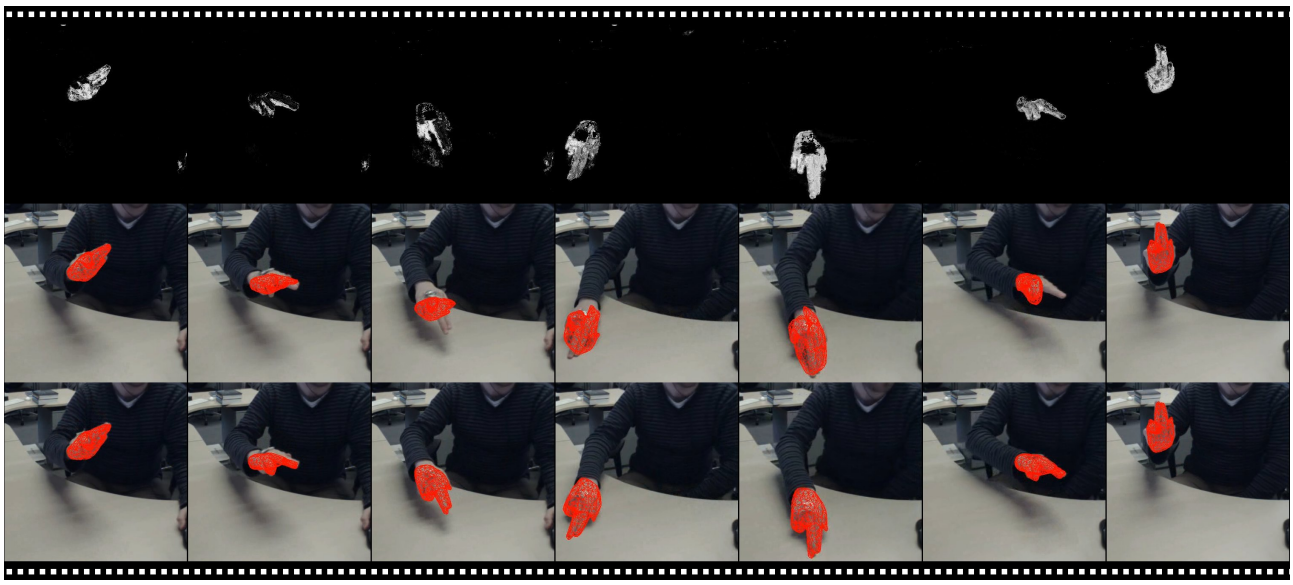

Figure 14: Filmstrip showing tracking results with and without the accelerometer. The top row shows the posterior map (the per pixel difference $P_f - P_b$, where $P_f - P_b > 0$). The middle and bottom rows show wireframe renderings made by our rendering engine and used by the visual tracker. The maximum allowed number of iterations was not enough for the visual tracker alone to converge, which quickly finds an incorrect local minima. The accelerometer positions the visual tracker closer to the correct result, so fewer iterations are required and the results are better.

could be used, by rendering a virtual cup and a virtual tennis racket, as it would appear if it were held by the hand.

7.1 Quantitative Benchmark

Quantitative comparison against absolute ground truth is particularly challenging, and to our knowledge, no datasets exist that would enable a fair comparison. To that end, here we present a methodology for testing our system against an alternative input method, and we provide the means to reproduce our comparison data. We generate absolute positions of a fingertip by moving a hand on the surface of a tablet pc, simultaneously tracking the hand using our system (both with and without the accelerometer). Using the Robust Planar Pose algorithm of [40], we recover the pose of the tablet pc touch screen relative to the camera, allowing us to compare two sets of poses in the same coordinate system. For this experiment we used an HP TouchPad tablet pc, running Android 2.3.7. Strictly speaking, the touch pad does not provide an absolute ground truth and its accuracy is limited by the area of the finger in contact and the resolution of the touch-sensitive array on the screen. To measure the uncertainty, the user was instructed to tap the screen at various points and we measured the standard deviation of the user's input accuracy. This then yields a confidence interval for the "ground truth" tablet measurements against which we compare our method. This is in effect the area of the tip of the finger, being equal to 5.0752 millimetres (x axis) and 4.2035 (y axis) in touch screen 2D coordinates or 4.6936 (x axis), 2.8347 (y axis) and 3.6554 (z axis) in 3D camera coordinates. To serve the interests of those facing a similar challenge of evaluating visual tracking accuracy, our results and Matlab and Android code needed to replicate the experiments, are available online at www.robots.ox.ac.uk/ActiveVision.

Our first experiment focused on translation. Figure 17 shows quantitative results and Figure 19 shows example frames and qualitative results. Here the hand was moved in a rectangular pattern, aiming to keep it at a fixed angle. Since we only use the accelerometer readings to provide between frame rotation data, they were not necessary for this experiment. We obtain errors of 1.66146 millimetres (x axis), 1.81471 millimetres (y axis) and 7.21119 millimetres (z axis).

