Accurate, Robust, and Flexible Real-time Hand Tracking:
Supplementary Material

Toby Sharp† **Cem Keskin**† **Duncan Robertson**† **Jonathan Taylor**† **Jamie Shotton**†
David Kim **Christoph Rhemann** **Ido Leichter** **Alon Vinnikov** **Yichen Wei**
Daniel Freedman **Pushmeet Kohli** **Eyal Krupka** **Andrew Fitzgibbon**⋆ **Shahram Izadi**⋆

Microsoft Research

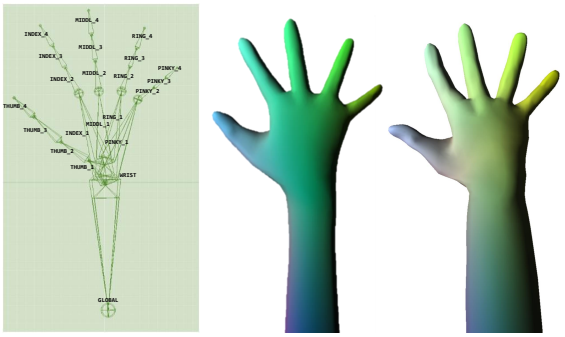

Figure 1: A kinematic skeleton (left) defines (using linear blend skinning) the possible deformations of various hand surface models (center and right).

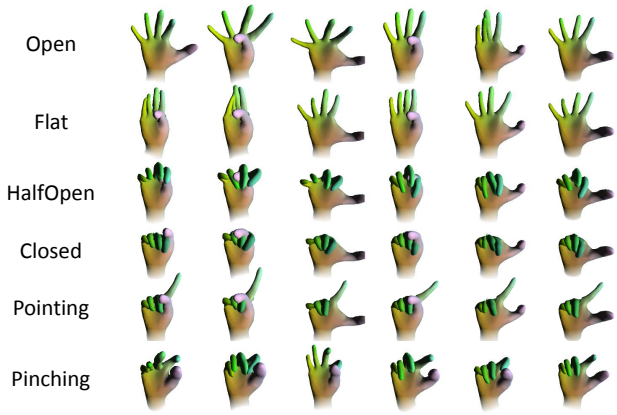

Figure 2: Prototype poses. Each row contains six random draws from one of the six prototype pose distributions.

This document includes supplementary material to accompany the SIGCHI 2015 Paper 'Accurate, Robust, and Flexible Real-time Hand Tracking'. Please also see the accompanying video.

Hand Model

Examples of the kinematic hand model skinned using linear blend skinning are given in Fig. 1.

The hand model is over-parameterized for ease of implementation (e.g. the finger joints can't really *twist* except through external influence). Our optimization strategy has complexity dependent only on the effective dimensionality of the state space, as defined by the range of perturbations applied. Therefore the effective dimensionality of our search space is 28 despite the 91-dimensional pose vector.

Proto-poses

We show in Fig. 2 some samples from the prototype pose distributions, and in Fig. 3 pseudo-code for the Pointing prototype.

Particle Pertubation

To global translation is added a uniform draw in the range ± 0.1m. Global rotation is perturbed by a quaternion drawn uniformly to have a rotation angle below 15 degrees. The wrist, fingers and thumb are perturbed uniformly at random within fixed ranges (see Table 1). For the wrist, the current position is offset by a uniform draw in the given offset range, for the others, the current value is replaced by a uniform draw in the given (min,max) range.

Parameter	Min	Max	Offset range
Wrist Flexion	-90	90	± 45
Wrist Abduction	-30	30	± 30
Thumb Flexion	-60	60	–
Thumb Abduction	-50	50	–
Finger Flexion	-10	50	–
Finger Abduction	-15	30	–

Table 1: Randomization ranges for hand joints.

GPU Implementation

During model fitting, we use Direct3D 11 to compute the golden energy for all particles as follows. We first upload the pose of each particle to a GPU buffer. In a compute shader, we traverse the kinematic tree, composing the transformation matrices that map the bones of each particle from model coordinates to world coordinates. These transforms are next read by a vertex shader that performs linear blend skinning, with a corresponding pixel shader that outputs the RoI depth image. All particles are rendered to a single render target in a tiled layout with a single Draw call via instancing.

We then perform further Draw calls to compute the golden energy for each particle using pixel shaders in a map-reduce style. An initial pass outputs the per-pixel truncated distance, and subsequent passes are used to sum the values over sub-windows, reducing the pixels in each tile until the result can be efficiently read back to system memory. The computation of energy values for a whole swarm requires just one Dispatch call and five Draw calls. We provide system timings in the main paper, but not that even using full linear blend skinning rather than a more approximate sphere or cylinder model, our

† denotes joint first authorship. ⋆ denotes joint last authorship.

