ManipNet: Neural Manipulation Synthesis with a Hand-Object Spatial Representation

HE ZHANG, The University of Edinburgh
YUTING YE, Facebook Reality Labs
TAKAAKI SHIRATORI, Facebook Reality Labs
TAKU KOMURA, The University of Hong Kong and The University of Edinburgh

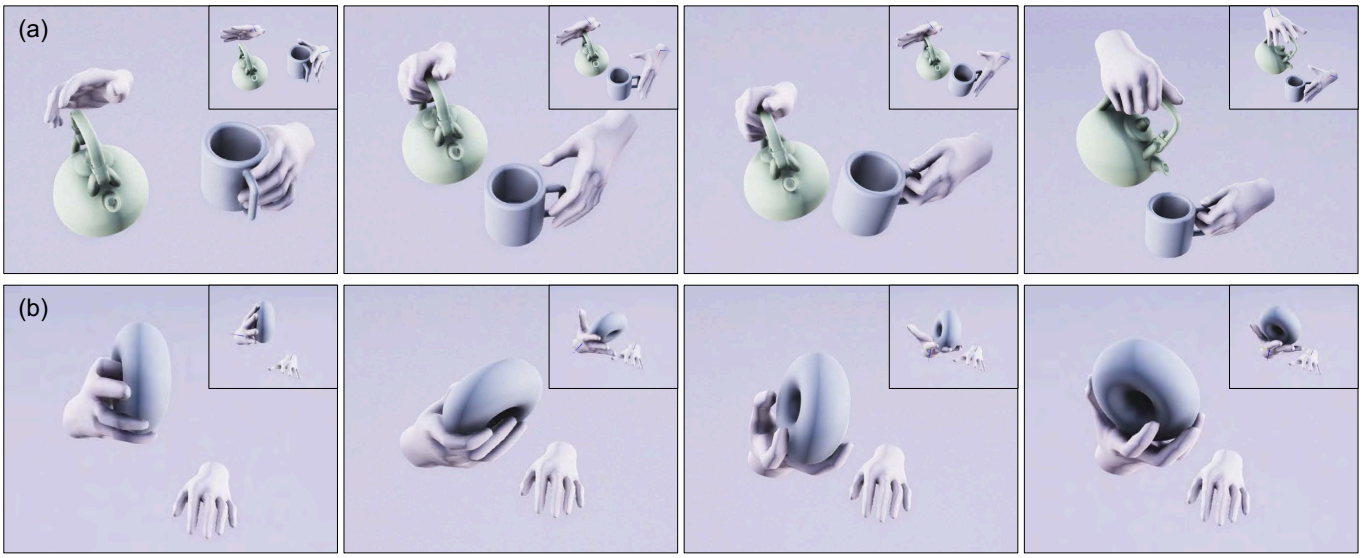


Fig. 1. Given the wrist and object trajectories, the system is able to synthesize dexterous manipulation of the object based on the proposed hand-object spatial representation at interactive frame rate, such as (a) the right hand grasps and holds a teapot while the left hand regrasps a mug and (b) the right hand turns a torus in hand.

Natural hand manipulations exhibit complex finger maneuvers adaptive to object shapes and the tasks at hand. Learning dexterous manipulation from data in a brute force way would require a prohibitive amount of examples to effectively cover the combinatorial space of 3D shapes and activities. In this paper, we propose a hand-object spatial representation that can achieve generalization from limited data. Our representation combines the global object shape as voxel occupancies with local geometric details as samples of closest distances. This representation is used by a neural network to regress finger motions from input trajectories of wrists and objects. Specifically, we provide the network with the current finger pose, past and future trajectories, and the spatial representations extracted from these trajectories. The network then predicts a new finger pose for the next frame as an autoregressive model. With a carefully chosen hand-centric coordinate system, we can handle single-handed and two-handed motions in a unified framework. Learning from a small number of primitive shapes and kitchenware objects, the network is able to synthesize a variety of finger gaits for grasping, in-hand manipulation, and bimanual object handling on a rich set of novel shapes and functional tasks. We also demonstrate a live demo of manipulating virtual objects in real-time using a simple physical prop. Our system is useful for offline animation or real-time applications forgiving to a small delay.

CCS Concepts: • **Computing methodologies → Motion capture**.

