Real-Time Hand-Tracking with a Color Glove

Robert Y. Wang[1] Jovan Popović[1,2,3]

[1]Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

[2]Advanced Technology Labs, Adobe Systems Incorporated
[3]University of Washington

Figure 1: *We describe a system that can reconstruct the pose of the hand from a single image of the hand wearing a multi-colored glove. We demonstrate our system as a user-input device for desktop virtual reality applications.*

Abstract

Articulated hand-tracking systems have been widely used in virtual reality but are rarely deployed in consumer applications due to their price and complexity. In this paper, we propose an easy-to-use and inexpensive system that facilitates 3-D articulated user-input using the hands. Our approach uses a single camera to track a hand wearing an ordinary cloth glove that is imprinted with a custom pattern. The pattern is designed to simplify the pose estimation problem, allowing us to employ a nearest-neighbor approach to track hands at interactive rates. We describe several proof-of-concept applications enabled by our system that we hope will provide a foundation for new interactions in modeling, animation control and augmented reality.

CR Categories: I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation; H.5.2 [Information interfaces and presentation]: User Interfaces—Input devices and strategies

Keywords: augmented reality, hand tracking, motion capture, user interface

1 Introduction

Recent trends in user-interfaces have brought on a wave of new consumer devices that can capture the motion of our hands. These include multi-touch interfaces such as the iPhone and the Microsoft Surface as well as camera-based systems such as the Sony EyeToy and the Wii Remote. These devices have enabled new interactions in applications as diverse as shopping, gaming, and animation control [Shiratori and Hodgins 2008]. In this paper, we introduce a new user-input device that captures the freeform, unrestricted motion of the hands for desktop virtual reality applications. Our motion-capture system tracks both the individual articulation of the fingers and the 3-D global hand motion. Interactive 3-D hand-tracking has been employed effectively in virtual reality applications including collaborative virtual workspaces, 3-D modeling, and object selection [Agrawala et al. 1997; Grossman et al. 2004; Wesche and Seidel 2001; Sheng et al. 2006; Olwal et al. 2003]. We draw with our hands [Keefe et al. 2005]. We are accustomed to physically-based interactions with our hands [Kry et al. 2008; Wilson et al. 2008]. Hand-tracking can even be combined with multi-touch to provide a combination of 2-D and 3-D input [Benko et al. 2005].

While a large body of research incorporates hand-tracking systems for user-input, their deployment has been limited due to the price and setup cost of these systems. We introduce a consumer hand-tracking technique requiring only inexpensive, commodity components. Our prototype system provides useful hand pose data to applications at interactive rates.

We achieve these results by proposing a novel variation of the hand-tracking problem: we require the user to wear a glove with a color pattern. We contribute a data-driven technique that robustly determines the configuration of a hand wearing such a glove from a single camera. In addition, we employ a Hamming-distance-based acceleration data structure to achieve interactive speeds and use inverse kinematics for added accuracy. Despite the depth ambiguity that is inherent to our monocular setup, we demonstrate our technique on tasks in animation, virtual assembly and gesture recognition.

2 Related work

Several types of optical motion capture systems are capable of tracking the 3-D position and configuration of the hand. The underlying technical difference between our approach and other optical approaches lies in how we correspond parts of the hand to an image. Traditional marker-based systems identify retro-reflective markers

attached to the hand using an array of overlapping cameras. Bare-hand tracking techniques employ multiple-hypothesis inference to overcome the lack of strong correspondences. Our approach uses the color pattern of the glove to infer the approximate pose of the hand, and thus the correspondence of each of its parts.

Marker-based motion-capture. Marker-based motion-capture has been demonstrated in several interactive systems and prototypes. However, these systems require obtrusive retro-reflective markers or LEDs [Park and Yoon 2006] and expensive many-camera setups. They focus on accuracy at the cost of ease of deployment and configuration. While our system cannot provide the same accuracy as high-end optical mocap, our solution is simpler, less expensive, and requires only a single camera.

Bare-hand tracking. Bare-hand tracking continues to be an active area of research. Edge detection and silhouettes are the most common features used to identify the pose of the hand. While these cues are general and robust to different lighting conditions, reasoning from them requires computationally expensive inference algorithms that search the high-dimensional pose space of the hand [Sudderth et al. 2004; Stenger et al. 2006; Dewaele et al. 2006]. Such algorithms are still far from real-time, which precludes their use for control applications.

Several bare-hand tracking systems achieve interactive speeds at the cost of resolution and scope. They may track the approximate position and orientation of the hand and two or three additional degrees of freedom for pose. Successful gesture recognition applications [Schlattman and Klein 2007; Dhawale et al. 2006; Lee et al. 1998] have been demonstrated on these systems. We propose a system that can capture more degrees-of-freedom, enabling direct manipulation tasks and recognition of a richer set of gestures.

Data-driven pose estimation. Our work builds upon the techniques of data-driven pose estimation. Shakhnarovich and colleagues [2003] introduced an upper body pose estimation system that searches a database of synthetic, computer-generated poses. Athitsos and colleagues [2003; 2004] developed fast, approximate nearest-neighbor techniques in the context of hand pose estimation. Ren and colleagues [2005] built a database of silhouette features for controlling swing dancing. Our system imposes a pattern on the glove designed to simplify the database lookup problem. The distinctive pattern unambiguously gives the pose of the hand, improving retrieval accuracy and speed.

Hand-tracking with instrumented gloves. Instrumented gloves systems such as the P5 Data Glove and the Immersion Cyberglove have demonstrated precise capture of 3-D input for real-time control. However, these systems are expensive and unwieldy. They rely on exoskeletons or embed more than a dozen sensors into a glove, which can be restrictive to movement.