CHI '15, Apr 18 – 23 2015, Seoul, Republic of Korea.

ACM 978-1-4503-3145-6/15/04...$15.00. http://dx.doi.org/10.1145/2702123.2702179.

```
//  Initialize to base pose vector
p = base_pose;

//  Choose open/closed thumb
if rand < 0.5
  flex = rand(-15, 40); // thumb closed
else
  flex = rand(40, 50); // thumb open

p(THUMB_1 + FLEX) += flex/20;
p(THUMB_2 + FLEX) += flex;
p(THUMB_3 + FLEX) += flex/4;
p(THUMB_4 + FLEX) += rand(0, 90);

//  Thumb abduction
p(THUMB_2 + ABDUCT) += rand(-5, 35);

//  Spread fingers
spread = rand(-23, 12)
p([INDEX_1, INDEX_2] + ABDUCT) += (spread + rand(-5, 5)) * -1.55 * [1/20, 1];
p([MIDDL_1, MIDDL_2] + ABDUCT) += (spread + rand(-5, 5)) * -0.75 * [1/20, 1];
p([ RING_1,  RING_2] + ABDUCT) += (spread + rand(-5, 5)) *  0.75 * [1/20, 1];
p([PINKY_1, PINKY_2] + ABDUCT) += (spread + rand(-5, 5)) *  2.20 * [1/20, 1];

//  Bend fingers towards palm: Joint 2
flex_1 = rand(60, 90);
p([INDEX_1, INDEX_2, INDEX_3] + FLEX) += (flex_1 + rand(-10, 10)) * [1/20, 1, 1/5];
p([MIDDL_1, MIDDL_2, MIDDL_3] + FLEX) += (flex_1 + rand(-10, 10)) * [1/20, 1, 1/5];
p([ RING_1,  RING_2,  RING_3] + FLEX) += (flex_1 + rand(-10, 10)) * [1/20, 1, 1/5];
p([PINKY_1, PINKY_2, PINKY_3] + FLEX) += (flex_1 + rand(-10, 10)) * [1/20, 1, 1/5];

//  Bend fingers towards palm: Joint 3
flex_2 = rand(60, 90);
p([INDEX_2, INDEX_3, INDEX_4] + FLEX) += (flex_2 + rand(-10, 10)) * [0.2, 1.0, 0.7];
p([MIDDL_2, MIDDL_3, MIDDL_4] + FLEX) += (flex_2 + rand(-10, 10)) * [0.2, 1.0, 0.7];
p([ RING_2,  RING_3,  RING_4] + FLEX) += (flex_2 + rand(-10, 10)) * [0.2, 1.0, 0.7];
p([PINKY_2, PINKY_3, PINKY_4] + FLEX) += (flex_2 + rand(-10, 10)) * [0.2, 1.0, 0.7];

//  Bend fingers towards palm: Joint 4
flex_3 = rand(60, 90);
p([INDEX_3, INDEX_4] + FLEX) += (flex_2 + rand(-10, 10)) * [0.2, 1.0];
p([MIDDL_3, MIDDL_4] + FLEX) += (flex_2 + rand(-10, 10)) * [0.2, 1.0];
p([ RING_3,  RING_4] + FLEX) += (flex_2 + rand(-10, 10)) * [0.2, 1.0];
p([PINKY_3, PINKY_4] + FLEX) += (flex_2 + rand(-10, 10)) * [0.2, 1.0];

//  Overwrite index 2..3 for pointing
p(INDEX_2 + ABDUCT) = rand(-23, 12);
p(INDEX_2 + FLEX) = rand(-5, 15);
p(INDEX_3 + FLEX) = rand(-5, 5);
p(INDEX_4 + FLEX) = rand(-5, 5);
```

Figure 3: Pseudocode (MATLAB-like syntax) for random sampling procedure for the 'Pointing' pose prototype. Uppercase constants are integer indices into the pose parameter vector.

GPU-based approach is fast enough to allow fully articulated hand tracking at 30Hz.

Datasets

This section provides more detail about the three datasets used in the evaluation.

DEXTER1

The DEXTER1 [1] dataset comprises sequences of a single subject performing 7 types of hand motion, and per-frame associated hand-labeled fingertip and mid-palm positions (up to six positions per frame). While considerable finger motion is present, almost no global pose variation is exhibited. Each frame contains multiple RGB and single depth (Creative Senz3D) images, but we only consider depth for our results. The subject's hand is close to the depth camera, occupying about half the vertical field of view. The subject wears a black sleeve so that the hand is segmented from the forearm and background. The captures are all short, around 15s long.

