Video-based Hand Manipulation Capture Through Composite Motion Control

Yangang Wang* Jianyuan Min† Jianjie Zhang† Yebin Liu*
Feng Xu* Qionghai Dai* Jinxiang Chai†
*Tsinghua University †Texas A&M University

Figure 1: *Modeling high-fidelity dexterous manipulation data from videos: (top) observed image data; (bottom) reconstructed motion.*

Abstract

This paper describes a new method for acquiring physically realistic hand manipulation data from multiple video streams. The key idea of our approach is to introduce a composite motion control to simultaneously model hand articulation, object movement, and subtle interaction between the hand and object. We formulate video-based hand manipulation capture in an optimization framework by maximizing the consistency between the simulated motion and the observed image data. We search an optimal motion control that drives the simulation to best match the observed image data. We demonstrate the effectiveness of our approach by capturing a wide range of high-fidelity dexterous manipulation data. We show the power of our recovered motion controllers by adapting the captured motion data to new objects with different properties. The system achieves superior performance against alternative methods such as marker-based motion capture and kinematic hand motion tracking.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—animation

Keywords: Hand grasping and manipulation, motion capture, motion control, hand tracking, physics-based simulation

Links: DL PDF

ACM Reference Format
Wang, Y., Min, J., Zhang, J., Liu, Y., Xu, F., Dai, Q., Chai, J. 2013. Video-based Hand Manipulation Capture Through Composite Motion Control. ACM Trans. Graph. 32, 4, Article 43 (July 2013), 13 pages.
DOI = 10.1145/2461912.2462000 http://doi.acm.org/10.1145/2461912.2462000.

DOI: http://doi.acm.org/10.1145/2461912.2462000

1 Introduction

Creating realistic animations of the human hand performing a dexterous task, such as "grasping the cup handle and spinning a magic cube with fingers", is one of the grand challenges in computer graphics. Data-driven approaches, where sets of example motions are available for editing, retargeting, interpolation and composition, are promising for manipulation, as there are many commonalities in how the hand manipulates similar objects. However, capturing high-fidelity hand grasping and manipulation data is extremely hard because it requires reconstructing not only delicate hand articulation and object movement but also subtle interaction and contact phenomena between the hand and object.

Decades of research in computer graphics and vision have explored a number of approaches to capturing hand grasping and manipulation data, including marker-based motion capture, glove-based systems, and image-based systems. Despite the efforts, acquiring high-fidelity hand manipulation data remains a challenging task. For example, marker-based motion capture systems (*e.g.* Vicon [2012]) often produce ambiguous solutions because of significant self-occlusions or the occlusions caused by the object. Glove-based systems such as CyberGlove [2012] are free of occlusions but recorded motion data is often noisy, thereby failing to capture delicate hand articulation. In addition, neither approach thus far has demonstrated that they can capture subtle interaction between the hand and object.

Image-based systems offer an appealing alternative to hand manipulation capture because they require no markers, no gloves, or no sensors and thereby do not impede the subject's ability to perform the motion. However, current image-based mocap techniques suffer from three major limitations. First, they are vulnerable to ambiguities caused by significant occlusions and a lack of discernible features on a hand. Second, they often focus on hand articulation alone and completely ignore interaction and constraints between the hand and object. Lastly and most importantly, they do not consider

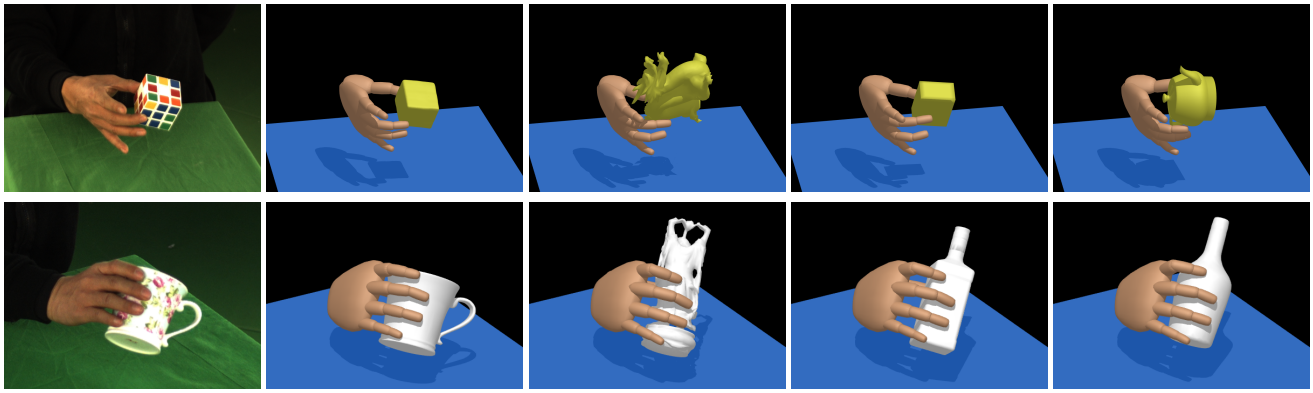

Figure 2: *Retarget captured motion data to new objects with different geometries. From left to right, we show the original image data, the reconstructed manipulation data, and the retargeted motions for grasping and manipulating three different objects.*

the dynamics that cause motion. Captured motion data, therefore, is often noisy and physically implausible, displaying unpleasant visual artifacts such as motion jerkiness, hand-object penetration, and improper interaction between the hand and object.

In this paper, we introduce a new image-based hand manipulation capture method that addresses all three challenges aforementioned. Our key idea is to introduce a composite motion control to simultaneously model hand articulation, object movement, and subtle interaction between the hand and object. Physics-based dynamics models are appealing to video-based manipulation capture because they naturally model subtle interaction between the hand and object. In particular, they can eliminate physically implausible contact phenomena between the hand and object. This is important as observed image data alone is often too ambiguous and noisy to reconstruct subtle interaction and contact phenomena between the hand and object. In addition, they can resolve ambiguity in pose inference by constraining the solution to lie in physically plausible motion space. Another benefit is that we can utilize recovered motion control to conveniently adapt captured motion data to new object with different properties, which is particularly appealing to graphics and animation applications.

We formulate video-based manipulation capture in an optimization framework by maximizing the consistency between the simulated motion and the observed image data. We search an optimal motion control that drives the simulation to best match the observed image data. We demonstrate the effectiveness of our system by modeling a wide range of hand manipulation data from video. In particular, we test our system on grasping and manipulating four different objects, including a cup, a solid ball, a long round stick, and a magic cube. Figure 1 shows that the system can capture dexterous manipulation data involving multiple objects such as "grasping a cup, turning it upside down, placing it on the table, then picking up a ball and putting it inside the cup." We show the power of recovered motion controllers by adapting captured motion data to new objects with different properties (Figure 2). We demonstrate superior performance of our system by comparing against alternative methods such as marker-based motion capture [Vicon Systems 2012] and kinematic motion tracking. Finally, we validate our algorithm by evaluating the importance of different components of the system.

2 Background

Our system combines 2D image data and physics-based motion control to capture high-fidelity hand manipulation data. We therefore focus our discussion on video-based motion capture and physics-based dynamics modeling for hand grasping and manipulation.

One way to capture 3D hand motion from video is model-based kinematic hand tracking (*e.g.*, [Rehg and Kanade 1995; de La Gorce et al. 2011]), which initializes a 3D hand pose at the first frame and sequentially tracks 3D poses by minimizing the inconsistency between the hypothesized poses and the observed image data. However, the approach has many restrictions because it is vulnerable to ambiguities in video (*e.g.*, significant occlusions and a lack of discernible features on the hand and/or objects). In practice, image measurements alone are often not sufficient to capture high-fidelity 3D hand articulation data.

An efficient way to reduce the tracking ambiguity is to utilize kinematic motion priors embedded in prerecorded motion data. Thus far, two different approaches have been taken, including generative approaches [Wu et al. 2001; Zhou and Huang 2003] and discriminative approaches [Athitsos and Sclaroff 2003; Wang and Popović 2009; Romero et al. 2010]. However, data-driven approaches are restricted to modeling motions that are similar to prerecorded motion data. In addition, they are mainly focused on tracking the articulated hand and therefore are not capable for modeling complex contact phenomena between the hand and object.

Recently, Oikonomidis and colleagues [2011] incorporated collision detection into the tracking process and simultaneously tracked the movement of the hand and object based on multiview image sequences. More recently, Ballan and colleagues [2012] proposed to use discriminatively learned salient points on the fingers to estimate the finger-salient point associations simultaneously with the estimation of the hand pose. They also introduced a differential objective function that takes edges, optical flow and collisions into account. These methods, while effective in preventing hand-object penetration, don't consider the underlying dynamics that cause subtle interaction between the hand and object. As a result, their results are often physically incorrect and display noticeable visual artifacts (*e.g.* uncoordinated movement between the hand and object).

Marker-based motion capture techniques have also been used to capture hand motion data (*e.g.* [Hoyet et al. 2012; Zhao et al. 2012]). Notably, Zhao and his colleagues [2012] showed how to combine marker-based motion capture with RGBD image data recorded by a single *Kinect* to capture high-fidelity hand articulation data. However, marker-based approaches have not demonstrated they can capture subtle contact phenomena between the hand and object thus far.