Additional Key Words and Phrases: Deep Learning, Neural Network, Motion Capture, Hand Motion, Manipulation, Interaction

ACM Reference Format:
He Zhang, Yuting Ye, Takaaki Shiratori, and Taku Komura. 2021. ManipNet: Neural Manipulation Synthesis with a Hand-Object Spatial Representation. *ACM Trans. Graph.* 40, 4, Article 121 (August 2021), 14 pages. https://doi.org/10.1145/3450626.3459830

Authors' addresses: He Zhang, The University of Edinburgh, he.zhang@ed.ac.uk; Yuting Ye, Facebook Reality Labs, yuting.ye@fb.com; Takaaki Shiratori, Facebook Reality Labs, tshiratori@fb.com; Taku Komura, The University of Hong Kong , The University of Edinburgh, taku@cs.hku.hk, tkomura@ed.ac.uk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2021/8-ART121 $15.00
https://doi.org/10.1145/3450626.3459830

1 INTRODUCTION

People interact with 140 objects on average daily [Zuccotti 2015] without much thought. These everyday objects are also remarkably diverse in shape and form across different cultures and geography. No other intelligent agents can yet match such dexterity of the human hand. In feature films and games, finger motions still involve tedious manual animation or cleaning of motion capture data, especially those involving object interactions. In virtual or augmented reality, the controller-based or pinch-based object interaction model falls short of producing subtle and realistic finger movements, breaking immersion in an otherwise magical experience. Robot hands can learn to perform dedicated tasks precisely, such as swing-peg-in-hole [Chebotar et al. 2019], connector insertions [Schoettler et al. 2019], or solving a Rubik's cube [Andrychowicz et al. 2020], thanks to recent advancements in deep reinforcement learning. However, a general model that can perform diverse manipulation tasks on a wide variety of objects is still beyond reach, let alone performing with grace.

In this work, we choose to learn natural manipulation behaviors directly from data using a deep neural network given the explosive success of deep learning. However, the combinatorial nature of a manipulation [Touvet et al. 2014] exacerbates the hunger for high quality data. A manipulation depends not only on the shape, size, and functionality of the object, but also on the intended task, the hand anatomy, and even personal preferences. Although it is now possible to capture hand-object interaction motions in real time [Han et al. 2018], the vast space of variations still seems daunting. Some researchers are dedicated to improving the quality and quantity of datasets for the community [Brahmbhatt et al. 2020; Taheri et al. 2020]. We instead aim to investigate how much we can generalize learning from a small number of object shapes to geometric variations, and to extend the supported manipulations beyond purposeful and goal-driven grasping to subtle finger gaits and in-hand manipulations people often perform naturally and unintentionally.

Our main idea is to utilize features that represent the spatial relation between the hand and the object, with insights from biomechanics literature on grasping. Studies [Jarrassé et al. 2014; Santello et al. 1998] indicate that the overall grasping pose lies in a low dimensional space that is largely determined by the task. Higher-order variations of the hand pose depend on details of the object shape. Moreover, variations in hand pose start to form as the hand is getting closer to the object [Santello et al. 2002]. We therefore choose a coarse representation for the global object shape, and a dense representation for local geometric details of the object surface, but only when the hand is in close proximity. Many solutions exist to represent 3D geometry for neural networks, such as voxel occupancy grids [Wu et al. 2015], signed distance fields [Dai et al. 2017; Song et al. 2017], and point samples [Qi et al. 2017a,b]. In our case, we attempt to reduce the feature dimensions to avoid overfitting while still capture important information. We use a low resolution voxel occupancy grid to represent the object shape. We find that distance samples between the hand and the object surfaces are effective low dimensional signals that capture details well.

Specifically, we train a neural network to predict finger poses of object manipulations from control signals and object geometric features. Control signals are 6D trajectories of the wrists and of the object. Previous work of Ye and Liu [2012] already demonstrated the efficacy of these control signals in constraining finger motions of a manipulation. However, a deep learning formulation additionally requires a minimum and unambiguous input representation for better generalization. We design our network to handle only a single hand-object pair. We mirror the other hand and run the network twice to generate predictions for both hands. This design allows us to transform input features in the space of the hand to remove ambiguity of the world coordinate, and enables us to handle different combinations of interacting hands and objects in a unified framework.