Color markers. Previous work in color-based hand-tracking have demonstrated applications in limited domains or for short motion sequences. Theobalt and colleagues track a baseball pitcher's hand motion with color markers placed on the back of a glove using four cameras and a stroboscope [2004]. Dorner uses a glove with color-coded rings to recognize (6 to 10 frame) sequences of the sign language alphabet [1994]. The rings correspond to the joints of each finger. Once the joint positions are identified, the hand pose is obtained with inverse kinematics. We use a data-driven approach to directly search for the hand pose that best matches the image.

Our design strikes a compromise between wearable motion-capture systems and bare-hand approaches. We require the use of an inexpensive cloth glove, but no sensors are embedded in or outside the glove to restrict movement. The custom pattern on the glove facilitates faster and more robust pose estimation. While our approach is not as accurate as traditional optical mocap, it requires only a single camera. The result is an unrestrictive and inexpensive hand-tracking device suitable for desktop virtual reality.

3 Overview

Our work is built around the idea that a distinctive glove simplifies pose inference to the extent that we can largely determine the pose of a hand from a single frame. Moreover, our pose estimation algorithm amounts to looking up the nearest neighbor in a database of hand-poses. We describe the construction of such a database and a means of robustly indexing it in Section 4.

To move from our pose estimation algorithm to a real-time system, we incorporate several improvements. We accelerate database queries by approximating the nearest-neighbor lookup. We use inverse kinematics (IK) to improve accuracy by deriving a set of projection constraints between the nearest-neighbor pose and the query image. We also add a temporal smoothness term to our IK optimization to reduce jitter along the optical axis–the axis of ambiguity in a single camera setup. These improvements to the speed, accuracy and precision of our system are described in Section 5.

We evaluate our resulting system with several test sequences exploring different types of hand motion and three interactive applications of hand-tracking as an input device (Section 6). We demonstrate stacking of a set of physically-simulated rigid bodies into a pyramid. We directly map two fingers to the feet of an animated character to make him walk. We perform American Sign Language finger spelling transcription with 3-D hand-pose recognition. These applications are a proof of concept of the capabilities of our system, and we hope to provide a foundation for new interactions in modeling, animation control and augmented reality.

4 Pose estimation with a colored glove

The core of our approach is to infer pose from an image of the hand wearing a color glove. We design our glove so that this inference task amounts to looking up the image in a database. We generate this database by sampling recordings of natural hand poses and index it by rasterizing images of the glove in these poses. A (noisy) input image from the camera is first transformed into a normalized query. It is then compared to each entry in the database according to a robust distance metric. We conclude with an evaluation of our data-driven pose estimation algorithm showing a steady increase in retrieval accuracy with the size of the database.

4.1 Glove design

Our use of a color glove is inspired by advances in cloth mocap, where dense patterns of colored markers have enabled precise deformation capture [White et al. 2007; Scholz et al. 2005; Guskov et al. 2003]. A variety of color patterns may be appropriate for tracking. Our particular design is motivated by the limitations of consumer-grade cameras, the significant amount of self-occlusion in natural hand motion, and the speed requirements of the inference algorithm.

We describe a glove with twenty patches colored at random with a set of ten distinct colors (See Figure 2). Our color set is limited by our computer vision system, which looks for fully saturated colors to segment the hand from the background. Our camera could reliably distinguish only ten of these colors due to shadows and shading. We chose to use a few large patches rather than numerous small patches because smaller features are less robust to occlusion and require more complex patch identification algorithms for

Figure 2: *Our glove design consisting of large color patches accounts for camera limitations, self-occlusion, and algorithm performance. The length of our glove is 24 cm.*

pose inference [White et al. 2007; Scholz et al. 2005; Guskov et al. 2003].

Our particular glove design is constructed as follows. We select twenty seed triangles on a 3-D hand model that are maximally distant from each other. We then assign the remaining triangles into patches based on their distance to each of these seeds. Each patch is assigned one of the ten colors at random. The jagged boundaries between the patches is an artifact of the low triangle count of our hand model.

Our glove design is sufficiently distinctive that we can reliably infer the pose of the hand from a single frame. This compares favorably to hand-tracking approaches that rely on an accurate pose from the previous frame to constrain the search for the current pose. When these approaches lose track of the hand, they have no means of recovery [de La Gorce et al. 2008]. Our pose estimation (versus tracking) approach effectively "recovers" at each frame. Traditional tracking techniques can still be used to enhance the temporal smoothness of our results.

In bare-hand pose estimation, two very different poses can map to very similar images. This is a difficult challenge that requires slower and more complex inference algorithms to address. With a gloved hand, very different poses always map to very different images (See Figure 3). This allows us to use a simple image lookup approach.

Figure 3: *The palm down and palm up poses map to similar images for a bare hand. These same poses map to very different images for a gloved hand.*

We construct a database of hand poses Λ consisting of a large set of hand configurations $\{\mathbf{q}^i\}$, indexing each entry by a tiny (40×40) rasterized image of the pose $\mathbf{r}^i$ (See Figure 4). Given a normalized query image from the camera, pose estimation amounts to searching a database of tiny images [Torralba et al. 2007]. The pose corresponding to the nearest neighbor is likely to be close to the actual pose of the hand (See Figure 5). To complete this process, we describe a means of constructing a database, normalizing an image from the camera to query our database, and judging distance between two tiny images.

4.2 Database sampling

Ideally, we would like a small database that uniformly samples all natural hand configurations. An overcomplete database consisting of many redundant samples would be inefficient. Alternatively, a database that does not cover all natural hand configurations would result in poor retrieval accuracy. Our approach uses low-dispersion sampling to select a sparse database of samples from a dense collection of natural hand poses.