Our method leverages physics-based motion control to simultaneously model hand articulation, object movement, and subtle interaction between the hand and object directly from video. Our work is motivated by recent efforts in modeling physics-based full-body motion models from image data (*e.g.*, [Brubaker and Fleet 2008; Vondrak et al. 2008; Wei and Chai 2010; Vondrak et al. 2012]). Our work is different because we focus on dexterous grasping and manipulation rather than full-body movement. This requires us to model subtle interaction and frequent contact phenomena between the hand and object, a new challenge that has not been addressed by previous work in video-based motion capture. To address the challenge, we introduce a composite motion control that combines PD servos and virtual forces for hand manipulation.

Our work builds on the success of physics-based dynamics models for hand grasping and manipulation [Pollard and Zordan 2005; Kry and Pai 2006; Liu 2008; Liu 2009; Ye and Liu 2012]. Pollard and Zordan [2005] explored a motion controller for physically based grasping that is automatically derived from prerecorded motion data. Kry and Pai [2006] used prerecorded hand motion data and contact force data to extract joint compliances. By adjusting the joint compliances, the prerecorded motion data can be retargeted to new objects with different properties. Liu [2008; 2009] explored a physics-based optimization approach for synthesizing physically-correct hand manipulation animation from both user-specified grasping poses and prescribed object motions. Our method does not require any prerecorded motion data or contact forces. Instead, we take ambiguous 2D image data as an input and compute a composite motion control that drives the simulation to best match the observed image data. Researchers have also explored how to construct anatomically realistic models to simulate unconstrained hand animation [Tsang et al. 2005; Sueda et al. 2008]. These methods approximate hand anatomy by modeling muscles, tendons, and their interdependency. However, it is not clear how these methods can be extended to efficiently model hand-object manipulation.

More recently, Ye and Liu [2012] explored a physics-based optimization approach to transform a sequence of full-body motion with accurate wrist movements and a simultaneously acquired sequence of object motion into physically plausible hand manipulation animations. Their methods, although powerful in modeling subtle contact phenomena between the hand and object, do not necessarily produce finger motions that are consistent with real world observations. In contrast, our approach produces hand manipulation motions that are not only physically realistic but also consistent with real world observations. Our physics-based motion model is also different from theirs because it builds on physically-based simulation via composite motion control rather than physics-based optimization adopted in their system.

3 Problem Statement and Overview

Acquiring high-quality hand manipulation data is difficult because it requires modeling not only hand articulation and object movement but also subtle contact phenomena between the hand and object. Video-based hand manipulation capture is even more challenging because of the ambiguities caused by the lack of discernible features on the hand and significant occlusions caused by frequent contacts between the hand and object. This section provides an overview of our solution to address these challenges.

State space. We describe a 3D hand pose using a set of independent joint coordinates $\mathbf{q}^h \in R^{28}$, including absolute root position and orientation as well as the relative joint angles of individual joints. Figure 3 shows the number of degrees of freedom for each joint. We describe 3D poses of the object using rigid transformations $\mathbf{q}^o \in R^6$.

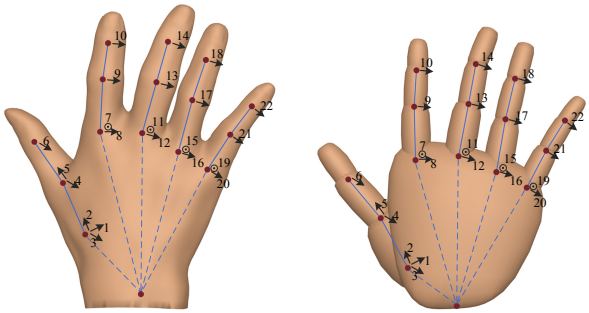

Figure 3: *Our hand pose consists of 28 degrees of freedom, including absolute root position and orientation (6) and the relative joint angles of individual joints (22). (left) skinned mesh model; (right) rigid body approximation for collision detection.*

The complete configuration of the hand and object at frame t can thus be described by a state vector $\mathbf{q}_t = [\mathbf{q}_t^h, \mathbf{q}_t^o]$.

Problem formulation. We build our motion control on PD-servos (Proportional Derivative). The PD controller for each DOF of the hand is parameterized by a target angle ($\bar{\theta}$). At each simulation time step, a torque (τ) for each DOF of the hand is generated by

$$\tau = k_p(\bar{\theta} - \theta) - k_d\dot{\theta} \tag{1}$$

where θ and $\dot{\theta}$ are the current joint angle and angular velocity of the hand. And k_p and k_d are the gain and damping coefficients of the PD controller. Note that the object cannot generate any active joint torques/forces by itself because it is passive and its movement is completely controlled by contact forces (and thus active joint torques of the hand).

Given an initial pose and velocity of the hand and object $\mathbf{s}_0 = [\mathbf{q}_0, \dot{\mathbf{q}}_0]$, we can obtain a sequence of simulated poses of the hand and object $\mathbf{q}_1, ..., \mathbf{q}_T$ by sequentially choosing an appropriate value of the target poses $\bar{\mathbf{q}}_0, ..., \bar{\mathbf{q}}_{T-1}$. This is achieved by applying the motion control to generate appropriate joint torques to advance the physical model through time via forward dynamics simulation.

We formulate our video-based motion capture problem in a nonlinear optimization framework by minimizing the inconsistency between the simulated motion M and the observed image data O:

$$\min_{\mathbf{s}_0, \bar{\mathbf{q}}_0, ..., \bar{\mathbf{q}}_{T-1}} E(M(\mathbf{s}_0, \bar{\mathbf{q}}_0, ..., \bar{\mathbf{q}}_{T-1}), O) \tag{2}$$

where the simulated motion M is dependent on both initial state of the system $\mathbf{s}_0 = [\mathbf{q}_0, \dot{\mathbf{q}}_0]$ and a sequence of the target poses $\bar{\mathbf{q}}_0, ..., \bar{\mathbf{q}}_{T-1}$ required for motion control. The function E measures the inconsistency between the simulated motion M and the observed image data O. Our goal here is therefore to search an optimal motion control that drives the simulation to best match the observed image data.

In the following, we highlight the issues critical for the success of this endeavor and summarize our approach for addressing them.

Image-based motion modeling. First and foremost, how can we define an appropriate function (E) to measure the inconsistency between the simulated motion (M) and the observed image data (O)? The problem is challenging because of ambiguities caused by significant occlusions and the lack of distinctive features on the hand. Texture-less and non-diffuse objects further complicate the problem. Section 4 introduces a cost function to evaluate how well the simulated motion matches the observed image data.

Composite motion control for dexterous manipulation. The suc-

cess of our approach is highly dependent on the ability of our motion control to model dynamics of the hand and object as well as subtle contact phenomena between the two. However, in practice, searching the target poses of PD servos alone is often not sufficient to produce exact contact forces to drive the simulation to precisely match the observed image data. This is because the PD control defined in Equation (1) only considers the target trajectory of the hand but completely ignores the movement of the object. Therefore, the "simulated" movement of the object often fails to match the "observed" movement. In Section 5, we propose a new motion control that combines PD servos and "virtual forces" for dexterous manipulation. In our application, the "virtual forces" are used to model the internal joint torques of the hand required to drive the "simulated" object to precisely match the "observed" image data. We compute the "virtual forces" from the target poses of the object.

Sampling-based control optimization. The last challenge is how to search an optimal motion control to drive the simulation to best match the observed image data. This requires searching the initial state $\mathbf{s}_0 = [\mathbf{q}_0, \dot{\mathbf{q}}_0]$ and a sequence of target poses of the hand and object across the entire sequence $\bar{\mathbf{q}}_0, ..., \bar{\mathbf{q}}_{T-1}$. The problem is particularly challenging because the solution space is high-dimensional. Moreover, the simulation function for dexterous manipulation is discontinuous whenever there is a collision between the hand and object. As a result, even a small change in the target poses could produce a significant change in the simulated motion. Our idea is to first reconstruct kinematic motion data of the hand and object from the observed image data and then search the target poses of the hand and object in the vicinity of the reconstructed kinematic poses to produce new target poses for advancing the simulation (see Section 6). We choose to use sampling techniques [Liu et al. 2010] to search optimal target poses because sampling-based approaches do not demand derivative computation, which is almost impossible to evaluate in our application. In addition, we discuss how to use contact information between the hand and object to significantly improve the efficiency of sampling process.

We describe these components in more detail in the next sections.

4 Image-based Motion Modeling

This section introduces a cost function that measures the inconsistency between the "hypothesized" hand manipulation data and the "observed" image data. The cost function serves two goals. First, it is used to evaluate how well the "simulated" motion matches the "observed" image data defined in Equation (2). Second, it is used to reconstruct kinematic motion of the hand and object from observed image data, which will later be utilized to initialize our sampling-based control optimization process described in Section 6.

In the following, we first describe our preparation step for cost function evaluation (Section 4.1). We then discuss the detail of the cost function (Section 4.2) and explain how it is used to reconstruct kinematic motion data from the observed image data (Section 4.3).