Our system can successfully synthesize a variety of finger gaits for coordinated bimanual tasks or in-hand manipulations on a diverse set of objects, such as serving tea from a tea set or turning a large torus in hand (see Fig. 1), without an exhaustive dataset. We also show a live demo to demonstrate its potential for real-time interactive applications in games and AR/VR. Comprehensive ablation studies are presented to support our design. We will publish the manipulation dataset in this work to support research on fine-grained finger gaits. We summarize our contributions as follows:

- A neural network-based motion synthesis system that can generate detailed finger motions for one-/two-hand dexterous object manipulation,
- a representation of hand-object spatial relation that enables a neural network to generalize manipulation motions to a wide range of object shapes and manipulation tasks,
- a hand-object interaction motion dataset that includes detailed finger motions and dexterous manipulations of 16 man-made objects.

2 RELATED WORK

In this section, we review research on grasp motion synthesis, dexterous object manipulation and hand motion tracking.

Synthesis of Grasping Motion. Grasp synthesis is a classic problem tackled by researchers in robotics and computer graphics.

In computer graphics, physics-based control is an approach that has been applied for synthesizing complex hand-object manipulation. Pollard and Zordan [2005] construct a finite state machine that tracks a set of reference poses for synthesizing grasping motions. Kry and Pai [2006] capture both the motion and the forces produced between the fingers and the object during grasping and use that for synthesizing grasping motion of different objects. Li et al. [2007] propose to shape-match the grasp pose and the object geometry first, and then prune the candidates based on physically based analysis.

The rise of deep learning has also resulted in a rise of techniques of data-driven grasp synthesis. In robotics, supervised learning techniques to grasp objects given static images [Lenz et al. 2015; Liu et al. 2019] or videos [Levine et al. 2016, 2018] are developed: the precision can be greatly improved by learning from a large dataset. Brahmbhatt et al. [2019] synthesize functional grasp by learning from contact demonstrations. In graphics, Taheri et al. [2020] propose a generative model for synthesizing grasps conditioned on an

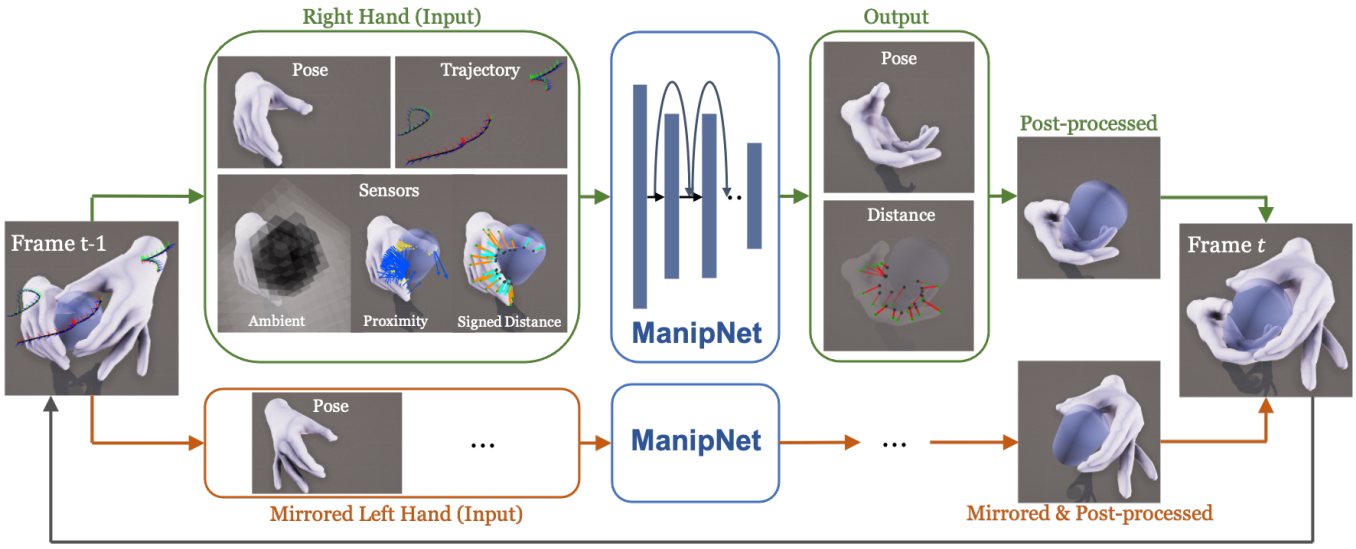


Fig. 2. The outline of our framework. Given the poses of two hands, the shapes of objects as well as the trajectories of two wrists and objects at frame $t-1$. The inputs of the two hands will be generated separately and fed into a shared neural network. Correspondingly, the poses for the two hands at frame t will be synthesized from the outputs of the neural network.

object. Karunratanakul et al. [2020] learn an implicit representation for object grasping. The idea of using a distance field to represent spatial relation between a hand and an object is actually similar to our idea. In this paper, we present that it is possible to further generalize not only grasping poses but a wide range of dynamic dexterous manipulations using our distance-based representations between a hand and object surface and virtual sensors we design to measure the distances. We thus review work related to hand-object manipulation in the following section.