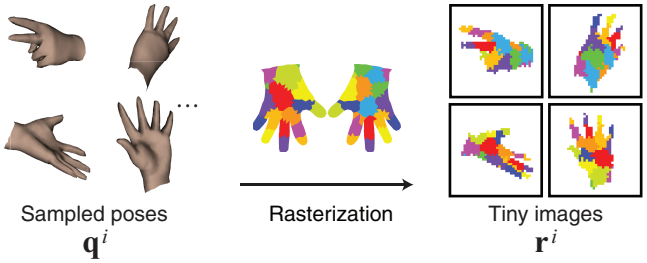

Figure 4: *Pose database. We sample a large set of hand poses which are indexed by their rasterized tiny images.*

We collected a set of 18,000 finger configurations using a Cyberglove II motion capture system. These configurations span the sign language alphabet, common hand gestures, and random jiggling of the fingers. We define a distance metric between two configurations using the root mean square (RMS) error between the vertex positions of the corresponding skinned hand models.

Given a distance metric $d(\mathbf{q}^i, \mathbf{q}^j)$, we can use low-dispersion sampling to draw a uniform set of samples Λ from our overcomplete collection of finger configurations Ω. The dispersion of a set of samples is defined to be the largest empty sphere that can be packed into the range space (of natural hand poses) after the samples have been chosen. We use an iterative and greedy sampling algorithm to efficiently minimize dispersion at each iteration. That is, given samples Λ_ℓ at iteration ℓ, the next sample $i_{\ell+1}$ is selected to be furthest from any of the previous samples.

$$i_{\ell+1} = \underset{i \in \Omega}{\operatorname{argmax}} \min_{j \in \Lambda_\ell} d(\mathbf{q}^i, \mathbf{q}^j)$$

The selected configurations are rendered at various 3-D orientations using a (synthetic) hand model at a fixed position from the (virtual) camera. The rendered images are cropped and scaled, forming our database of tiny images. The result is a database that efficiently covers the space of natural hand poses.

4.3 Image normalization

Figure 6: *We denoise the input image before using a mixture-of-Gaussians color classifier to segment the hand. Our normalization step consists of cropping and rescaling the hand region into a* 40×40 *tiny image.*

To query the database, we convert the camera input image into a tiny image (See Figure 6). First we smooth sensor noise and texture from the image using a bilateral filter. Next, we classify each pixel either as background or as one of the ten glove colors using Gaussian mixture models trained from a set of hand-labeled images. We train one three-component Gaussian mixture model per glove color, operating in the Chong color space [Chong et al. 2008].

After color classification, we are left with an image with glove-pixels and non-glove-pixels. In practice, we use mean-shift with a

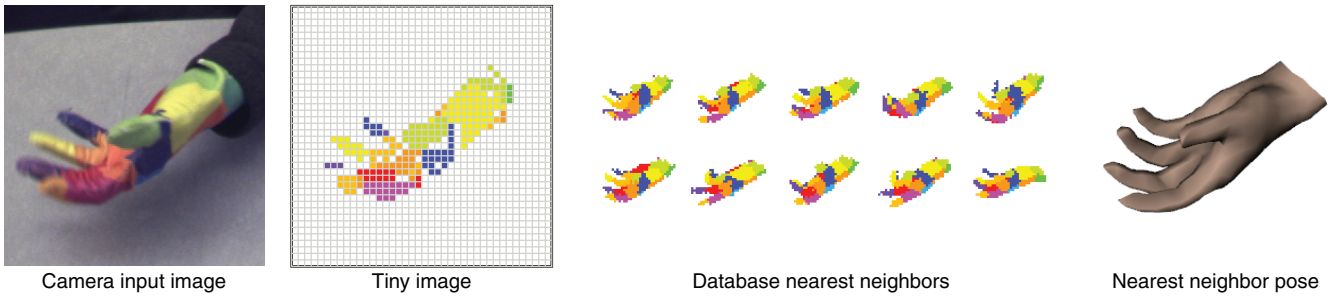

Figure 5: *Our pose estimation process. The camera input image is transformed into a normalized tiny image. We use the tiny image as the query for a nearest neighbor search of our pose database. The pose corresponding to the nearest database match is retrieved.*

uniform kernel of variable-bandwidth to crop the glove region. We start at the center of the image with a bandwidth that spans the entire image. After each iteration of mean-shift, we set the bandwidth for the next iteration based on a multiple of the standard deviation of the glove pixels within the current bandwidth. We iterate until convergence, using the final mean and bandwidth to crop the glove region.

4.4 Tiny image distance

To look up the nearest neighbor, we define a distance metric between two tiny images. We chose a Hausdorff-like distance. For each non-background pixel in one image, we penalize the distance to the closest pixel of the same color in the other image (See Figure 7):

$$\tilde{d}(\mathbf{r}^i, \mathbf{r}^j) = \sqrt{\frac{1}{|\mathcal{C}_i|} \sum_{(x,y)\in\mathcal{C}_i} \min_{(u,v)\in\mathcal{S}_{xy}} (u-x)^2 + (v-y)^2}$$

$$\text{where } \mathcal{S}_{xy} = \left\{ (u,v) | \mathbf{r}^i_{x,y} = \mathbf{r}^j_{u,v} \right\}$$

$$\mathcal{C}_i = \left\{ (x,y) | \mathbf{r}^i_{x,y} \neq \text{background} \right\}$$

$$d(\mathbf{r}^i, \mathbf{r}^j) = \tilde{d}(\mathbf{r}^i, \mathbf{r}^j) + \tilde{d}(\mathbf{r}^j, \mathbf{r}^i)$$

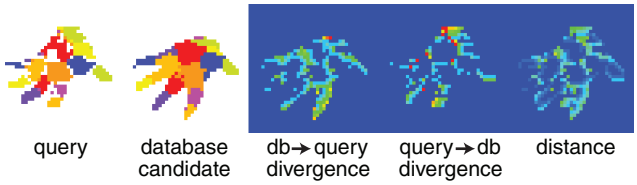

Figure 7: *Hausdorff-like image distance. A database image and a query image are compared by computing the divergence from the database to the query and from the query to the database. We then take the average of the two divergences to generate a symmetric distance.*

We found that our Hausdorff-like image distance metric was more robust to alignment problems or minor distortions of the image than the L_2 distance. Our distance is also more robust to color misclassification than a Hausdorff distance that takes the maximum error across all pixels rather than an average.