4.1 Data Preparation

In the preparation step, we first construct 3D surface models of the hand and object using a Minolta VIVID 910 laser scanner. We then build a skinned hand mesh model from the scanned mesh model of the hand so that we can deform the mesh model according to pose changes of an underlying articulated skeleton using Skeleton Subspace Deformation (SSD) (Figure 3(left)). For physics-based simulation of the hand, we approximate geometry of each bone segment of the hand mesh model with a small number of geometric primitives (Figure 3(right)), which are required for running collision detection in the open dynamics engine (ODE) library.

Figure 4: *Extracting silhouette maps of the hand: (a) the original image; (b) the extracted silhouette map S_o; (c) the synthesized silhouette map S_r automatically generated by the rendering process.*

4.2 Objective Function

We adopt an *analysis-by-synthesis* strategy to evaluate the inconsistency between the "hypothesized" motion data $\mathbf{q}_t, t = 1, ...T$ and the "observed" image data $\mathbf{I}_t^v$, $t = 1, ..., T$, where v is the camera index. Our system employs six synchronized video cameras to capture dexterous manipulation data.

Given a hypothesized pose $\mathbf{q}$, we apply the corresponding transformation $T_\mathbf{q}$ to each vertex of the surface mesh models to obtain 3D geometric models of the hand and object under the hypothesized pose. Given the calibrated camera parameters [Zhang 1999], we can further project the transformed 3D mesh models onto the image plane and render synthetic images at each viewpoint via view-dependent texture mapping techniques [Debevec et al. 1998]. Corresponding texture images for view-dependent texture mapping are obtained by the initialization step discussed in Section 4.3. We choose to do texture mapping using registered multiview images in the first frame in order to reduce the drifting problem in sequential tracking.

One way to measure the consistency between the "hypothesized" 3D motion data and the "observed" image data is to compute the color differences between the "synthesized" image data and the "observed" image data. However, color information alone is often insufficient for motion reconstruction because the hand, sometimes even the object, lacks distinctive colors. In addition, the surface of the object might not be perfectly diffused. This motivates us to extract both edge and silhouette information from the "synthesized" and "observed" images and include them for measuring the distance between the "hypothesized" motion data and the "observed" image data.

We define the following cost function to measure the distance between the "hypothesized" hand manipulation pose $\mathbf{q}_t$ and the 'observed" image data $\mathbf{I}_t^v, v = 1, ..., 6$:

$$E = w_{silh} \sum_v E^v_{silh} + w_{color} \sum_v E^v_{color} + w_{edge} \sum_v E^v_{edge} \qquad (3)$$

where E^v_{silh}, E^v_{color} and E^v_{edge} measure the silhouette, color and edge discrepancy between the "synthesized" and "observed" images. The weights w_{silh}, w_{color} and w_{edge} control the importance of each term and are set to 1.0, 0.6 and 1.0, respectively.

Silhouette term. This term ensures that silhouette maps of the synthesized images match those extracted from the observed images. A silhouette map is encoded as a binary image whose foreground and background pixels are set to one and zero, respectively. The system automatically extracts silhouette maps of the hand in observed images using a learned probabilistic model for silhouette pixels of the hand. The probabilistic model is constructed from the color histogram of the first frame of multiview images. We

Figure 5: *Rendering the color images of the hand: (a) the original image; (b) the "observed" color image of the hand using silhouette information; (c) the "rendered" color image of the hand using view-dependent texture mapping.*

model the color histogram as Super-Gaussian Mixture Models (SGMMs) [Palmer et al. 2006]. The probability of a pixel x belonging to the hand is thus defined as:

$$H(x) = \sum_j \lambda_j \exp\left(-|x-\mu_j|^{0.8}\right),\ \lambda_j \geq 0,\ \sum_j \lambda_j = 1 \qquad (4)$$

where j is an index of mixture components. The model parameters λ_j and μ_j are automatically estimated by foreground pixels of the hand at the first frame via an EM algorithm.

For each observed image I_t^v, we evaluate the probability values of each pixel belonging to the silhouette map of the hand using the learned SGMMs. We threshold the computed probability values to generate the silhouette map of the hand, denoted as S_o. In contrast, the silhouette maps of each rendered image, denoted as S_r, are automatically generated by the rendering process. Figure 4 shows the observed and synthesized silhouette maps.

The silhouette term for each input image is defined as follows:

$$E_{silh}^v = \frac{\sum(S_r \cap \overline{S}_o)}{\sum S_r + \varepsilon} + \frac{\sum(S_o \cap \overline{S}_r)}{\sum S_o + \varepsilon} \qquad (5)$$

where $\overline{S}_o = 1 - S_o$ and $\overline{S}_r = 1 - S_r$ are the complements of the observed (S_o) and rendered (S_r) silhouette maps, respectively. ε is a small quantity to prevent division by zero. Sums are computed over every pixel of the entire silhouette maps. Intuitively, the silhouette term minimizes the area of non-overlapping regions between the synthesized and observed silhouette maps, therefore maximizing the area of their overlapping regions. In our experiment, we evaluate the silhouette term based on the silhouette maps of the hand alone because, unlike the object, the hand silhouettes can often be robustly extracted from the observed images (see Figure 4).

Color term. This term measures the color inconsistency between the observed and rendered images, denoted as I_t^v and R_t^v respectively. We apply view-dependent texture mapping techniques to render the images R_t^v from each camera viewpoint. Figure 5(b) and (c) show the "observed" and "rendered" color images of the hand, respectively. One common way to measure the differences of the two color images is based on the sum-of-squared pixel differences. This idea, however, does not work well for our application because initial registration errors as well as non-diffuse surfaces of the object often result in a noisy and unstable cost function, particularly when manipulation objects have rich distinctive texture.

To address this concern, we introduce the following cost function to robustly evaluate the inconsistency between the rendered and observed images:

$$E_{color}^v = \sum_i \min\left(\min_{j \in N(i)} \left(w_{i,j}\,|R_t^v(i) - I_t^v(j)|\right), T\right) \qquad (6)$$

Specifically, for each pixel i in the rendered image R_t^v, we compute the color difference between the rendered pixel $R_t^v(i)$ and every observed pixel $I_t^v(j)$ in a small window centered around pixel i, denoted as $j \in N(i)$. We obtain the minimum error within the window and return it as the color difference at pixel i. In our experiment, we choose a 5-by-5 window. We also associate a weight with the error based on the distance between the two pixels. We choose the weight function as $w_{i,j} = e^{-\|i-j\|^2}$ in order to assign a larger weight to the pixel closer to pixel i. In addition, we define a cut-off threshold T for the return error to ensure the cost function is robust to outliers caused by non-diffuse surfaces. The threshold T is set to 0.25.

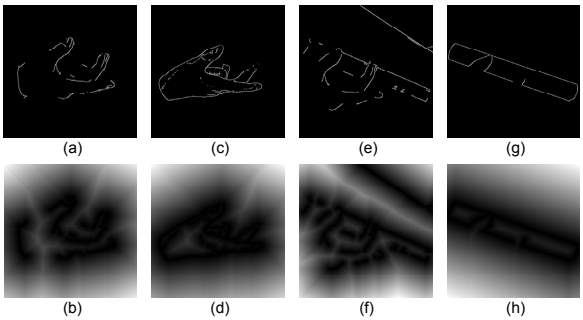

Figure 6: *Edge maps and distance transform images: (a)–(b) the edge map and distance transform image of the observed hand (e_o^{hand} and d_o^{hand}); (c)–(d) the edge map and distance transform image of the rendered hand (e_r^{hand} and d_r^{hand}); (e)–(f) the edge map and distance transform image of the whole observed image (e_o and d_o); (g)–(h) the edge map and distance transform image of the rendered object (e_r^{object} and d_r^{object}).*

Edge term. This term measures the discrepancy between the rendered and observed edge maps. We use a binary image to encode the edge map, whose edge and non-edge pixels are set to 1 and 0, respectively. Edges are not only relevant to the color of the hand and object but also related to their geometry and illumination conditions. As a result, they can be used to improve the tracking accuracy, particularly for objects without discernable textures.

We define the edge terms for the hand and object separately. Specifically, we have

$$E_{edge}^v = \frac{\sum e_r^{hand} \cdot d_o^{hand}}{\sum e_r^{hand} + \varepsilon} + \frac{\sum e_o^{hand} \cdot d_r^{hand}}{\sum e_o^{hand} + \varepsilon} + 2\frac{\sum e_r^{object} \cdot d_o^{object}}{\sum e_r^{object} + \varepsilon} \qquad (7)$$

where e_r^{hand} and e_o^{hand} are the rendered and observed edge maps of the hand. Canny edge detectors [1986] are applied to extract edges. d_r^{hand} and d_o^{hand} are the distance transform images of the rendered and observed edge images, respectively. The pixel values in our distance transform images indicate the distance of the pixels to the closest edge pixels in the corresponding edge images. As a result, the distance transform images provide a robust and smooth measurement for the edge maps. Figure 6 visualizes the edge maps and the corresponding distance transform images for the hand and object.