The second experiment focused on rotation. Figure 18 shows the quantitative results and Figure 20 shows example frames and the qualitative results. Here the hand was moved in a circular pattern, aiming to vary only the angle. Without the accelerometer, the method undergoes catastrophic failure, the mean error on the z axis being 31.8893 millimetres. While the errors on the other axis are much smaller (8.5125 millimetres (x axis) and 2.20229 millimetres (y axis)), the recovered pose would not be usable for any human computer interaction system. Failure here is caused by ambiguities in the silhouette to pose mapping, as, on the one hand, there exists an ambiguity between scale (z axis) and rotation and, on the other hand, between different values for rotation. When using the accelerometer the error on the z axis decreases considerably, to 10.665 millimetres.

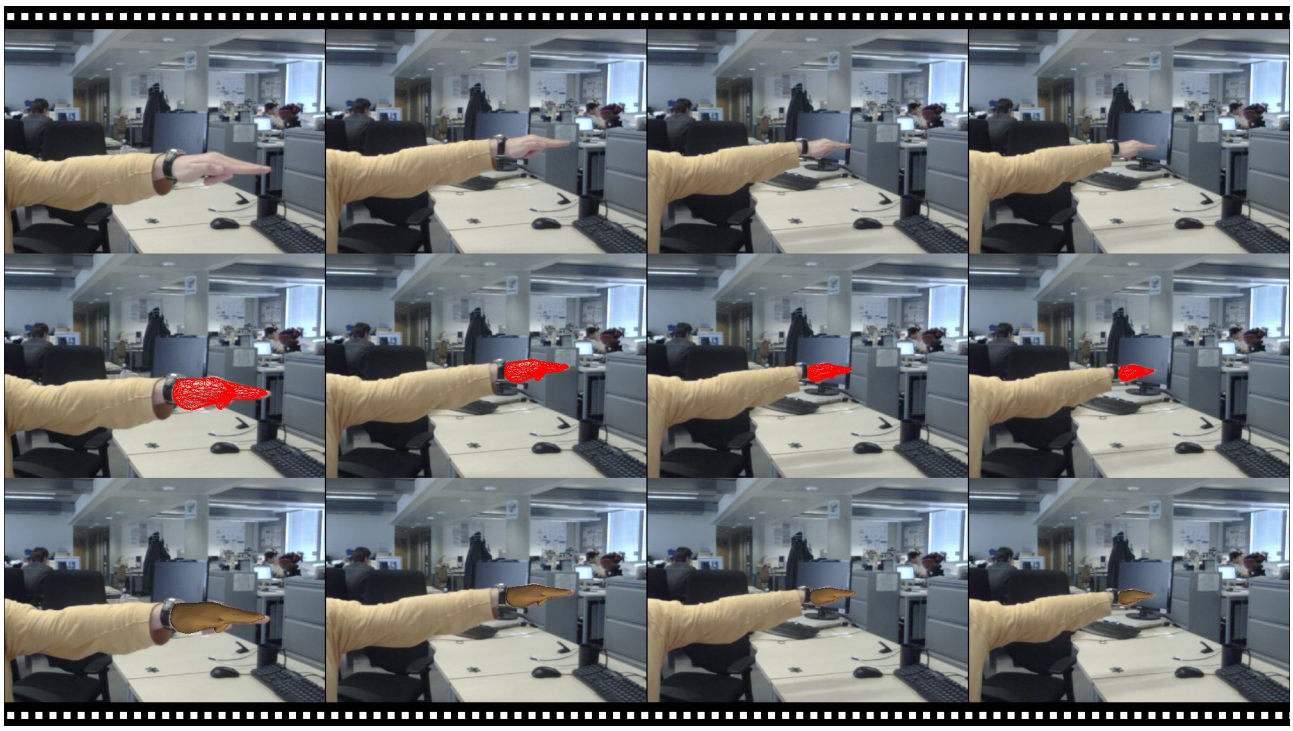

Figure 15: Filmstrip showing tracking a typical failure case. The accelerometer does not provide any yaw, so it cannot help the visual tracker disambiguate between similar silhouettes.

The errors on the x and y axis remain comparably small: 3.11955 (x axis) and 5.07215 (y axis).

7.2 Proof-of-concept Human Computer Interface

In Figure 21 we show several filmstrips depicting a proof-of-concept human computer interface based on our hand tracker. The user first has to mark a plane, by moving the hand along the edges of a rectangular board (Figure 21 – top filmstrip). The shape and colour of the board do not influence the hand tracking in any way. The plane does not need to lie on a real surface i.e. it could have been marked in mid-air. We chose this shape as a visual cue to the user and because it provides an easy and structured way of displaying information. The calibration motion could have been different (for example a circle). To extract the plane position we used RANSAC on the 3D positions of the tip of the index finger (which we computed using the pose of the full hand and the 3D model). After the plane has been marked, it can be "clicked", by moving the hand so that the distance between the tip of the index finger and the plane is 0 or negative. We display a menu interface on the virtual surface, which the user navigates by clicking. The UI is displayed on the computer screen but could also be projected on the board or shown through a pair of AR glasses.