Synthesis of Object Manipulation. Dexterous manipulation of objects is a rather difficult problem that requires the consideration of the dynamic interaction between the hand and the object.

Methods to manipulate objects have been developed over physically based simulation. Liu et al. [2009] compute finger motion by spacetime optimization given the initial pose and the motion of the object. Ye and Liu [2012] further extend this idea to optimize the finger contact location to produce a motion in a long horizon given a complex object motion. Mordach et al. [2012] compute finger motion by contact invariant optimization - despite only using local optimization, they can synthesize complex manipulation such as flipping pens. Zhao et al. [2013] apply particle swarm optimization to control the physics-based model to track a motion demonstrated by the user in front of a Kinect. Motions purely computed by optimization tend to lack subtle styles of human motion, which are difficult to be defined by physics-oriented constraints. We thus instead adopt a strategy to learn such subtleties from human motion.

Deep reinforcement learning (DRL) is recently being applied for controlling characters in physically based environment for hand-object manipulation movements [Andrychowicz et al. 2020; Rajeswaran et al. 2017] as well as full-body locomotion [Bergamin et al. 2019; Park et al. 2019b; Peng et al. 2018, 2017] and carrying objects [Merel et al. 2020]. One issue of DRL is that they have difficulty generalizing to different setups, such as the change in object geometry or hand morphology. Although techniques to retarget the models to characters of different body sizes are being developed for full body motion [Won and Lee 2019], such development for hand-object manipulation is yet to be explored. The representation with high generality that we develop in this paper can potentially be applied for DRL-based techniques for physically based object manipulation.

Hand Motion Tracking. A hand is surprisingly difficult to capture even with a motion capture system because of high degrees of freedom (~30 per hand), self-similarity of fingers and self-occlusion despite its small size. Thus animators often had to manually design finger motion that matches the captured full body motion, or use separate devices such as Cyberglove[1] to produce finger motion and synchronize them with the full body motion [Majkowska et al. 2006]. Thanks to the increase in the resolution of motion capture cameras and the development of techniques to track the finger joints even in the existence of occlusions [Han et al. 2018; Holden 2018], nowadays the finger motion tracking is becoming feasible, enabling data-driven approaches for synthesis of conversational hand gestures [Jörg et al. 2012; Lee et al. 2019; Wheatland et al. 2015].

Advances of commodity cameras such as Kinect [Shotton et al. 2011] and powerful deep learning algorithms have been leading to research on markerless hand motion capture from a single depth map [Tompson et al. 2014; Yuan et al. 2018] or an RGB image [Ge et al. 2019; Simon et al. 2017; Zimmermann and Brox 2017]. Real-time tracking of interacting two hands with a commodity sensor [Han et al. 2020; Mueller et al. 2019; Wang et al. 2020] has been also

[1] http://www.cyberglovesystems.com/

demonstrated recently. Tracking of object grasping and manipulation with hands, which is one of the key function of the hands still remains challenging, due to severe occlusions by an object. Ballan et al. [Ballan et al. 2012] use discriminatively learned finger salience points to predict the hand pose. Kyriazis et al. [2014] achieve tracking interactions between two hands and objects by sharing states of all objects and hands. Tzionas et al. [2016] consider physical validity to optimize poses of f hands and an object. Sridhar et al. [2016] develop a real-time tracking system of a single hand and a known object using contact point constraints and segmentation of hand parts and the object from an RGB-D image. Mueller et al. [2017] develop a real-time single hand tracking system from an egocentric RGB-D image by using synthetically created images of hands and objects. Zhang et al. [2019] develop a real-time system to capture the shape of an object grasped by a hand. Hasson et al. [2019] predict the hand pose and the object geometry from an image using a network trained with synthetic images. While these approaches, together with public datasets for hand-hand interaction [Moon et al. 2020] and hand-object interaction [Mueller et al. 2017; Tzionas et al. 2016], are actively developed, real-time tracking of two-hand grasping and manipulation of an object is still under exploration.