Intuitively, the first and second terms describe the discrepancies between the rendered and observed edge maps of the hand in a bi-directional manner. In contrast, the third term, where e_r^{object} is the rendered edge map of the object and d_o^{object} is the distance transform image of the observed edge map of the object, only measures the unidirectional inconsistency between the observed and rendered edge maps of the object. This is because silhouette maps of the ob-

ject in observed images usually cannot be extracted reliably. Again, ε is a small quantity to avoid division by zero.

4.3 Kinematic Motion Reconstruction

We now discuss how to use the cost function defined in Equation (3) to reconstruct kinematic motion of the hand and object from the observed image data. This step is important because the reconstructed kinematic poses will later be used to initialize our sampling-based control optimization process described in Section 6.

Pose tracking. Our kinematic motion reconstruction process runs in a sequential mode. Given an initial pose of the hand and object, the kinematic tracking process sequentially estimates 3D poses of the hand and object by minimizing the cost function defined in Equation (3). In addition, we include a smoothness term into the objective function to penalize the sudden changes of velocities (*i.e.* $||\mathbf{q}_t - 2\mathbf{q}_{t-1} + \mathbf{q}_{t-2}||^2$). We apply interacting simulated annealing (ISA) techniques [Gall et al. 2008], a variant of simulated annealing algorithm, to search an optimal solution at each frame. ISA is based on an interacting particle system that converges to the global optimum similar to simulated annealing. Briefly, we randomly sample a number of poses based on the reconstructed poses from the previous frame. We evaluate the cost of each sample, assign an appropriate weight to each sample, and use the weights to resample a number of new poses. This process is repeated until convergence. The final solution is a weighted combination of samples. In our experiment, we set the number of iterations to 25. And the number of particles at each iteration is set to 300.

Pose initialization. Sequential tracking, however, requires an initialization of kinematic poses at the first frame. We estimate the initial poses of the hand and object by registering their 3D mesh models with the first frame of multiview images. Specifically, we formulate the registration process as an optimization problem. The optimization function consists of the silhouette term defined in Equation (5) and the edge term defined in Equation (7). Note that we can easily extract silhouette maps of the hand and object via background subtraction because there was no overlap between the hand and object at the first frame. Again, ISA is applied to solve the registration problem. The global position and orientation of the hand are initialized by doing Iterative Closest Points (ICP) estimation between the visual hull of the hand at the first frame and the skinned hand mesh model under the default pose ("flat hand"). The joint angle pose of the hand is initialized by the same default pose. Note that we instructed the mocap subjects to start with the "flat hand" in our experiment. The global position and orientation of the object are initialized via a similar ICP process.

5 Composite Motion Control for Dexterous Manipulation

This section introduces a composite motion control for modeling dynamics of the hand and object as well as interaction and subtle contact phenomena between the two.

5.1 PD Control

All joints in the hand have proportional derivative (PD) controllers that are active at all times. At any time instance, the internal joint torques (τ_{pd}) of the hand are calculated as

$$\tau_{pd} = k_p(\bar{\mathbf{q}}^h - \mathbf{q}^h) - k_d(\dot{\mathbf{q}}^h), \tag{8}$$

where $\mathbf{q}^h$ and $\dot{\mathbf{q}}^h$ are the current joint angle pose and angular velocity of the hand and $\bar{\mathbf{q}}^h$ is the target hand pose. Intuitively, the PD controller drives the current pose $\mathbf{q}^h$ towards the target pose $\bar{\mathbf{q}}^h$.

Theoretically, directly searching the target hand pose ($\bar{\mathbf{q}}^h$) of PD servos could generate appropriate contact forces/torques (τ_{pd}) to track the observed object movement. In practice, even given the ground-truth reference trajectory of the hand, finding appropriate target poses of PD servos for dexterous manipulation is a daunting task due to huge search space. Imagine picking up a box on table. Searching the target poses in the vicinity of reference hand poses would probably work for "freehand motion" or a slow moving lightweight object. But large joint torques are often required to counteract inertia forces/moments and gravitational forces of the object. In other words, the target poses required for picking up an object would often be very different from the reference poses. Therefore, searching the target poses around the reference poses often fails to find appropriate target poses for PD servos.

This motivates us to introduce "virtual forces" to counteract object's inertia forces and moments as well as the force of gravity. It is worth pointing out that the virtual forces cannot be directly applied to the passive object, but are realized by active joint torques of the hand via contact forces/torques instead. In our application, we use the "virtual forces" to model internal joint torques of the hand required to move the simulated object to follow the desired trajectory of the object. Figure 7 visualizes the internal joint torques, as well as the corresponding contact forces/torques, required to realize the virtual forces that help to move the "simulated" object to follow the desired trajectory of the object.

5.2 Virtual Forces and Augmented Contact Forces

We now explain the virtual forces that help to drive the simulation of the object to match the reference trajectory of the object, including the reference position data $\bar{\mathbf{q}}^{o,p}$ and reference orientation data $\bar{\mathbf{q}}^{o,r}$. Since virtual forces are realized via contact forces/toques, we focus discussion on how to compute appropriate contact forces/torques to achieve the effects of virtual forces in the following.

First, a virtual force $\mathbf{f}_v$ is used to regulate the object to follow the reference position trajectory of the object according to a virtual PD controller defined as follows (see left/top in Figure7):

$$\mathbf{f}_v = k_p(\bar{\mathbf{q}}^{o,p} - \mathbf{q}^{o,p}) - k_d(\dot{\mathbf{q}}^{o,p}) - m_o\mathbf{g} \tag{9}$$

where $\mathbf{q}^{o,p}$ and $\dot{\mathbf{q}}^{o,p}$ are the current position and linear velocity of the object. $\bar{\mathbf{q}}^{o,p}$ is the target position of the object. The virtual force drives the current position of the object $\mathbf{q}^{o,p}$ towards its desired position $\bar{\mathbf{q}}^{o,p}$. We include constant gravity compensation force into the virtual forces so that the motion control can react to the object's mass.

Unlike the hand, we cannot directly apply the above PD control to simulate the object. This is because the object is not active and its movement is completely determined by contact forces. We, therefore, must compute appropriate contact forces $\mathbf{f}_{aug}$, which we term "augmented contact forces", to achieve the same effects of virtual forces applied to the object (see middle/top in Figure 7). This is a challenging task for two reasons. First, the solution is not unique when there are multiple contacts between the hand and object. For instance, when the hand is rotating a cube with four fingers, there are an infinite number of solutions that result in the same object movement. Second, augmented contact forces must be consistent with contact configurations between the hand and object and at the same time stay inside the friction cone.

To remove the ambiguity, we compute initial contact forces between the hand and object at the current step using the LCP solver in Open Dynamics Engine (ODE) library and use them to regular-

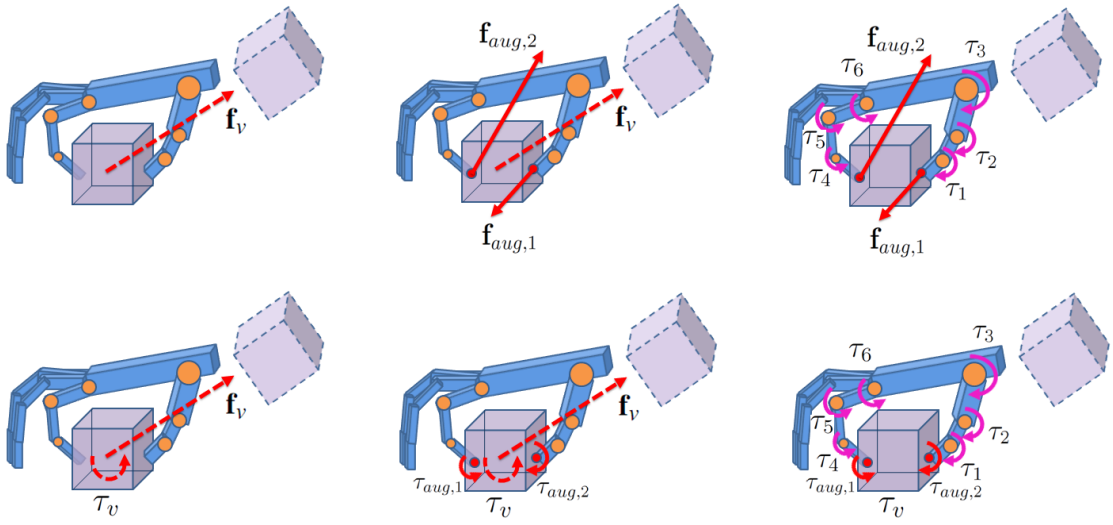

Figure 7: *Visualization of virtual forces/toques, augmented contact forces/torques, and internal joint torques that are required to move object to its target pose: (left) virtual forces $\mathbf{f}_v$ and torques τ_v; (middle) augmented contact forces $\mathbf{f}_{aug,i}$ and augmented contact toques $\tau_{aug,i}$ to achieve virtual forces and torques; (right) the required internal joint torques to achieve virtual forces and torques.*

ize the solution space of augmented contact forces. Briefly, at each time step, we first detect all the contact points between the hand and object using collision detection in ODE library. We then use the LCP solver [Cottle et al. 2009] to calculate the contact forces at each contact point, denoted as $\mathbf{f}_{c,i}, i = 1,...,K$, where K is the total number of contact points. We regularize the solution space of augmented contact forces by penalizing the deviation from the initial contact forces $\mathbf{f}_{c,i}, i = 1,...,K$. To ensure augmented contact forces to stay within the friction cone, we further constrain the augmented contact forces, denoted as $\mathbf{f}_{aug,i}, i = 1,...,K$, to point in the same directions as the initial contact forces: $\mathbf{f}_{aug,i} = w_i \mathbf{f}_{c,i}$, $w_i \geq 0$, $i = 1,...,K$.