The nearest neighbor tiny image can provide an approximate pose of the hand, but cannot account for global position (e.g. distance to the camera) since each query is resized to a tiny image. To address this, we associate 2-D projection constraints with each tiny image for the centroids of every color patch. Thus we can obtain the global hand position by transforming these projection constraints into the coordinate space of the original query image and using inverse kinematics.

Given the database construction and lookup algorithms described above, we can quantitatively evaluate the effect of database size on the accuracy of retrieval. For each database size, we measure the average performance of five hundred test poses sampled randomly from our collection of recorded natural hand configurations. We observe the distance to the nearest neighbor in the database according to the pose distance metric and the image distance metric (See Figure 8).

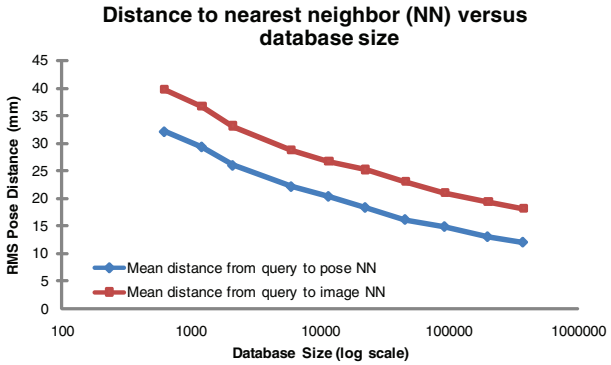

Figure 8: *Database coverage evaluation (log scale). As the database size increases, pose estimation becomes more accurate.*

The consistent decrease of the pose nearest-neighbor distance with database size validates the efficiency and coverage of our database sampling. The consistent decrease of the image nearest-neighbor distance validates our tiny images distance metric and indicates that retrieval becomes more accurate with database size.

4.5 Blending nearest neighbors

In addition to selecting the nearest neighbor in our database, we can also use a blend of the k-nearest-neighbors. This provides a smoother and more accurate result. We blend a neighborhood $\mathcal{N}$ of the ten nearest tiny images with a Gaussian radial basis kernel,

$$\mathbf{q}_\mathrm{p}(\mathbf{r}) = \frac{\sum_{i\in\mathcal{N}} \mathbf{q}^i \exp\left(-d(\mathbf{r}^i, \mathbf{r})^2/\sigma^2\right)}{\sum_{i\in\mathcal{N}} \exp\left(-d(\mathbf{r}^i, \mathbf{r})^2/\sigma^2\right)} \quad (1)$$

where σ is chosen to be the average distance to the neighbors.

5 Real-time hand-tracking system

In this section, we describe our experimental setup and propose several enhancements to our basic pose estimation algorithm for a real-time system. Querying a database of images is well-studied in the computational photography and computer vision literature, and we adapt an acceleration data structure for faster database search to our

pose estimation task. Given an approximate pose from the database, we use gradient descent (via inverse kinematics) to greedily converge to an even more accurate pose. Lastly, we add a smoothness prior to the motion of our hands to reduce jitter. Together, these components improve the speed, accuracy and precision of our system for user-interface applications.

5.1 Experimental setup

We use a single Point Grey Research Dragonfly camera with a 4 mm lens that captures 640×480 video at 30 Hz. The camera is supported by a small tripod placed on a desk and provides an effective capture volume of approximately $60 \times 50 \times 30$ cm (See Figure 1).

We use the Camera Calibration Toolbox for Matlab[1] to geometrically calibrate the camera with respect to the desk to establish a ground plane for our applications. Color calibration is performed by capturing a set of eight images and hand-labeling each of the color regions of the glove. While our current prototype requires manual camera calibration, we intend to provide an instantaneous and automatic calibration tool in the future [White and Forsyth 2005].

We use a 26 degree of freedom (DOF) 3-D hand model: six DOFs for the global transformation and four DOFs per finger. We calibrate the global scale of the hand model to match the length of the user's hand. While this approach has been sufficient on the five subjects who have used our system, calibrating the precise shape and size of the hand would improve the accuracy of our results.

5.2 Faster nearest-neighbor search

To achieve satisfactory accuracy, we use a database size of 100,000 entries. However, querying a database of this size is computationally expensive. While our simple Hausdorff-like image distance is robust, it isn't fast enough to be evaluated millions of times per second. Instead, we follow Torralba and colleagues, compressing each database entry into a short (e.g. 192-bit) binary code for faster retrieval [Torralba et al. 2008; Athitsos et al. 2004]. The codes are designed so that nearby poses are encoded as bit strings with a small Hamming distance. Since these codes are short and the Hamming distance is fast, we can significantly accelerate database search with this approach.

For our database of images $\{\mathbf{r}^i\}$ and distance metric $d(\mathbf{r}^i, \mathbf{r}^j)$, we seek a bitvector representation for each image $\mathbf{b}^i = [h_1(\mathbf{r}^i)h_2(\mathbf{r}^i)\dots h_B(\mathbf{r}^i)]$ such that the hamming distance $d_H(\mathbf{r}^i, \mathbf{r}^j) = \sum_{n=1}^{B} |h_n(\mathbf{r}^i) - h_n(\mathbf{r}^j)|$ preserves the neighborhood relationships of our original distance metric. Our task is to learn the functions h_n. Once the functions h_n, $n = 1 \dots B$, have been selected, we can encode any query as a bitvector $\mathbf{b}$ and use the faster Hamming distance to determine its nearest neighbors in the database.