In the first example (Figure 21 – middle filmstrip) the user is prompted to place a DVD on the marked surface. We used two video game DVDs as examples. The system identifies the DVD and displays relevant information on the virtual surface. Recognition is achieved by using the TinEye API [41] and the information is extracted from sources such as Amazon and Wikipedia. The user is also shown a 3D object, related to the identified DVD. For the first DVD, the main starship from the game is shown and for the second DVD a 3D model of planet Mars. The user can interact with the 3D object. In the case of the first DVD the user can place the hand under the starship which then "climbs" on top of the hand. The user can move the ship by moving the hand. In the case of the second DVD the user can rotate the model of planet Mars by moving the hand.

In the second example (Figure 21 – bottom filmstrip) the user first draws a 2D closed shape on the virtual surface. The 2D shape can be turned into a 3D object (by clicking inside it and raising the hand above the virtual surface). The 3D object can then be picked up, similar to the previous example (by placing the hand under it and waiting for it to "climb" on top of the hand).

8 Conclusions

In this work we have proposed a 3D hand tracker which combines a generic level set, region and model based 3D tracker, with an off-the-shelf accelerometer and a multiclass one-versus-all pose classifier. Our method is robust to occlusions, motion blur, can work in cluttered environments and can often deal with ambiguities in

Figure 16: Our system tracking multiple hand configurations and proof-of-concept augmented reality application.

the mapping from silhouette to pose. We can also distinguish between a number of activity based possible hand configurations.

Our system does have several failure scenarios. First, while the accelerometer does often help deal with pose–silhouette ambiguities, it cannot measure yaw, meaning that the 3D tracker can still fail to reach a correct pose. One possible solution would be the use of a more complex IMU. Second, like most region-based tracker, the speed of movement is constrained by the need to have some overlap in the positions of the hand at consecutive frames. At present we do not make predictions about future motion such as could be provided by a motion model, but it would be relatively straight forward to incorporate. Finally, our pose classifier is limited to a (small) discrete number of possible hand configurations and poses. This works well for a variety of activities, since people often move their hands rapidly between "standard" shapes, such as open hand and grasping, and our method could be easily extended to more configurations if required. However if continuous articulations of a fully dexterous motion must be tracked, modelling the continuous changes in some manner may be a better long-term approach.

A straightforward extensional to our work would be the use of depth data. A more interesting extension would be the ability to learn and use shared shape spaces [42], which would allow for continuous, fully articulated 3D tracking.