In our implementation, we formulate the problem in an optimization framework. We optimize the augmented contact forces by solving the following quadratic programming (QP) problem:

$$\begin{aligned} &\min_{w_1,...,w_K} \ \|\mathbf{f}_v - \textstyle\sum_{i=1}^{K} w_i \mathbf{f}_{c,i}\|^2 + \lambda_1 \textstyle\sum_{i=1}^{K} (w_i - 1)^2 \\ &\text{subject to } w_i \geq 0, \ \ i = 1,...,K, \end{aligned} \tag{10}$$

where the first term ensures the resultant of augmented contact forces $\mathbf{f}_{aug,i} = w_i \mathbf{f}_{c,i}$ matches the virtual forces $\mathbf{f}_v$. The second term is a regularization term which penalizes the difference between the augmented contact forces and the initial contact forces from the LCP solver. The inequality constraints ensure augmented contact forces face the same directions as the initial contact forces because fingers can only push but not pull at contact points. The weights λ_1 are set to 1.

Thus far we have focused on the virtual forces for translating the object. We now take into account virtual torques for rotating the object. Similarly, we introduce a virtual torque τ_v to follow the desired orientation of the object. This allows us to drive the current object orientation $\mathbf{q}^{o,r}$ towards the desired object orientation $\bar{\mathbf{q}}^{o,r}$. Again, we need to estimate augmented contact torques at each contact point, denoted as $\tau_{aug,i}, i = 1,...,K$, based on the virtual torque τ_v (see middle/bottom in Figure 7). In addition, the augmented contact torques also need to compensate the net torques caused by the augmented contact forces at each contact point.

We model the augmented contact torque at each contact patch using a torsional torque: $\tau_{aug,i} = l_i \vec{n}_i$, where l_i is the scale of the i-th augmented contact torque and $\vec{n}_i$ is the unit vector of the surface normal at the i-th contact patch. Torsional torques are commonly used in humanoid robotics and biomechanics community (*e.g.*, [Lee and Goswami 2010]) to model the resultant torque of multiple contact forces from the same patch. Mathematically, the resultant of multiple contact forces applied at a planar contact patch can be represented as a combination of a contact force and torsional torque applied at the center of pressure (COP). Note that COP is chosen to ensure the net torque caused by vertical contact forces at multiple points is zero. As a result, the net torque caused by multiple contact forces at the COP is just a torsional torque caused by multiple friction forces. Another benefit of torsional torques is to enable us to approximately model the contact torques caused by soft bodies, thereby producing more subtle contact phenomena between the object and hand.

Again, we formulate the estimate of augmented contact torques as a quadratic programming problem. We have

$$\min_{l_1,...,l_K} \ \|\textstyle\sum_{i=1}^{K} (\mathbf{r}_i \times \mathbf{f}_{aug,i} + l_i \vec{n}_i) - \tau_v\|^2 + \lambda_2 \textstyle\sum_{i=1}^{K} l_i^2 \tag{11}$$

where $\mathbf{r}_i$ is the vector from the center of mass of object to its i-th contact point. The first term measures the difference between the virtual torques and the net torques generated by augmented contact forces and torques. The second term is the regularization term. The weight λ_2 is set to 1.

Contact forces and torques can be optimized jointly or sequentially using Equation (10) and (11). In our implementation, we choose sequential optimization to prioritize force optimization because it results in a smaller optimization space and therefore requires less computation time.

5.3 Composite Motion Control

Once we compute both augmented contact forces and torques, we apply the Jacobian transpose to obtain the internal joint torques required to move the simulated object toward the target pose $\bar{\mathbf{q}}^o$ (see Figure 7(right)):

$$\tau_{hand,v} = \sum_{i=1}^{k} J^T \mathbf{f}_{aug,i} + A^T \tau_{aug,i} \tag{12}$$

where $\tau_{hand,v}$ is active joint forces and torques of the hand requires to achieve virtual forces and torques that move the object to follow the target poses. The matrix J and A are the Jacobian matrices that map the instantaneous internal joint angular velocities to the instantaneous linear velocities and angular velocities at the i-th contact point under the current pose.

Our final motion control for dexterous grasping and manipulation consists of two components: (1) the PD control τ_{pd} which drives the current hand pose to the desired hand pose; and (2) the augmented joint torques $\tau_{hand,v}$ that drive the current object pose to the desired object pose, subject to contact and friction limits constraints. Specifically, we have

$$\tau_{joint} = \tau_{pd} + \tau_{hand,v} \tag{13}$$

The final joint torques τ_{joint} are now dependent on the desired poses of the hand and object $\bar{\mathbf{q}}_t = [\bar{\mathbf{q}}^h, \bar{\mathbf{q}}^o]$. The control representation also considers contact and friction constraints at each contact point as well as the gravity compensation of object. Therefore, it is suitable to track both hand and object trajectories.

6 Sampling-based Control Optimization

In this section, we discuss how to search an optimal motion control that advances the simulation to best match the observed image data. This requires searching both the initial state $\mathbf{s}_0 = [\mathbf{q}_0, \dot{\mathbf{q}}_0]$ and the target poses $\bar{\mathbf{q}}_0, ..., \bar{\mathbf{q}}_{T-1}$ across the entire sequence. Our solution is to use random sampling techniques to search the target poses of the hand and object in the vicinity of the reference poses obtained from kinematic motion tracking described in Section 4. Similarly, we set the initial state $\mathbf{s}_0$ to the reference initial state obtained from kinematic motion tracking. Sampling-based techniques are well suited to this task because of their lack of dependence on derivatives, which are difficult to estimate in contact-rich scenarios.

Briefly, at any time step t, we initialize the target poses using the reference poses obtained from the kinematic motion tracking process. The algorithm then samples in the vicinity of the initial target poses to produce a number of new target poses to advance the simulation; and a series of new states are then generated via forward dynamics simulation. We can select the best states based on the cost function defined in Equation (3) and iterate the process using the selected best states as the start state at time $t + \delta t$. Progressively advancing the simulation in this fashion will eventually return a simulated motion that precisely matches the observed image data. In our experiment, we set the sampling time step and the simulation time step to 0.05s and 0.0005s, respectively.

Random sampling around the reference poses, however, does not take into account contact configuration between the hand and object. Note that reference poses from kinematic motion tracking are often very noisy and fail to reconstruct accurate contact information between the hand and object. Take "grasping the cup's handle" as an example, the reference poses from kinematic motion tracking often include completely wrong contact information of contact fingers (see Figure 10). Therefore, sampling in the vicinity of the reference poses might produce a large number of samples with wrong contact information, thereby failing to accomplish dexterous manipulation tasks (Figure 8(left)).

Contact-based sampling. We now describe our idea of how to incorporate contact information into the sampling process. The challenge here is that an accurate detection of contact information is almost impossible due to noisy reference poses obtained from kinematic tracking process. To address this challenge, we derive a complete list of contact configurations consistent with noisy reference poses within a particular error bound. During the sampling process, we go through each of contact configurations in the list one by one and incorporate the corresponding contact information into the sampling process.

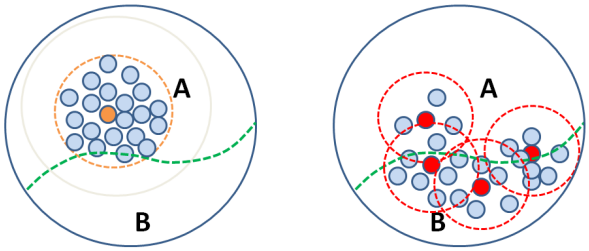

Figure 8: *Contact-based sampling: (left) randomly sampling in the vicinity of the noisy reference pose without considering contact configuration might produce bias towards the wrong contact configuration. The "orange" and "blue" circles represent the reference pose obtained from kinematic tracking and the sampled target poses, respectively. (right) our contact-based sampling process first generates a small number of "seed" poses based on potential contact configuration between the hand and the object and then randomly samples the target poses around each of the "seed" poses. The "red" circles represent the "seed" poses generated by the contact-based sampling process. Note that region "A" and "B" represent two possible contact states between the object and a particular hand bone segment.*

Given an initial target pose obtained from kinematic tracking process, we first compute the closest distances between the object and each bone segment of the hand to determine if there is a potential contact between the object and each bone segment of the hand. If the closest distance is larger than a threshold, we assume there is "no contact" between the object and bone segment. Otherwise, there is a "potential contact". For a "potential contact", we further assume the probability of being "contact" or "no contact" is equal. This allows us to enumerate all the possible contact configurations consistent with noisy kinematic poses within a particular error bound. For k "potential contact" states, there are at most 2^k contact combinations in total, each of which is associated with a particular probability value. The contact-based sampling process first samples contact configurations between the hand and object based on the probability values associated with each contact combination. For each of generated contact configuration samples, we modify the initial target pose using inverse kinematics techniques in such a way that the contact state of the new pose is consistent with the sampled contact configuration (see Figure 8(right)). The new poses are then used as the "seed" poses to randomly sample the target poses. Specifically, we draw a number of samples uniformly in the hypercubes centered at each of the new poses and use them as new target poses. This strategy significantly speeds up the converge of our search process.