SYNTHETIC

We add synthetic per-pixel Gaussian noise and a coarse simulation of time-of-flight 'flying pixel' noise at occlusion boundaries.

FINGERPAINT

To capture a sequence, the subject's forearm is covered by a sleeve of uniform white, and each finger is painted with a different distinctive color. Following the depth and color acquisition, each color frame is warped into its corresponding depth frame, and each hand pixel is classified based on its color into one of the seven hand parts (forearm, palm, and five fingers) using a semi-automated tagging tool.

The dataset consists of captures of five subjects (see rows of Fig. 4), with three captures per subject, for a total duration of about 30 minutes at 30fps. The subjects include four adults and a child; both genders are represented. We capture three types of sequence: 'global', 'poses', and 'combined'. For global, the subject rotates and translates the hand while keeping the hand open and fingers spread with relatively little motion. For poses, the subject goes through a sequence of articulated hand motions while minimizing global rotations and translations. For combined, the subject is encouraged to fully articulate, rotate, and translate their hand. The articulated motions consist of various common gestures such as counting, pointing, victory, OK, perfect, etc., as well as letters from American sign language.

Additional Results

In figure Fig. 5 we show both the 'worst case' error on SYNTHETIC in addition to the 'average' error shown in the paper. While the absolute numbers may appear somewhat low in places, bear in mind that this dataset is designed to stress-test the algorithm by presenting all possible global rotations, and very poor initializations, making it extremely challenging. Furthermore, the worst case metric is much more stringent than metrics such as mean absolute error that are typically used for evaluation. Also note that many test images have occluded fingers, and the metric still penalizes these if they are wrong, making the absolute numbers appear lower than they might otherwise. (Our experiments on real data and in the supplementary video suggest that these synthetic numbers are really quite good).

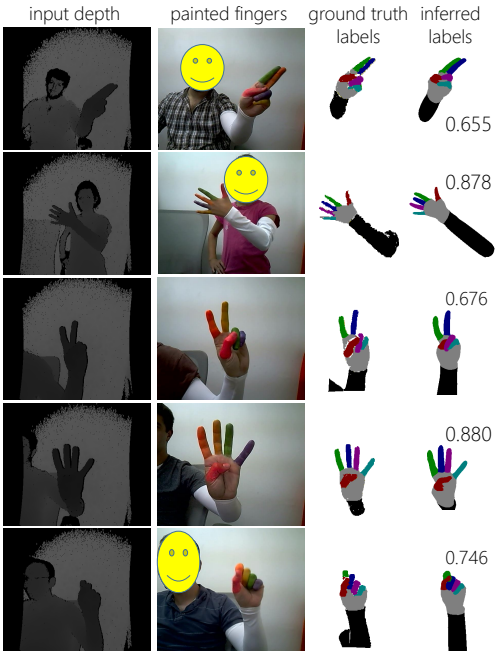

Figure 4: Examples from our FingerPaint dataset.

Further results on FINGERPAINT are shown in Fig. 4. In figure Fig. 6 we show both the 'worst case' classification accuracy metric on FINGERPAINT in addition to the 'average' classification accuracy error shown in the paper.

REFERENCES

1. Sridhar, S., Oulasvirta, A., and Theobalt, C. Interactive markerless articulated hand motion tracking using RGB and depth data. In *Proc. ICCV* (Dec. 2013).