In summary, despite the advance in tracking techniques and machine learning priors, synthesis of dexterous hand-object manipulation by inference is still a developing area. Dexterous hand-object manipulation suffers from a great amount of occlusion and thus a significant amount of data post-processing is needed to obtain high quality motion capture data to be used for training. This is actually one of the factors that increases the difficulty of applying deep learning techniques for dexterous hand object manipulation. In this research, we provide a high quality hand-object manipulation database for partly overcoming this problem.

3 SYSTEM OVERVIEW

An overview of our system is shown in Fig. 2. It takes as input the trajectories of both wrists and the object(s) being manipulated, together with a skinned mesh of the hand and the 3D geometry of the objects. The system then generates detailed finger poses for both hands frame by frame using a deep neural network, ManipNet, as an autoregressive model. While the network itself considers only a single hand, we explain in Section 4 how our system handles different combinations of hands and objects in a unified framework.

Sensing spatial relationships. To increase the generalizability of the network and avoid overfitting to training data, we introduce in Section 5 three types of virtual sensors to encode the object geometry and its spatial relation with the hand. These sensors capture the global object shape as a coarse voxel grid and local geometric details as point samples. While the global features help to plan the overall pose and to anticipate future motion, the local features play an important role to enable generalization to variations in geometry.

ManipNet. It is a time-series model based on the residual dense network architecture. Input to the network include a hand pose in the previous frame, the sensor features, and the control signals, which include the past and future trajectories of both wrists and the object centered around the previous frame. The network then predicts the distance between the fingers and the object in addition to a new hand pose. The predicted hand pose is further processed using the predicted distance such that the spatial relations between the hand and the object are corrected. This corrected pose is then used to compute inputs to the network at the next frame. We will discuss the network in detail in Section 6.

4 RIGHT HAND-CENTRIC COORDINATE SYSTEM

Defining a coordinate system that is consistent for any motion, and for either and both hands with and without objects is surprisingly challenging. We choose to anchor the coordinate system at the right hand wrist of the previous frame in the input trajectory. The entire scene, including all the trajectories and geometries, are transformed into this coordinate system before we extract features for the network, such as the previous hand pose and the sensor features. To process the left hand, we simply mirror the whole scene, process the mirrored left hand as if it is the right hand, then mirror the output back. When there is only one object in the scene, both hands use it to compute sensor features. When each hand is handling their own object, the respective object is used. As a result, we only need to train the network with the right hand. At training time, both hand data are given so there is no bias on handedness. At inference time, we run the network on both hands to recover the entire scene, as shown in Fig. 2. Meanwhile, the handedness bias can be preserved at the inference time as the left/right hand preference will appear in the input trajectories. This coordinate choice is agnostic to the arbitrary definition of world coordinate or object local coordinate, makes no assumption about the trajectories, and handles all combinations of hands and objects in a unified framework. Based on this setup, we will focus the discussion on a single hand in the following sections.

We can alternatively work with the object local coordinate. However, there is no consistent way to define it among different objects, especially for symmetric shapes. It also doesn't apply to cases where each hand holds a different object. We may also work with a body-centric coordinate if the training data contains body motion. But we suspect the relative motion between the body and the hands could hurt performance, or will require much more training data to cover this additional source of variation.

5 SENSING THE SPATIAL RELATIONS BETWEEN HANDS AND OBJECTS

Object geometries provide crucial constraints to the hand motion. If the same input trajectories are applied to different objects, the hand needs to perform different manipulations and adapt to the object shape. We therefore design three virtual sensors to extract features that represent the spatial relation between a hand and an object. These features aim to provide sufficient information to determine the detailed finger positions, at the same time enable the network to generalize from relatively sparse training samples to a wide range of scenarios. We combine representations of global shape information with local geometric details based on proximity. While the former is helpful for the overall hand shape, the latter shares more commonalities among different object shapes and is more relevant for the manipulation task.

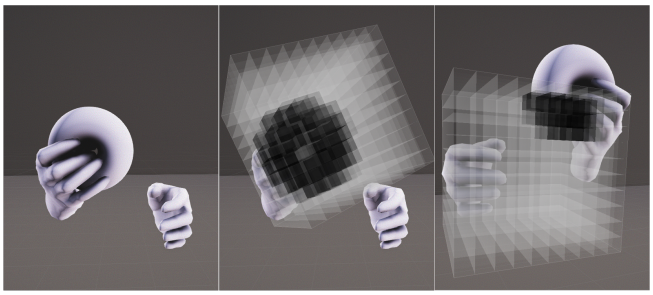


Fig. 3. Left: Right hand holding a torus. Middle: Ambient sensor values for the right hand. Right: Ambient sensor values for the right hand at a different frame. Darker the voxel color larger the occupancy value.