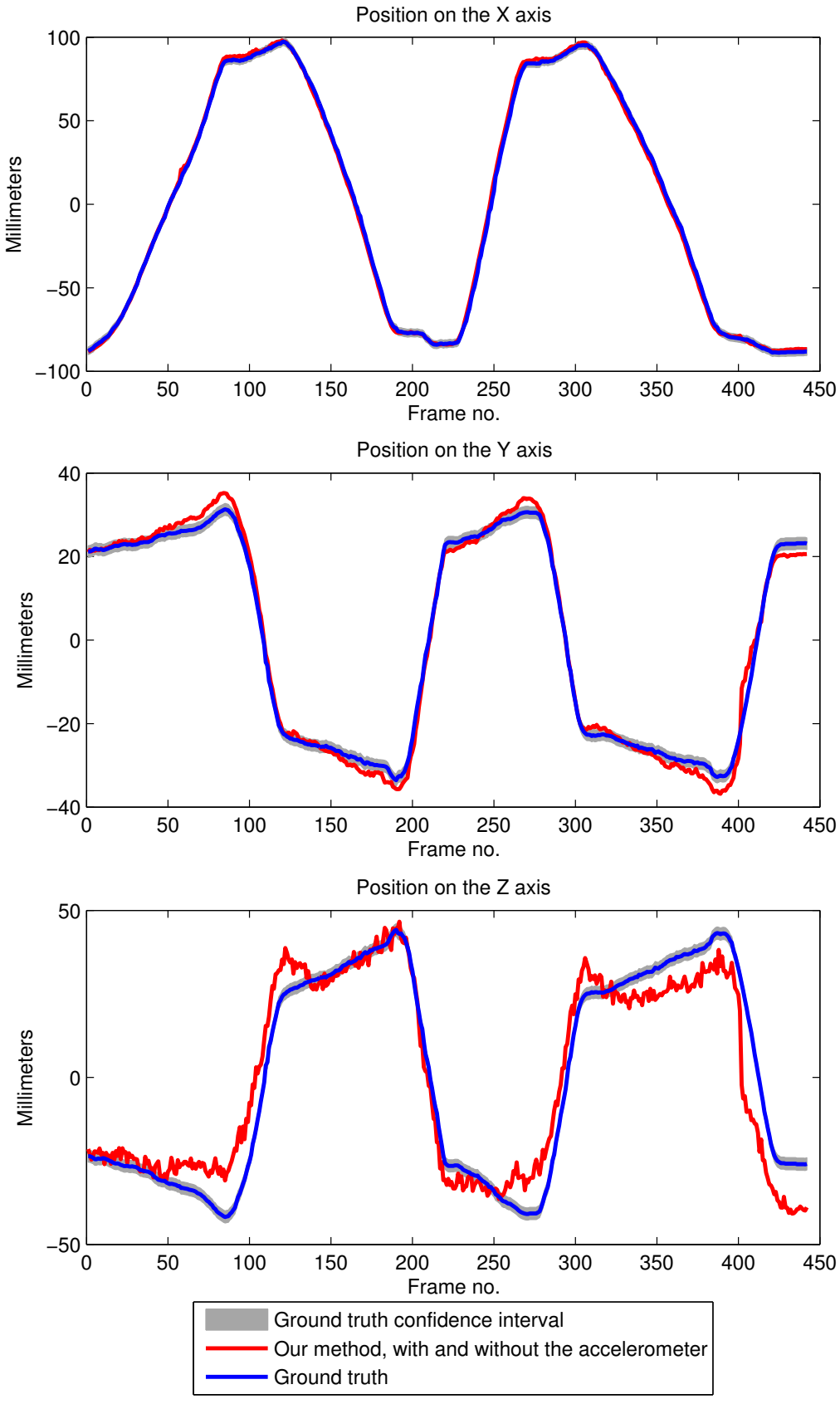

Figure 17: Comparison between our method and the ground truth, focusing on translation.