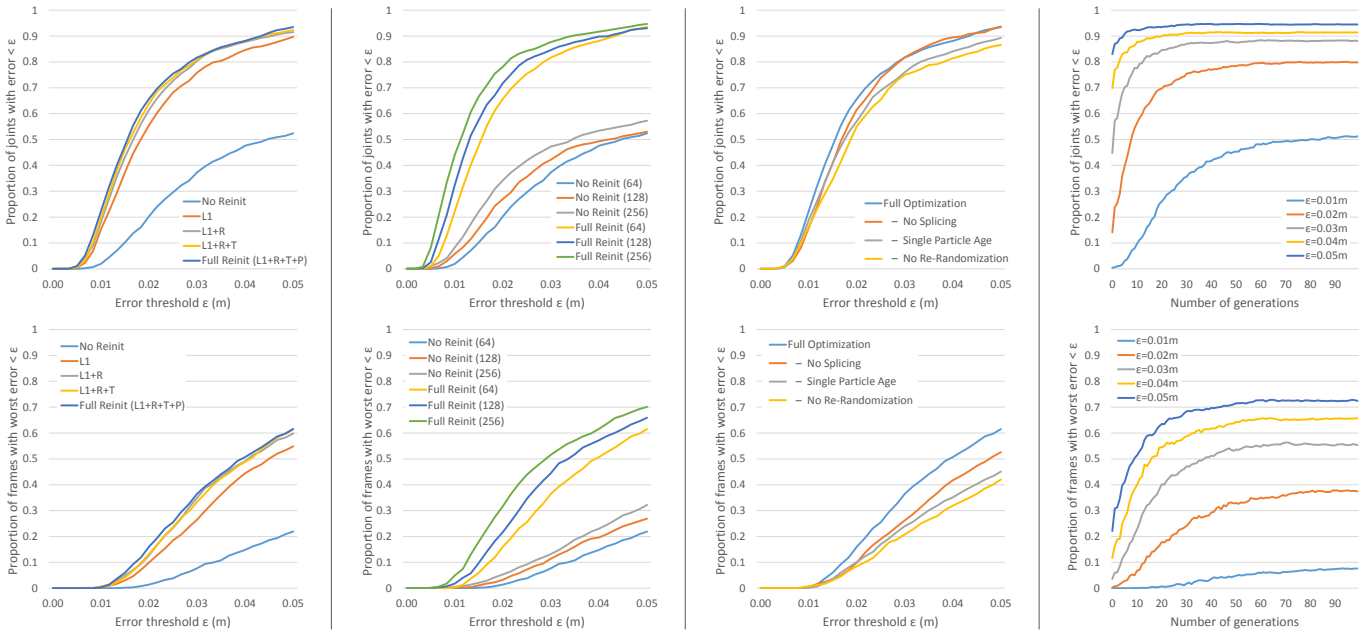

Figure 5: Experiments on our per-frame synthetic test set; see text for description. Top row uses 'average' error metric; bottom row uses 'worst case' error metric (see text). (a) Effect of reinitialization on model fitting accuracy. (L1 = layer one quantized rotation classification, R = layer two rotation regression, T = layer two offset translation regression, P = layer two pose classification). (b) Accuracies achieved using different numbers of particles for both full and no reinitialization. (c) Effect of removing components of the model fitting optimization. (d) Convergence plots for varying levels of error threshold. (Top row reproduced from main paper for comparison purposes).

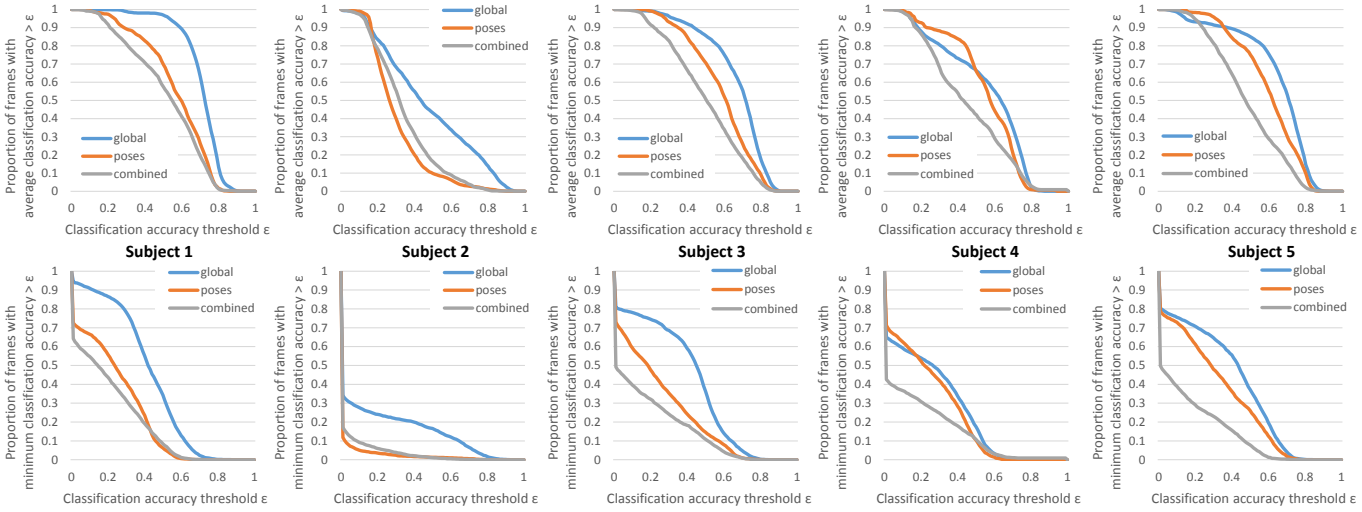

Figure 6: Experiments on the FingerPaint dataset for the five subjects ordered from left to right here corresponding to the examples in Fig. 4 where they are ordered from top to bottom. The top row uses 'average' classification accuracy metric; bottom row uses 'worst case' classification error metric (see text). (Top row reproduced from main paper for comparison purposes).