Sampling selection. One remaining issue is how to select samples at each time step. Each sample is associated with an energy value computed by the objective function defined in Equation (2). One possible solution is to adopt a greedy strategy by saving the best samples at each time step. However, this strategy is often prone to fall into local minima. This is because the sample with the lowest energy does not necessarily produce successive low energy samples in the long term. We address this challenge by adopting a similar strategy used in [Liu et al. 2010]. First, we discard samples lying in the top 50% of the energy distribution. Let E_{low} and E_{high} be the lowest and highest energy of the remaining samples.

Joint	K_p	K_d	Joint	K_p	K_d
$root_{translate}$	400.0	50.0	$root_{rotate}$	600.0	50.0
$thumb_1$	10.0	0.3	$thumb_2$	8.0	0.2
$thumb_3$	4.0	0.4	$index_1$	4.0	0.2
$index_2$	3.0	0.1	$index_3$	2.0	0.1
$middle_1$	4.0	0.4	$middle_2$	4.0	0.1
$middle_3$	2.0	0.1	$ring_1$	4.0	0.4
$ring_2$	3.0	0.1	$ring_3$	2.0	0.1
$pinky_1$	4.0	0.4	$pinky_2$	3.0	0.1
$pinky_3$	2.0	0.1			

Table 1: *Parameters for physics simulation. Finger id is named in a way from the root bottom to tip. For example: $thumb_1$ is the thumb joint closest to the root. And $thumb_3$ is the tip of the thumb joint. All the examples have the same K_p, K_d.*

And let m be the number of samples we want to keep. We select m samples $s_0,...,s_{m-1}$ to cover the range of $[E_{low},E_{high}]$ and at the same time distribute mostly in the low cost region. In our implementation, we select samples based on a polynomial function $s(x) = (E_{high} - E_{low})x^6 + E_{low}$, where $x = i/m, i = 0,...,m-1$. We save the samples that have the closest cost to $s(x)$. For all examples we tested, we found 1500 samples per step is sufficient to generate good reconstruction results. And after pruning and diversification, we finally save $m = 150$ samples for each step.

7 Evaluation and Results

We have demonstrated the power of our system by capturing a wide range of dexterous grasping and manipulation tasks. In our experiment, we recorded the multi-view image sequences with a resolution of 1024×768 using six synchronized cameras running at a frame rate of 20 frames per second (fps). We use the Open Dynamics Engine (ODE) to simulate our motion. Table 1 shows all the simulation parameter values used in our motion control. Our results are best seen in the accompanying video although we show sample frames of a few motions in the paper (see Figure 9).

7.1 Testing on Real Data

We have tested the effectiveness of our motion capture system on grasping and manipulation of four different objects, including ball, cup, cube and stick. By assuming the object has uniform density, we calculate the inertia of each object by discretizing internal volumes of object meshes into 3D voxels and integrating over the discretized voxels. The accompanying video shows that the system can capture subtle interaction between the hand and object, such as "grasping the cup's handle, lifting it, and pouring three times", "grasping a cube and continuously spinning it with fingers", and "picking up a long round stick on the table, rotating it continuously in the air, stopping and rolling it in the palm". In addition, we have shown that the system can capture subtle interaction with multiple objects. For example, the 4-th row in Figure 9 shows our result on capturing interaction between the hand, ball, and cup, specifically, "grasping a cup, turning it upside down, and placing it on the table, then picking up a ball and put it inside the cup."

System robustness. Our evaluation shows that the system is capable of capturing high-fidelity manipulation data for objects with/without distinctive texture patterns. Both the "ball" and the "stick" are uniform-colored while the "cube" and the "cup" contain discernible texture patterns. The system also demonstrates the robustness of tracking both diffuse and non-diffuse objects. Note that the "cup", the "stick", and the "cube" have non-diffuse surfaces and while the surface of the "ball" is approximately diffused. In addition, the system is capable of acquiring fast motion with complicated backgrounds (see the 5-th row in Figure 9). For example, the accompanying video shows we can accurately acquire fast hand manipulation data under a very complicated background with non-diffuse surfaces.

7.2 Generalization of Motion Capture Data

One appealing property of recovering physics-based motion control from video is to apply the reconstructed motion control to animate new objects with different properties. In our experiment, we have retargeted the captured motion to interact with new objects with significantly different geometries (Figure 2). In addition, we have generalized the captured motion to grasp and manipulate the same objects by changing their physical properties such as friction coefficients.

Retargeting to new objects. Motion retargeting is particularly appealing to dexterous grasping and manipulation because we can capture hand manipulation data for a small number of prototype objects and adapt the captured motion data to new objects that may not be available for real-word capture. Given a motion control reconstructed from video, we first align the target object (*e.g.* "happy buddha") with the "source" object (*e.g.* "cup"). This is achieved by aligning the center of mass and major axes of the target object with those of the source object. Next, we apply the recovered motion control to the target object and generate physically realistic motion that is consistent with the target object. In our video, we show the motion retargeting results from the "cube" to a "soap bar", a "tea pot", and a "Chinese dragon" (Figure 2(top)). We also demonstrate how to retarget the reconstructed motion control of grasping and manipulating the "cup" to three significantly different objects (Figure 2(bottom)), including a "sauce bottle", a "wine bottle", and a "happy buddha".

Modifying physical properties. We can edit the captured motion data by applying the recovered motion control to grasp and manipulate the same objects with different physical properties. The accompanying video shows the effects of modifying the friction coefficient of the "cube". In particular, we change the friction coefficient of the "cube" from "0.75" to "0.3", "0.2" and "0.01", respectively. As the friction coefficient gradually decreases, the simulated object appears more slippery in the hand and eventually falls off.

7.3 Comparisons Against Marker-based Systems

In this section, we first evaluate the effectiveness of our system by comparing against alternative methods, including marker-based motion capture [Vicon Systems 2012] and a combination of marker-based mocap and RGBD data [Zhao et al. 2012]. The video shows superior performance of our system for one test sequence "grasping the cup's handle, lifting, and pouring three times".

Comparison against Vicon [2012]. Since we were unable to co-locate the video-based mocap system and the optical mocap system in the same room, we have instructed the same subject to perform the same manipulation task in front of both systems. The marker-based motion capture system is based on Vicon optical mocap system [2012]. In particular, we attached a full marker set (21 markers) on the hand and used twelve Vicon cameras to capture hand manipulation data. We also attached six markers on the object to capture object movement. We manually labeled all the markers for each frame and used them to reconstruct the poses of the hand and object across the entire sequence. The video shows poor tracking results for all three contact fingers, including the thumb, index and middle. In addition, the reconstructed motion displays inaccurate interaction and contact phenomena between the hand and object.