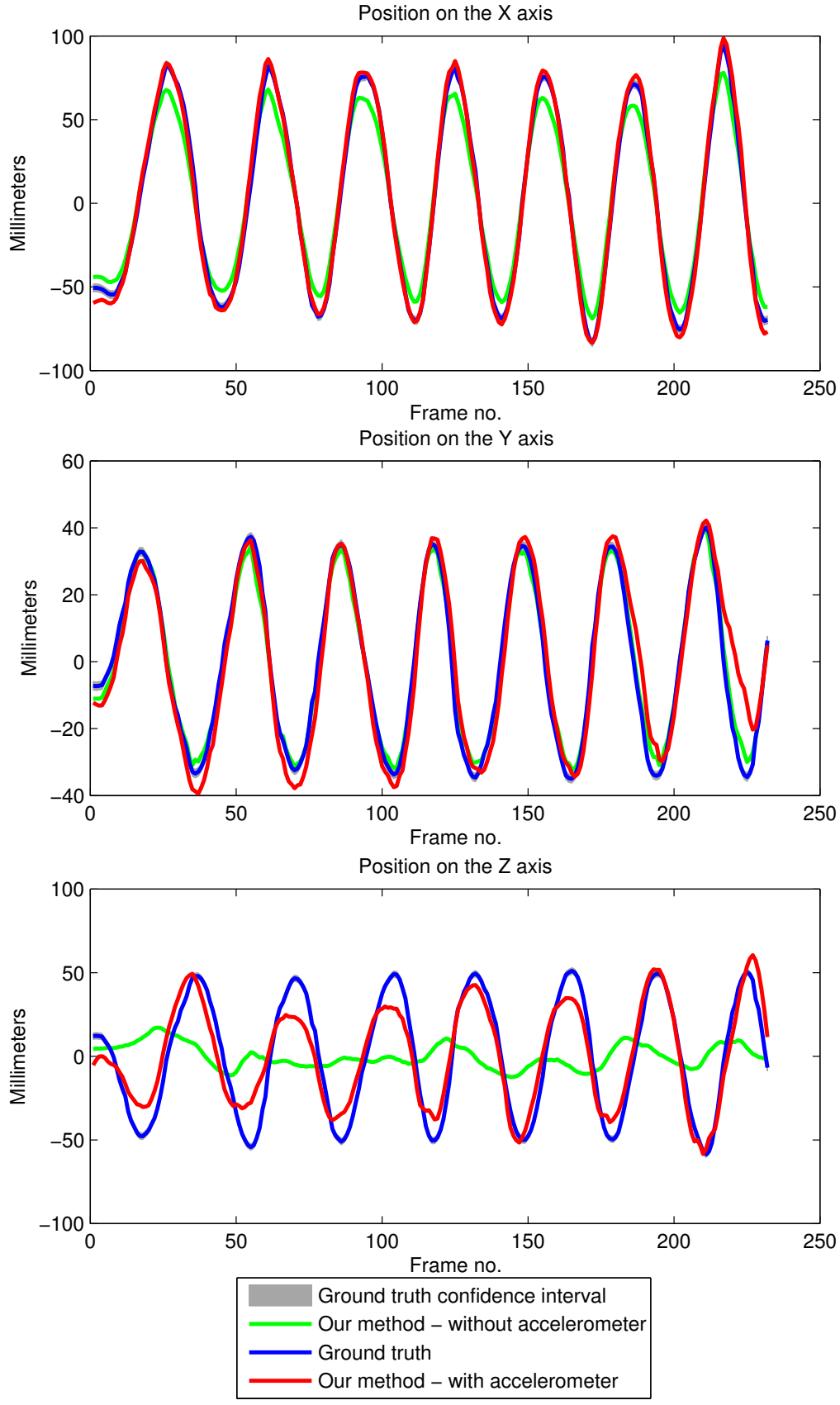

Figure 18: Comparison between our method and the ground truth, focusing on rotation.

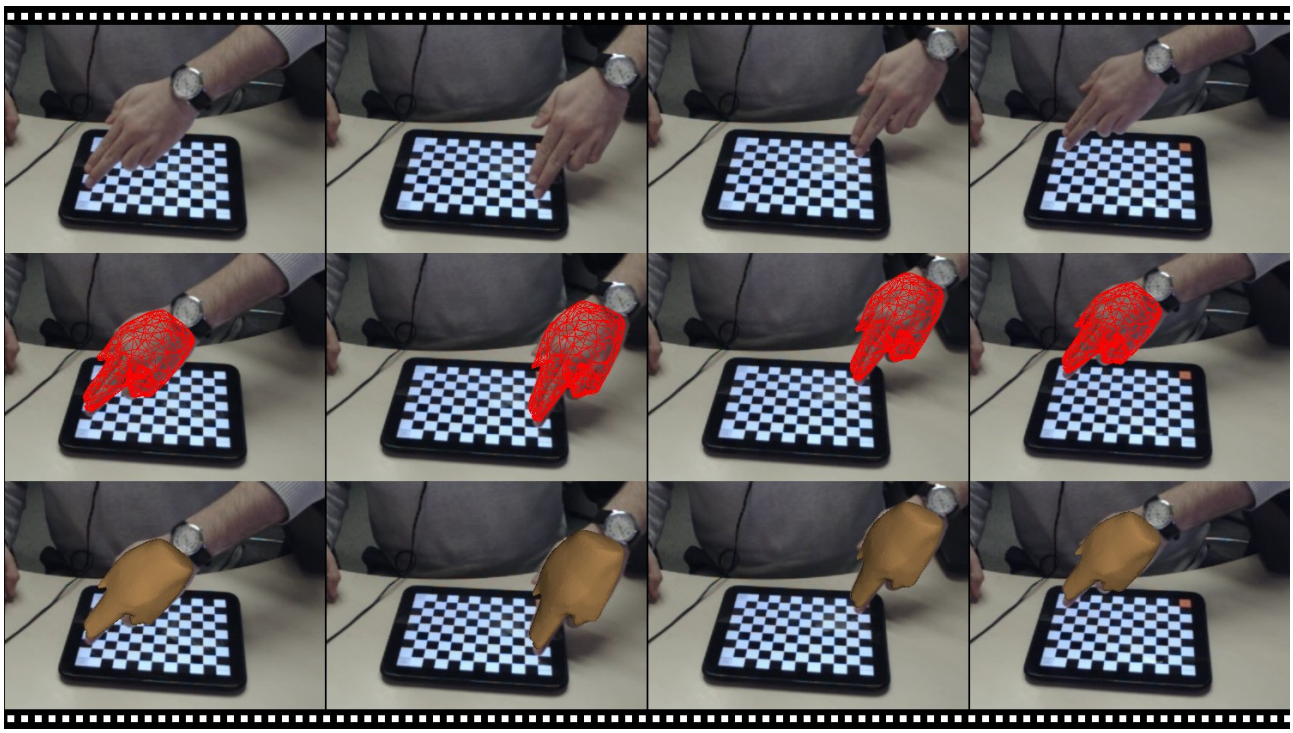

Figure 19: Example frames and results from the translation accuracy test. Original image (first row), wireframe (second row) and full rendering (third row) results for our method, with and without the accelerometer.

References

[1] V. Prisacariu, I. Reid, PWP3D: Real-time segmentation and tracking of 3D objects, in: International Journal on Computer Vision, 2011.

[2] B. Stenger, A. Thayananthan, P. Torr, R. Cipolla, Filtering using a tree-based estimator, in: International Conference on Computer Vision, Vol. 2, 2003, pp. 1063–1071.

[3] T. E. de Campos, D. W. Murray, Regression-based hand pose estimation from multiple cameras, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 1, 2006, pp. 782–789.