In the learning phase, we construct a training set composed of example pairs $(\mathbf{r}^i, \mathbf{r}^j)$. Positive examples are pairs that are nearest-neighbors. Negative examples are pairs that are not. We follow the *similarity sensitive coding* [Shakhnarovich et al. 2003] formulation of Torralba and colleagues [2008], defining h_n to be of the form $h_n(\mathbf{r}^i) = e_n^{\mathrm{T}}\mathbf{vec}(D(\mathbf{r}^i)) > T_n$, where e_n and T_n are learned parameters and $D(\mathbf{r}^i)$ is the feature matrix representing the tiny image $\mathbf{r}^i$. We use the GentleBoost algorithm to select the parameters of h_n: e_k is a unit vector, so that $e_k^{\mathrm{T}}x$ selects the kth component of a feature vector x; and T_n is a scalar threshold. Our choice of features resembles the Hausdorff-like image distance metric that we seek to approximate. Each component of the feature matrix $D(\mathbf{r}) \in \mathbb{R}^{40\times40\times10}$, $D_{xyz}(\mathbf{r})$, expresses the shortest distance to a pixel with color z from the location (x, y):

$$D_{xyz}(\mathbf{r}) = \min_{(u,v)\in\mathcal{S}_z} (u-x)^2 + (v-y)^2$$
$$\text{where } \mathcal{S}_z = \{(u,v)|\mathbf{r}_{x,y} = z\}$$

We sampled 10,000 pairs of training examples from our database and experimented with bitvectors of various lengths. We measured the retrieval accuracy of our codes by computing the rank of the true nearest neighbor according to the learned Hamming distance approximation. We found that the rank decreases quickly with the length of the code (See Figure 9).

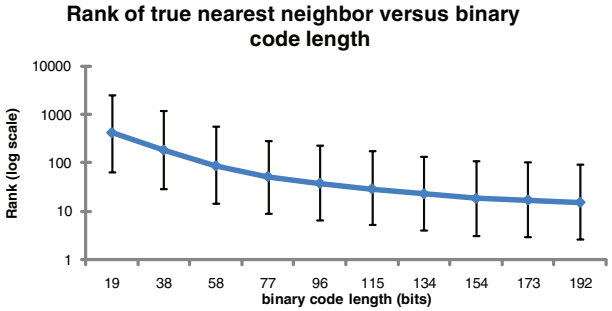

Figure 9: *The rank of the true nearest neighbor (log scale) according to the Hamming distance approximation decreases quickly with longer (more descriptive) codes.*

For our real-time system, we used 192-bit codes, yielding an average rank of 16 for the true nearest neighbor and a standard deviation of 78. To robustly obtain the true nearest neighbor, we re-rank the top 300 approximate nearest-neighbors with the original image distance. Overall, we achieve results approximately 50 times faster than the brute-force nearest-neighbor search described in the previous section.

5.3 Inverse kinematics for accuracy

We improve upon our nearest-neighbor pose estimate by using inverse kinematics to penalize differences between the rasterization of the pose estimate and the original image. However, rasterization is too slow to perform at every iteration of inverse kinematics, and the image Jacobian is difficult to evaluate [de La Gorce et al. 2008]. Instead, we establish correspondences between points on the rasterized pose and the original image. We compute the centroids of each of the visible colored patches in the rasterized pose and identify the closest vertex to each centroid. We then constrain these vertices to project to the centroids of the corresponding patches in the original image (see Figure 10). Note that our correspondences are still valid for poses with self-occlusion because the nearest-neighbor result is usually self-occluded in the same way as the image query.

We use inverse kinematics to minimize the difference between each projected vertex from our hand model and its corresponding centroid point in the original image. We regularize by using the blended nearest neighbor $\mathbf{q}_{\mathrm{p}}$ (See Equation 1) as a prior:

$$\mathbf{q}^* = \underset{\mathbf{q}}{\operatorname{argmin}} \|f(\mathbf{q}) - \mathbf{c}\|_R^2 + \|\mathbf{q} - \mathbf{q}_{\mathrm{p}}\|_Q^2 \qquad (2)$$

where f is a nonlinear function that projects the corresponded points of our hand model into image space; R and Q are the covariances of the constraints $\mathbf{c}$ and the blended nearest neighbor $\mathbf{q}_{\mathrm{p}}$

[1] http://www.vision.caltech.edu/bouguetj/calib_doc

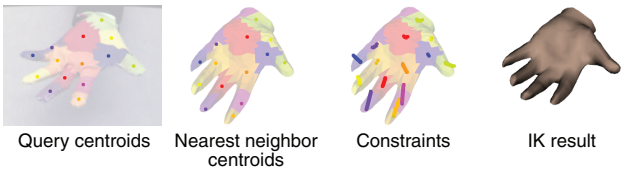

Figure 10: *Correspondences for inverse kinematics. We compute centroids of each patch in the query image and the nearest neighbor pose. We establish correspondences between the two sets of points and use IK to penalize differences between the two.*

respectively; and $\|\cdot\|_X$ is the Mahalanobis distance with respect to covariance X.

We learn the covariance parameters R and Q on a training sequence as follows. We define a local average of the estimated pose $\bar{\mathbf{q}}_\mathrm{p}^i = \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} \mathbf{q}_\mathrm{p}^{i+j}$ over each consecutive five-frame window $\mathcal{S} = \{-2,\ldots,2\}$, and compute covariances of $\mathbf{c}$ and $\mathbf{q}_\mathrm{p}$ about these local averages,

$$R = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{c}^i - f(\bar{\mathbf{q}}_\mathrm{p}^i)\right)\left(\mathbf{c}^i - f(\bar{\mathbf{q}}_\mathrm{p}^i)\right)^T$$

$$Q = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{q}_\mathrm{p}^i - \bar{\mathbf{q}}_\mathrm{p}^i\right)\left(\mathbf{q}_\mathrm{p}^i - \bar{\mathbf{q}}_\mathrm{p}^i\right)^T$$

We use Gauss-Newton iteration to solve Equation 2, with an update of

$$\Delta\mathbf{q} = \left(J^T R^{-1} J + Q^{-1}\right)^{-1}\left(J^T R^{-1}\Delta\mathbf{c} + Q^{-1}\Delta\mathbf{q}_\mathrm{p}\right)$$

where $\Delta\mathbf{q}_\mathrm{p} = \mathbf{q}_\mathrm{p} - \mathbf{q}$, $\Delta\mathbf{c} = \mathbf{c} - f(\mathbf{q})$ and J is the Jacobian matrix $\mathrm{D}f(\mathbf{q})$.