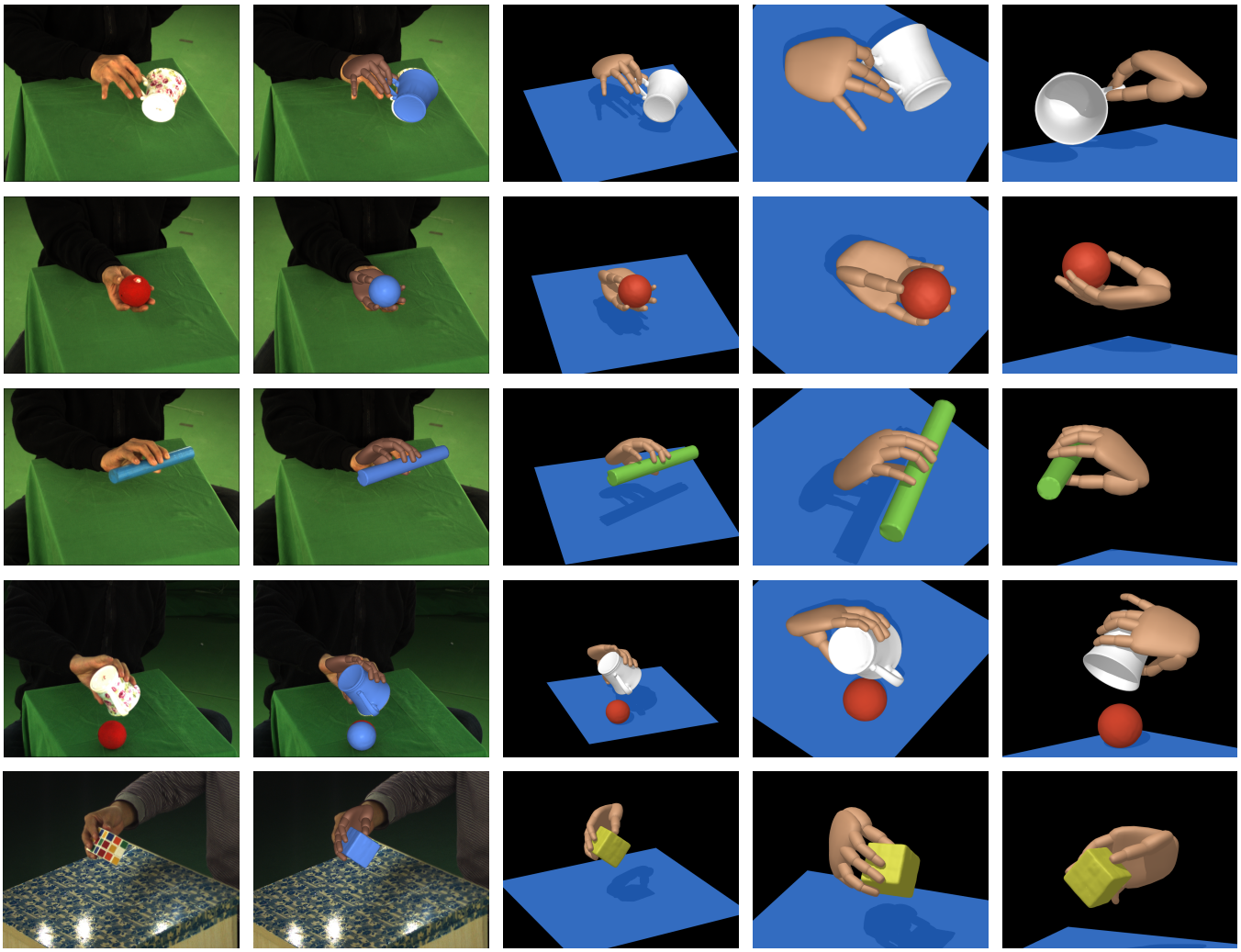

Figure 9: *Video-based motion capture of dexterous manipulation: we show input images, the reconstructed poses superimposed on the input images, the reconstructed poses in the same viewpoint, and the reconstructed poses from two novel viewpoints (from left to right).*

Comparison against Zhao et al. [2012]. Zhao and his colleagues [2012] recently extended marker-based mocap methods by combining with RGBD image data obtained from a *Kinect* and demonstrated state-of-the-art accuracy for hand motion capture. We, therefore, also compare our system against theirs. We first extended the system to hand manipulation capture since their system is mainly focused on capturing hand articulations alone. Similar to their system, we formulated the pose reconstruction as an optimization problem, which maximizes the consistency between the reconstructed poses of the hand and object and the observed data, including both 3D marker positions obtained from the Vicon system and RGBD image data captured by a single *Kinect* camera. Like their system, we employed Particle Swarm Optimization (PSO) algorithm to search optimal poses of the hand and object over time. Complementing marker-based mocap with RGBD data improves the tracking accuracy. However, their method still suffers from the same occlusion problem and fails to track contact fingers accurately, thereby resulting in wrong interaction between the hand and object.

7.4 Comparison Against Kinematic Motion Tracking

We evaluate the importance of physics-based motion control to video-based hand manipulation capture by comparing our method against kinematic tracking process with/without collision detection. The accompanying video shows a side-by-side comparison between our algorithm and kinematic motion tracking described in Section 4.3. Kinematic motion tracking results appear noisy and jerky. They are also physical implausible and fail to capture accurate interaction and contact phenomena between the hand and object (see the second column of Figure 10). Smoothing kinematic motion data across the entire sequence reduces jerky effect but cannot remove implausible interaction and contact phenomena between the hand and object (see the third column of Figure 10). This is because kinematic motion tracking alone often cannot determine whether a particular finger (*e.g.* the thumb) contacts the handle or not because of significant occlusions and noisy image measurement.

In addition, our evaluation video shows a side-by-side comparison between our result and kinematic tracking with collision detection. We added a collision detection constraint term into Equation (3) to penalize the penetration between the hand and object. We used the same configuration (the same number of particles and the same

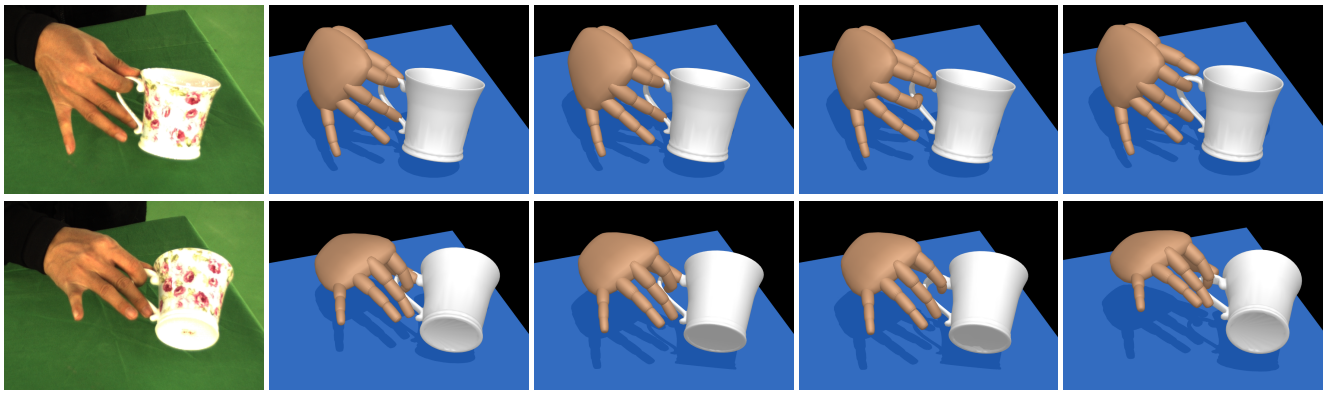

Figure 10: *Kinematic motion tracking vs. physics-based motion capture. From left to right, we show original image data, kinematic motion tracking result, kinematic motion tracking result after temporal filtering, kinematic motion tracking with collision detection, and physics-based motion capture result. Note that filtering kinematic mocap data cannot correct wrong contact information between the hand and object.*

number of iterations in ISA) to do kinematic tracking with collision detection. Adding the collision detection constraint term into kinematic motion tracking removes the penetration artifacts between the hand and object. However, similar to kinematic motion tracking, there is no guarantee that contact phenomena and subtle interaction between the hand and object are physically plausible due to ambiguity caused by the occlusions and noisy image measurement. Sometimes, the optimization intentionally avoids the hand-object penetration to produce even worse results than the kinematic motion tracking. This often results in unnatural interaction between the hand and object (see the last column of Figure 10).

With physics-based motion control, our system produces physically realistic manipulation data which not only are consistent with the observed image data but also model physically plausible interaction and contact phenomena between the hand and object. More importantly, recovering physics-based motion control from video allows us to conveniently adapt the captured motion to new objects with different properties, a capability that has not been demonstrated by any kinematic motion tracking process.

7.5 More Evaluations

We now evaluate key components of our video-based hand manipulation capture process.

With/without virtual forces. To demonstrate the importance of virtual forces to our motion control, we compare the reconstruction results using motion control with and without virtual forces. Specifically, we reconstructed the motion by using PD controllers described in Equation (8) and compared against the reconstruction results obtained by the composite motion control described in Equation (13). We adopted the same sampling strategy and generated the same number of samples for each time step. The accompanying video shows a side-by-side comparison between the two. Without the "virtual forces", the system fails to generate an appropriate motion control for picking up the object. This is because PD control without "virtual forces" ignores the desired movement of the object and thereby fails to find an appropriate motion control to follow the movement of the object.

The importance of contact-based sampling. This experiment shows the effectiveness of our contact-based sampling in sampling-based control optimization. For comparison's sake, we used the same number of samples (1500) to optimize motion control at each

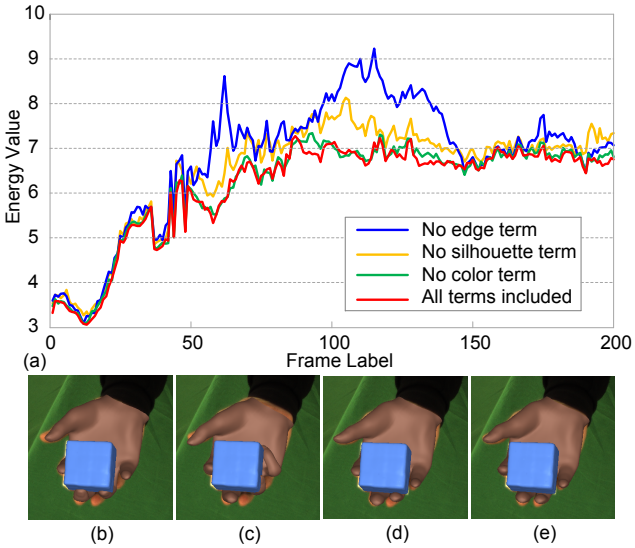

Figure 11: *Evaluation of kinematic tracking terms: (a) the tracking errors of the reconstructed poses evaluated by Equation (3); (b-e) the reconstructed poses projected in a novel camera viewpoint for each method, including (b) no edge term,(c) no silhouette term, (d) no color term, and (e) all terms included.*

step. Our accompanying video shows a side-by-side comparison between the sampling methods with and without contact consideration. Without contact consideration, random sampling techniques failed to track the reference motion right after the hand started to interact with the object. In contrast, our contact-based sampling method successfully tracked the reference trajectory with the same number of samples.