[4] M. de La Gorce, N. Paragios, D. J. Fleet, Model-based hand tracking with texture, shading and self-occlusions, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[5] M. Inc., Shapehand data glove (2009).
URL `http://www.shapehand.com/`

[6] R. Y. Wang, J. Popović, Real-time hand-tracking with a color glove, ACM Transactions on Graphics 28.

[7] P. Maes, SixthSense: Integrating information and the real world, in: IEEE International Symposium on Mixed and Augmented Reality, 2009, pp. 21–27.

[8] V. A. Prisacariu, I. Reid, Robust 3D hand tracking for human computer interaction, in: IEEE International Conference on Automatic Face and Gesture Recognition, 2011, pp. 368–375.

[9] V. I. Pavlovic, R. Sharma, T. S. Huang, Visual interpretation of hand gestures for human-computer interaction: A review, IEEE Transactions Pattern Analysis and Machine Intelligence 19 (1997) 677–695.

[10] J. MacCormick, M. Isard, Partitioned sampling, articulated objects, and interface-quality hand tracking, in: European Conference on Computer Vision, 2000, pp. 3–19.

[11] H. Zhou, T. S. Huang, Tracking articulated hand motion with eigen dynamics analysis, in: International Conference on Computer Vision, 2003, pp. 1102–1110.

[12] B. Stenger, A. Thayananthan, P. H. S. Torr, R. Cipolla, Model-based hand tracking using a hierarchical bayesian filter, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006) 1372–1384.