5.4 Temporal smoothness

We can add an additional term to our inverse kinematics optimization to smooth our results temporally:

$$\mathbf{q}^* = \underset{\mathbf{q}}{\operatorname{argmin}} \|f(\mathbf{q}) - \mathbf{c}\|_R^2 + \|\mathbf{q} - \mathbf{q}_\mathrm{p}\|_Q^2 + \|\mathbf{q} - \mathbf{q}_\mathrm{h}\|_P^2$$

where P is the covariance of the pose in the last frame $\mathbf{q}_\mathrm{h}$, and is learned on a training sequence similarly to Q and R. This yields a Gauss-Newton update of the form:

$$\Delta\mathbf{q} = \left(J^T R^{-1} J + Q^{-1} + P^{-1}\right)^{-1}$$
$$\left(J^T R^{-1}\Delta\mathbf{c} + Q^{-1}\Delta\mathbf{q}_\mathrm{p} + P^{-1}\Delta\mathbf{q}_\mathrm{h}\right)$$

where $\Delta\mathbf{q}_\mathrm{h} = \mathbf{q}_\mathrm{h} - \mathbf{q}$.

6 Results

We evaluate the tracking results of our system on several ten second test sequences composed of different types of hand motions. We show that our algorithm can robustly identify challenging hand configurations with fast motion and significant self-occlusion. Lacking ground truth data, we evaluated accuracy by measuring reprojection error and jitter by computing the deviation from the local average $d(\mathbf{q}^i, \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} \mathbf{q}^{i+j})$ of a five-frame window $\mathcal{S} = \{-2,\ldots,2\}$ (See Figure 11).

As expected, inverse kinematics reduces reprojection error by enforcing a set of corresponded projection constraints. However, there is still significant jitter along the optical axis. By imposing the

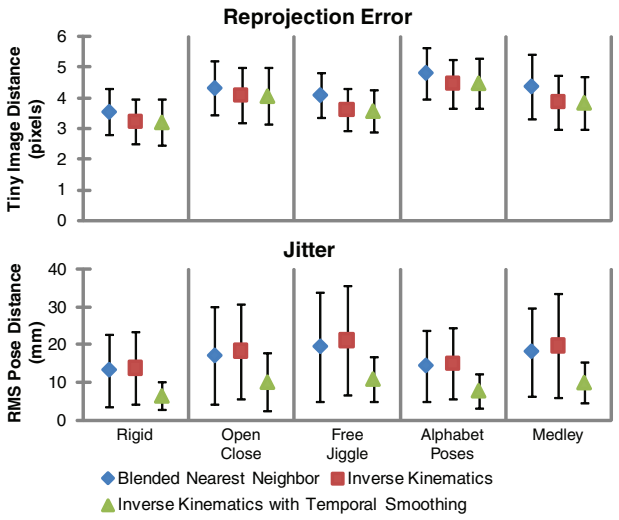

Figure 11: *Reprojection error and jitter on several tracking sequences (lower is better). Inverse kinematics reduces the reprojection error of the scene while temporal filtering reduces the jitter without sacrificing accuracy.*

temporal smoothness term, this jitter is heavily penalized and the tracked motion is much smoother.

While temporal smoothing reduces jitter, it does not eliminate systematic errors along the optical axis. To visualize these errors, we placed a second camera with its optical axis perpendicular to the first camera. We recorded a sequence from both cameras, using the first camera for pose estimation, and the second camera for validation. In our validation video sequence, we commonly observed global translation errors of between five to ten centimeters (See Figure 12). When the hand is distant from the camera, the error can be as high as fifteen centimeters. We attempt to compensate for this systematic error in our applications by providing visual feedback to the user with a virtual hand. While our system was designed to be low cost, the addition of a second camera significantly reduces depth ambiguity and may be a good trade-off for applications that require higher accuracy.

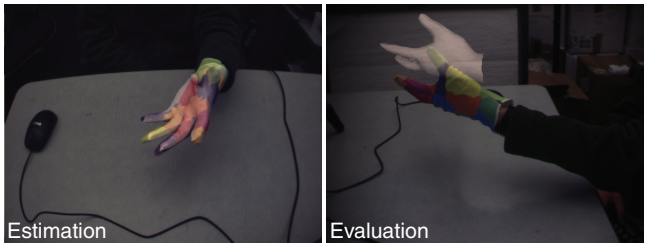

Figure 12: *Even when our estimate closely matches the input frame (left), monocular depth ambiguities remain a problem, as shown from a different camera perspective (right).*

Single-threaded performance of our system is approximately 150 ms per frame split across the following tasks: 50 ms for image processing, 20 ms for nearest-neighbor lookup, and 30 ms for inverse kinematics. We use pipelining on a multi-core processor to increase throughput. All of our interactive demos run at 10 Hz with a latency of 160 ms on a 2.4 GHz Intel Core 2 Quad CPU.

6.1 Applications

We demonstrate our system as a hand-tracking user-interface with three applications. First, we show a simple character animation

system using inverse-kinematics. Next, we use the hand in a physically-based simulation to manipulate and assemble a set of rigid bodies. Finally, we demonstrate pose recognition performance with a sign language alphabet transcription task.

Driving an animated character Our system enables real-time control of a character's walking motion. We map the tips of the index and middle fingers to the feet of the character with kinematic constraints (See Figure 13) [Sturman and Zeltzer 1993]. When the fingers move, the character's legs are driven by inverse kinematics. A more sophisticated approach could learn a hand-to-character mapping given training data [Dontcheva et al. 2003; Lam et al. 2004]. Alternatively, Barnes and colleagues [2008] demonstrate using cutout paper puppets to drive animated characters and facilitate story telling. We hope to enable more intuitive and precise character animation as we reduce the jitter and improve the accuracy of our technique.