Reducing the number of cameras. Our system employs multiple video cameras to capture physically realistic hand manipulation data. Here we evaluate how increasing/decreasing the number of video streams affects the quality of output motions. We have observed reducing the number of cameras from six to five does not degrade the quality of output motion. However, when the number of cameras is smaller than five, the quality of output motion begins to drop off. This is mainly caused by poor kinematic tracking data.

Please refer to our evaluation video for more details.

Evaluation of three image terms. The objective function for kinematic tracking consists of three image terms, including *silhouette* term, *color* term and *edge* term. We have evaluated the importance of each term to our kinematic motion tracking process. Specifically, we dropped off each term described in Equation (3) and used them to sequentially track kinematic poses of the hand and object from multiple video streams. We compared the results obtained by dropping off each term against the results obtained by a combination of all three terms. For comparisons, we started with the same initialization poses and employed the same optimization techniques (ISA) to track kinematic poses over time. We also used the same number of particles for ISA.

Since we do not have ground truth tracking data, we choose to evaluate the quality of the reconstruction motions in the two following ways:

- One way to evaluate the quality of the reconstructed poses is based on the overall cost function described in Equation (3). Figure 11(a) shows the energy of the tracking results obtained by each of tracking methods. The energy curves indicate the tracking errors for each of tracking methods.
- Alternatively, we can evaluate the quality of the reconstruction results by projecting the reconstructed poses into an image sequence taken from a novel viewpoint. Specifically, we excluded an image sequence obtained from one camera view from the cost function evaluation and evaluate the accuracy of each tracking method by projecting the reconstructed poses into the novel image sequence. Figure 11(b–e) shows the reconstructed poses of each method projected into a novel camera viewpoint.

Both evaluation methods indicate that a combination of *silhouette* term, *edge* term, and *color* term produces the best tracking results.

Timings. In all our experiments, the computational times of the video-based kinematic motion tracking were about 2 to 2.5 minutes per frame which consists of six multi-view images. And for sampling-based optimization process, the computational times were about 1 to 1.5 minutes per frame. All of our experiments were tested on an Intel core i7 CPU, 8GB RAM with NVIDIA GTX580 graphics cards. The whole system for motion capture takes about 3 to 4 minutes per frame on a standard PC using unoptimized code.

8 Conclusion and Discussion

In this paper, we present an end-to-end video-based motion capture system for acquiring physically realistic dexterous manipulation data from multiple video streams. Our system is appealing to hand manipulation capture because it requires no markers, no gloves or no sensors. It naturally models hand articulation, object movement and subtle interaction between the hand and object and thereby produces physically realistic motions that are consistent with real world observations.

We demonstrate the power and effectiveness of our approach by modeling a wide variety of dexterous manipulation data, including grasping and manipulation of four different objects, from video. The result shows that our system is robust to variations in object appearances and background conditions. In addition, we show the generalization of our recovered motion controllers by adapting them to manipulating new objects with different properties.

Our system benefits from the combined power of video-based motion capture and physics-based motion modeling. Physics-based motion modeling is a mathematically ill-posed problem because there are many ways to adjust a motion so that physical laws are satisfied, and yet only a subset of such motions are consistent with real world observations. By accounting for physical constraints and observed image data simultaneously, we can generate physically plausible motions that are consistent with real world observations. On the other hand, video-based motion capture can utilize physical constraints to reduce modeling ambiguity and ensure the reconstructed motion is physically plausible.

Recently, there is an increasing body of work regarding the use of GPUs or parallel computing for general-purpose computing. We believe most components of the current system can be GPU-accelerated. For example, both search processes, including interactive simulated annealing and random sampling, can be implemented on a GPU. The evaluation of the objective function in Equation (3) is also well suitable for GPU accelerations because it mainly involves image processing, which is easy to parallelize. One of the immediate directions for future work is, therefore, to speed up the system with GPU implementations and parallel computing.

We have tested our system by capturing a wide range of manipulation tasks involving four different objects. In the future, we would like to test our system on more objects and manipulation tasks. This would enable us to build a comprehensive and detailed grasping and manipulation database required for data-driven hand manipulation. We are also interested in modifying and reusing the captured motion data to achieve new tasks such as motion transformation, interpolations, and composition.

Acknowledgement

The authors would like to thank Wenping Zhao for his assistance of experiments in comparison against marker-based systems [Vicon Systems 2012; Zhao et al. 2012]. This work was supported in part by the National Science Foundation under Grants No. IIS-1065384 and IIS-1055046 and the NSFC under Grants No. 61035002, No. 60932007, No. 61073072, and No. 61021063.

References

ATHITSOS, V., AND SCLAROFF, S. 2003. Estimating 3d hand pose from a cluttered image. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, CVPR '03, 432–439.

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

BRUBAKER, M. A., AND FLEET, D. J. 2008. The Kneed Walker for human pose tracking. In *Proceedings of IEEE CVPR*, 1–9.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*. 8(6): 679–698.

COTTLE, R., PANG, J., AND STONE, R. 2009. *The linear complementarity problem*. Society for Industrial Mathematics.

CYBERGLOVE, 2012. http://www.cyberglovesystems.com/.

DE LA GORCE, M., FLEET, D. J., AND PARAGIOS, N. 2011. Model-based 3d hand pose estimation from monocular video. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, 1793–1805.

DEBEVEC, P., YU, Y., AND BORSHUKOV, G. 1998. Efficient view-dependent image-based rendering with projective texture-mapping. In *9th Eurographics Rendering Workshop*, Springer.

GALL, J., ROSENHAHN, B., AND SEIDEL, H.-P. 2008. Human motion - understanding, modeling, capture and animation. *Computational Imaging and Vision*. 36: 319–345.

HOYET, L., RYALL, K., MCDONNELL, R., AND O'SULLIVAN, C. 2012. Sleight of hand: perception of finger motion from reduced marker sets. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, 79–86.

KRY, P. G., AND PAI, D. K. 2006. Interaction capture and synthesis. *ACM Trans. Graph. 25*, 3, 872–880.

LEE, S.-H., AND GOSWAMI, A. 2010. Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 3157–3162.

LIU, L., YIN, K., VAN DE PANNE, M., SHAO, T., AND XU, W. 2010. Sampling-based contact-rich motion control. *ACM Trans. Graph. 29*, 4, 128:1–128:10.

LIU, C. K. 2008. Synthesis of interactive hand manipulation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, 163–171.

LIU, C. K. 2009. Dextrous manipulation from a grasping pose. *ACM Trans. Graph. 28*, 3, 59:1–59:6.

OIKONOMIDIS, I., KYRIAZIS, N., AND ARGYROS, A. A. 2011. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2088–2095.

PALMER, J. A., KREUTZ-DELGADO, K., AND MAKEIG, S. 2006. Super-gaussian mixture source model for ica. In *Independent Component Analysis and Blind Signal Separation*. Springer, 854–861.

POLLARD, N. S., AND ZORDAN, V. B. 2005. Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, 311–318.

REHG, J., AND KANADE, T. 1995. Model-based tracking of self-occluding articulated objects. In *Proceedings of the Fifth IEEE International Conference on Computer Vision*, ICCV '95, 612–617.

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

SUEDA, S., KAUFMAN, A., AND PAI, D. K. 2008. Musculotendon simulation for hand animation. *ACM Trans. Graph. 27*, 3, 83:1–83:8.

TSANG, W., SINGH, K., AND FIUME, E. 2005. Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 319–328.

VICON SYSTEMS, 2012. http://www.vicon.com.

VONDRAK, M., SIGAL, L., AND JENKINS, O. C. 2008. Physical simulation for probabilistic motion tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

VONDRAK, M., SIGAL, L., HODGINS, J., AND JENKINS, O. 2012. Video-based 3d motion capture through biped control. *ACM Trans. Graph. 31*, 4 (July), 27:1–27:12.

WANG, R. Y., AND POPOVIĆ, J. 2009. Real-time hand-tracking with a color glove. *ACM Trans. Graph. 28*, 3, 63:1–63:8.

WEI, X., AND CHAI, J. 2010. Videomocap: modeling physically realistic human motion from monocular video sequences. *ACM Trans. Graph. 29*, 4 (July), 42:1–42:10.

WU, Y., LIN, J., AND HUANG, T. S. 2001. Capturing natural hand articulation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 426–432.

YE, Y., AND LIU, C. K. 2012. Synthesis of detailed hand manipulations using contact sampling. *ACM Trans. Graph. 31*, 4 (July), 41:1–41:10.

ZHANG, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the International Conference on Computer Vision*. 666–673.

ZHAO, W., CHAI, J., AND XU, Y.-Q. 2012. Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, 33–42.

ZHOU, H., AND HUANG, T. S. 2003. Tracking articulated hand motion with eigen dynamics analysis. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, ICCV '03, 1102–1109.