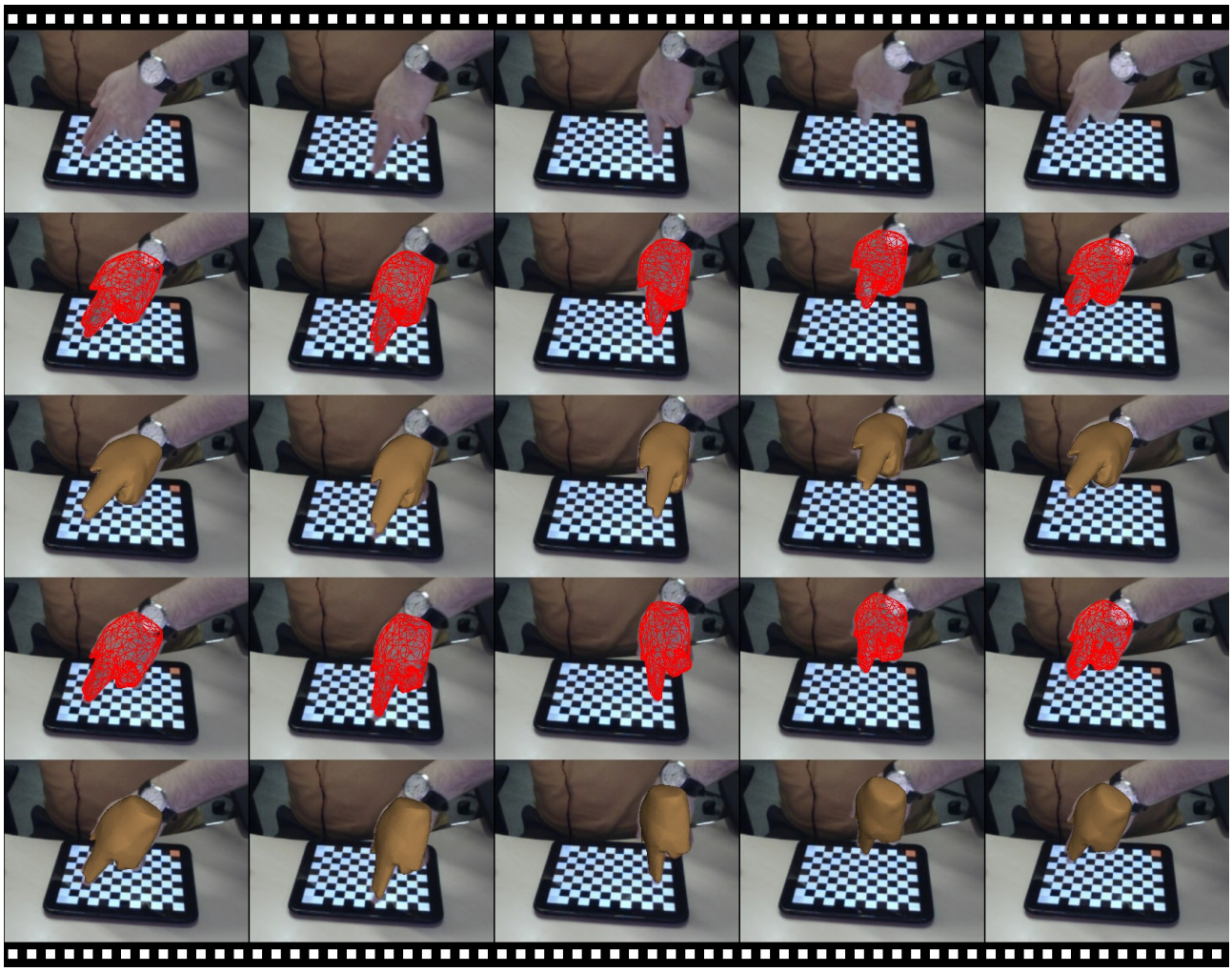

Figure 20: Example frames and results from the rotation accuracy test. Original image (first row), wireframe (second row) and full rendering (third row) results of our method, without the accelerometer, wireframe (forth row) and full rendering (fifth row) results of our method, with the accelerometer.

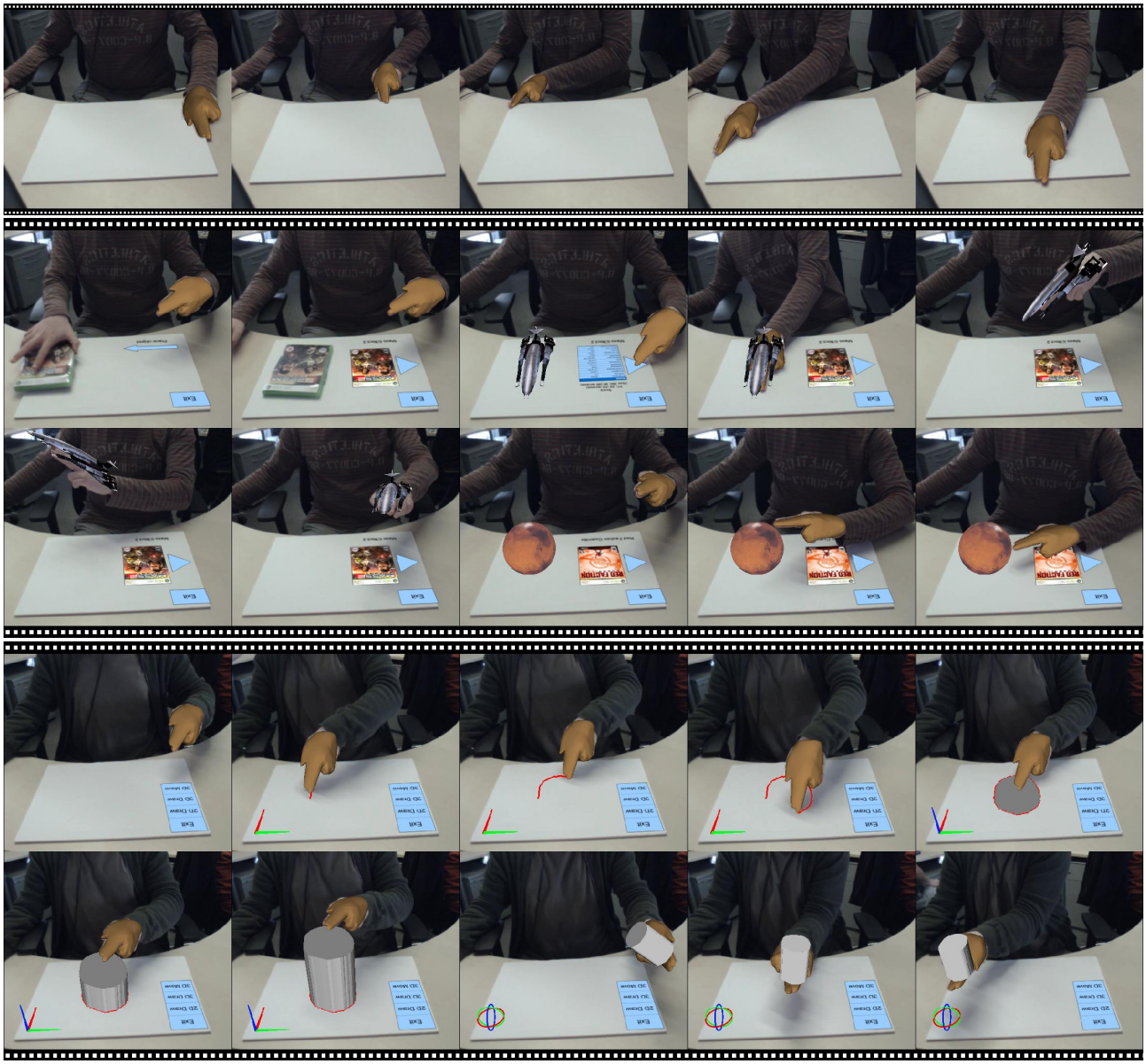

Figure 21: Example human-computer interface based on our tracker. Top filmstrip – calibration – the user marks a plane by moving the hand along the edges of the board. Middle filmstrip – the user places a DVD box on the surface, prompting the system to show facts about the DVD box and a 3D object related to it, which can be manipulated with the hand. For the first DVD, the user can interact with a 3D model of a starship. For the second DVD, the user can interact with a model of planet Mars. Bottom filmstrip – the user can draw on the 2D surface, transform the 2D drawing into a 3D object (by tapping inside it and lifting the hand) and pick up and move the 3D object.