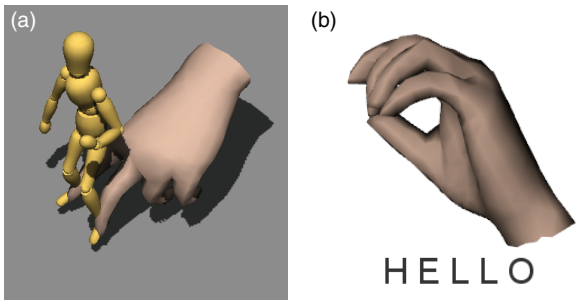

Figure 13: *We demonstrate animation control (a) and a sign language finger spelling application with our hand-tracking interface.*

Manipulating rigid bodies with physics We demonstrate an application where the user can pick up and release building blocks to virtually assemble 3-D structures (See Figure 1). When we detect that a block is within the grasp of the hand, it is connected to the hand with weak, critically damped springs. While this interaction model is unrealistic, it does suffice for the input of basic actions such as picking up, reorienting, stacking and releasing blocks. We expect that data-driven [Li et al. 2007] or physically-based [Pollard and Zordan 2005; Kry and Pai 2006] interaction models would provide a richer experience.

We encountered several user-interface challenges in our rigid body manipulation task. The depth ambiguities caused by our monocular setup (See Figure 12) result in significant distortions in the global translation of the hand. The user must adjust his hand motion to compensate for these distortions, compromising the one-to-one mapping between the actions of the real and virtual hand. Moreover, it was difficult for the user to judge relative 3-D positions in the virtual scene on a 2-D display. These factors made the task of grabbing and stacking boxes a challenging one. We found that rendering a virtual hand to provide feedback to the user was indispensable. Shortening this feedback cycle by lowering latency was important to improving the usability of our system. We also experimented with multiple orthographic views, but found this to complicate the interface. Instead, we settled on using cast shadows to provide depth cues [Kersten et al. 1997].

Sign language finger spelling To demonstrate the pose recognition capabilities of our system, we also implemented an American Sign Language finger spelling application [Starner et al. 1998]. We perform alignment and nearest-neighbor matching on a library of labeled hand poses (one pose per letter) to transcribe signed messages one letter at a time. A letter is registered when the pose associated with it is recognized for the majority of ten frames. This nearest-neighbor approach effectively distinguishes the letters of the alphabet, with the exception of letters that require motion (J, Z) or precise estimation of the thumb (E, M, N, S, T). Our results can be improved with context-sensitive pose recognition and a mechanism for error correction.

7 Conclusions

We have introduced a hand-tracking user-input device composed of a single camera and a cloth glove. We demonstrated this device for several canonical 3-D manipulation and pose recognition tasks. Specifically, we have shown that our technique facilitates useful input for several types of interactive applications.

There are many possible extensions to our system. We can support more cameras for more accuracy or bimanual input as long as the hands do not occlude each other or interlock. Our system can be combined with multi-touch input devices to facilitate precise 2-D touch input. We can design props such as a clicker or trigger to ease selection tasks. We can refine our camera calibration process to support instantaneous deployment in a wider range of settings.

We can envision applications in computer animation and 3-D modeling, new desktop user-interfaces and more intuitive computer games. We can leverage existing design methods for hand interactions [Sturman and Zeltzer 1993] to apply hand-tracking to applications such as virtual surgery to virtual assembly. The goal of this paper is primarily to deliver a robust and low-cost user-input device: we intend to provide a software development kit and inexpensive gloves so that other researchers may develop imaginative applications of their own.

Acknowledgments We thank Sylvain Paris, Frédo Durand, and Randall Davis for their regular feedback; and Anthony Liu for his help with the real-time implementation. This work was supported by grants from the Foxconn Technology Group and software donations from Autodesk and Adobe Systems.

References

AGRAWALA, M., BEERS, A. C., FRÖHLICH, B., HANRAHAN, P. M., MCDOWALL, I., AND BOLAS, M. 1997. The two-user responsive workbench: Support for collaboration through independent views of a shared space. In *Proceedings of SIGGRAPH 97*, 327–332.

ATHITSOS, V., AND SCLAROFF, S. 2003. Estimating 3D hand pose from a cluttered image. In *Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 432–439.

ATHITSOS, V., ALON, J., SCLAROFF, S., AND KOLLIOS, G. 2004. BoostMap: A method for efficient approximate similarity rankings. In *Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 268–275.

BARNES, C., JACOBS, D. E., SANDERS, J., GOLDMAN, D. B., RUSINKIEWICZ, S., FINKELSTEIN, A., AND AGRAWALA, M. 2008. Video puppetry: a performative interface for cutout animation. *ACM Transactions on Graphics 27*, 5, 1–9.

BENKO, H., ISHAK, E., AND FEINER, S. 2005. Cross-Dimensional Gestural Interaction Techniques for Hybrid Immersive Environments. In *IEEE Virtual Reality Conference*, 209–216.

CHONG, H. Y., GORTLER, S. J., AND ZICKLER, T. 2008. A perception-based color space for illumination-invariant image processing. *ACM Transactions on Graphics 27*, 3, 1–7.

DE LA GORCE, M., PARAGIOS, N., AND FLEET, D. 2008. Model-Based Hand Tracking with Texture, Shading and Self-

occlusions. In *Computer Vision and Pattern Recognition (CVPR)*, 1–8.

DEWAELE, G., DEVERNAY, F., HORAUD, R. P., AND FORBES, F. 2006. The alignment between 3-d data and articulated shapes with bending surfaces. In *European Conference on Computer Vision (ECCV)*, 578–591.

DHAWALE, P., MASOODIAN, M., AND ROGERS, B. 2006. Bare-hand 3D gesture input to interactive systems. In *New Zealand Chapter's International Conference on Computer-Human Interaction: Design Centered HCI (CHINZ)*, 25–32.

DONTCHEVA, M., YNGVE, G., AND POPOVIĆ, Z. 2003. Layered acting for character animation. *ACM Transactions on Graphics 22*, 3, 409–416.

DORNER, B. 1994. *Chasing the Colour Glove: Visual Hand Tracking*. Master's thesis, Simon Fraser University.

GROSSMAN, T., WIGDOR, D., AND BALAKRISHNAN, R. 2004. Multi-finger gestural interaction with 3d volumetric displays. In *User Interface Software and Technology (UIST)*, 61–70.

GUSKOV, I., KLIBANOV, S., AND BRYANT, B. 2003. Trackable surfaces. In *Symposium on Computer Animation (SCA)*, 251–257.

KEEFE, D. F., KARELITZ, D. B., VOTE, E. L., AND LAIDLAW, D. H. 2005. Artistic collaboration in designing vr visualizations. *IEEE Computer Graphics and Applications 25*, 2, 18–23.

KERSTEN, D., MAMASSIAN, P., AND KNILL, D. 1997. Moving cast shadows induce apparent motion in depth. *Perception 26*, 2, 171–192.

KRY, P., AND PAI, D. 2006. Interaction capture and synthesis. *ACM Transactions on Graphics 25*, 3, 872–880.

KRY, P., PIHUIT, A., BERNHARDT, A., AND CANI, M. 2008. HandNavigator: hands-on interaction for desktop virtual reality. In *Virtual Reality Software and Technology (VRST)*, 53–60.

LAM, W., ZOU, F., AND KOMURA, T. 2004. Motion editing with data glove. In *International Conference on Advances in Computer Entertainment Technology*, 337–342.

LEE, C.-S., GHYME, S.-W., PARK, C.-J., AND WOHN, K. 1998. The control of avatar motion using hand gesture. In *Virtual Reality Software and Technology (VRST)*, 59–65.

LI, Y., FU, J. L., AND POLLARD, N. S. 2007. Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Transactions Visualization and Computer Graphics 13*, 4, 732–747.

OLWAL, A., BENKO, H., AND FEINER, S. 2003. Senseshapes: Using statistical geometry for object selection in a multimodal augmented reality system. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 300.

PARK, J., AND YOON, Y. 2006. LED-glove based interactions in multi-modal displays for teleconferencing. In *International Conference on Artificial Reality and Telexistence–Workshops (ICAT)*, 395–399.

POLLARD, N., AND ZORDAN, V. 2005. Physically based grasping control from example. In *Symposium on Computer Animation (SCA)*, 311–318.

REN, L., SHAKHNAROVICH, G., HODGINS, J., PFISTER, H., AND VIOLA, P. 2005. Learning silhouette features for control of human motion. *ACM Transactions on Graphics 24*, 4, 1303–1331.

SCHLATTMAN, M., AND KLEIN, R. 2007. Simultaneous 4 gestures 6 dof real-time two-hand tracking without any markers. In *Virtual Reality Software and Technology (VRST)*, 39–42.

SCHOLZ, V., STICH, T., KECKEISEN, M., WACKER, M., AND MAGNOR, M. A. 2005. Garment motion capture using color-coded patterns. *Computer Graphics Forum 24*, 3, 439–447.

SHAKHNAROVICH, G., VIOLA, P., AND DARRELL, T. 2003. Fast pose estimation with parameter-sensitive hashing. In *International Conference on Computer Vision (ICCV)*, 750–757.

SHENG, J., BALAKRISHNAN, R., AND SINGH, K. 2006. An interface for virtual 3d sculpting via physical proxy. In *Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (GRAPHITE)*, 213–220.

SHIRATORI, T., AND HODGINS, J. K. 2008. Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics 27*, 5, 1–9.

STARNER, T., WEAVER, J., AND PENTLAND, A. 1998. Real-time American sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1371–1375.

STENGER, B., THAYANANTHAN, A., TORR, P., AND CIPOLLA, R. 2006. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions Pattern Analysis and Machine Intelligence 28*, 9, 1372–1384.

STURMAN, D. J., AND ZELTZER, D. 1993. A design method for "whole-hand" human-computer interaction. *ACM Transactions on Information Systems 11*, 3, 219–238.

SUDDERTH, E. B., MANDEL, M. I., FREEMAN, T., AND WILLSKY, S. 2004. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In *Neural Information Processing Systems (NIPS)*, 1369–1376.

THEOBALT, C., ALBRECHT, I., HABER, J., MAGNOR, M., AND SEIDEL, H.-P. 2004. Pitching a baseball – tracking high-speed motion with multi-exposure images. *ACM Transactions on Graphics 23*, 3, 540–547.

TORRALBA, A., FERGUS, R., AND FREEMAN, W. T. 2007. Tiny images. Tech. Rep. MIT-CSAIL-TR-2007-024, Computer Science and Artificial Intelligence Lab, MIT.

TORRALBA, A., FERGUS, R., AND WEISS., Y. 2008. Small codes and large databases for recognition. In *Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 1–8.

WESCHE, G., AND SEIDEL, H.-P. 2001. Freedrawer: a free-form sketching system on the responsive workbench. In *Virtual Reality Software and Technology (VRST)*, 167–174.

WHITE, R., AND FORSYTH, D. 2005. Deforming objects provide better camera calibration. Tech. Rep. UCB/EECS-2005-3, EECS Department, University of California, Berkeley.

WHITE, R., CRANE, K., AND FORSYTH, D. A. 2007. Capturing and animating occluded cloth. *ACM Transactions on Graphics 26*, 3, 34:1–34:8.

WILSON, A., IZADI, S., HILLIGES, O., GARCIA-MENDOZA, A., AND KIRK, D. 2008. Bringing physics to the surface. In *User Interface Software and Technology (UIST)*, 67–76.