[13] B. Stenger, A. Thayananthan, P. Torr, R. Cipolla, Hand pose estimation using hierarchical detection, in: Workshop on Human-Computer Interaction, 2004, pp. 102–112.

[14] B. Rosenhahn, T. Brox, J. Weickert, Three-dimensional shape knowledge for joint image segmentation and pose tracking, International Journal of Computer Vision 73 (2007) 243–262.

[15] M. Bray, E. Koller-Meier, P. Muller, L. Van Gool, N. Schraudolph, 3D hand tracking by rapid stochastic gradient descent using a skinning model, in: European Conference on Visual Media Production, 2004, pp. 59–68.

[16] M. Bray, E. Koller-Meier, L. Van Gool, Smart particle filtering for 3D hand tracking, in: IEEE International Conference on Automatic Face and Gesture Recognition, 2004, pp. 675–680.

[17] L. Bretzner, I. L. tev tev tev tev, T. Lindeberg, Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering, in: IEEE International Conference on Automatic Face and Gesture Recognition, 2002, pp. 405–410.

[18] S. Lu, D. Metaxas, D. Samaras, Using multiple cues for hand tracking and model refinement, in: International Conference on Pattern Recognition, 2003, pp. 443–450.

[19] Q. Delamarre, O. Faugeras, Finding pose of hand in video images: A stereo-based approach, in: IEEE International Conference on Automatic Face and Gesture Recognition, 1998, pp. 585–590.

[20] J. M. Rehg, Visual analysis of high dof articulated objects with application to hand tracking, Ph.D. thesis (1995).

[21] J. Rehg, T. Kanade, Digiteyes: Vision-based hand tracking for human-computer interaction, in: Workshop on Motion of Non-Rigid and Articulated Bodies, 1994, pp. 16–22.

[22] H. Sidenbladh, M. J. Black, L. Sigal, Implicit probabilistic models of human motion for synthesis and tracking, in: European Conference on Computer Vision, 2002, pp. 784–800.

[23] V. Athitsos, S. Sclaroff, Estimating 3D hand pose from a cluttered image, in: IEEE Conference on Computer Vision and Pattern Recognition, 2003, pp. 432–439.

[24] M. E. Tipping, Sparse bayesian learning and the relevance vector machine, Journal of Machine Learning Research 1 (2001) 211–244.

[25] A. Agarwal, B. Triggs, Recovering 3D human pose from monocular images, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006) 44–58.

[26] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 509–522.

[27] GmbH UZR, iModeller 3D Professional (2009).
URL `http://www.imodeller.com/`

[28] P. R. Giaccone, G. A. Jones, Segmentation of global motion using temporal probabilistic classification, in: British Machine Vision Conference, 1998.

[29] S. Farahani, ZigBee Wireless Networks and Transceivers, 2008.

[30] Freescale, Tilt sensing using linear accelerometers, AN3461.

[31] J. D. Hol, T. B. Schön, F. Gustafsson, Modeling and calibration of inertial and vision sensors, International Journal of Robotics Research 29 (2010) 231–244.

[32] P. Lang, A. Pinz, Calibration of hybrid vision / inertial tracking systems, in: Workshop on Integration of Vision and Inertial Systems, 2005.

[33] C. Bibby, I. Reid, Robust real-time visual tracking using pixel-wise posteriors, in: European Conference on Computer Vision, 2008, pp. 831–844.

[34] C.Li, C.Xu, C.Gui, M.D.Fox, Level set evolution without re-initialization: A new variational formulation, in: Computer Vision and Pattern Recognition, 2005, pp. 430–436.

[35] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision, IEEE Transactions on Pattern Analysis and Machine Intelligence (2004) 1124–1137.

[36] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection (2005) 886–893.

[37] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2004) 91–110.

[38] K. Crammer, Y. Singer, On the algorithmic implementation of multi-class svms, Journal of Machine Learning Research 2 (2001) 265–292.

[39] V. Prisacariu, I. Reid, fastHOG - a real-time GPU implementation of HOG, Tech. Rep. 2310/09, Department of Engineering Science, Oxford University (2009).

[40] G. Schweighofer, A. Pinz, Robust pose estimation from a planar target, IEEE Transactions Pattern Analysis and Machine Intelligence 28 (2006) 2024–2030.

[41] TinEye, Tineye commercial api http://www.tineye.com (2010).

[42] V. Prisacariu, I. Reid, Shared shape spaces, in: International Conference on Computer Vision, 2011.