The three types of virtual sensors include an *Ambient* sensor that encodes the volume occupied by an object within the hand's reach. It provides a broad context to help with anticipation and long term planning. To capture local geometric variations for accurate finger placement, we place a few *Proximity* sensors on the hand surface to inform its short term movement, and a small number of *Signed Distance* sensors on the object surface to inform where to make contact on the object.

5.1 Ambient Sensor

The Ambient sensor feature is the voxel occupancy of a $18 \times 18 \times 18cm^3$ grid with a resolution of $10 \times 10 \times 10$ (see Fig. 3). The voxel grid is rigidly attached to the bone connecting the wrist and the origin of the middle finger. The center of the grid is at a constant offset ($x = 4cm, y = 0cm, z = 9cm$, see Fig. 3) from the origin of the middle finger, such that the grid center comes in front of the palm.

At each time step, we compute the overlap between the voxel grid and the object geometry to update the object occupancy value at each cell. Similar to [Starke et al. 2019], we approximate the occupancy percentage o for each individual cell as follows:

$$o = \begin{cases} 0 & \text{no intersection,} \\ 1 - \frac{d}{s} & \text{intersection,} \end{cases}$$

where $d \geq 0$ is the distance from the cell center to the object, and $s = 1.8cm$ is the cell edge length. The Ambient sensor perceives the size and rough shape of the object within the hand's reach. Such global information is helpful for shaping the pre-grasping behavior of the hand, as the grasp aperture depends on both the object size and distance. It can also inform a finger gaiting plan for future finger placements. Although increasing the voxel resolution can help it acquire more detailed geometry, the memory/computation costs increase in cubic order. Moreover, with limited training objects, the network may overfit to their distinct occupancy patterns. We therefore use a coarse voxel grid in the Ambient sensor, and use distance-based sensors as described below to capture more fine-grained geometric details.

5.2 Proximity Sensor

The Proximity sensors are distributed across the hand's surface to sense the closest object surface for a given hand pose. We sample

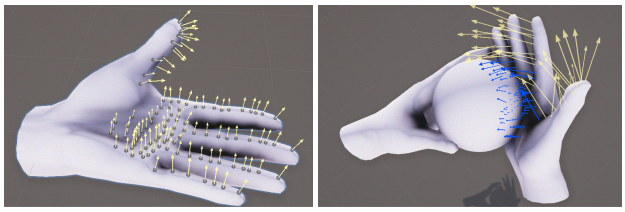


Fig. 4. Left: The 104 Proximity sensors on the hand mesh. Right: Proximity Sensors cast rays along the hand surface normal until they hit the object surface (blue arrows), or at a maximum distance (yellow arrows).

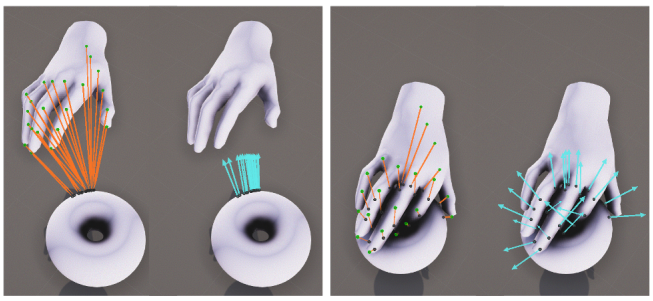


Fig. 5. Two examples of the Signed Distance sensors for the right hand. The hand joints are shown in green. Orange lines indicate the distance from the hand joints to the torus. Cyan arrows are surface normals on the torus.

them uniformly from the hand mesh vertices on the palm side (see Fig. 4 left). There are a total of 104 sensors to cover all the finger segments and the palm area where contacts most likely occur. These sensors cast rays along the normal direction on the hand mesh until they hit the object surface or reach the distance threshold $\delta_{\max} = 10cm$. We reverse the ray direction if a sensor is already inside the object. For a sensor j, the distance feature d_j is computed as:

$$d_j = \begin{cases} sign(\mathbf{p}_j) \left\| \mathbf{p}_j - \mathbf{p}_c \right\| & \text{ray-object intersection,} \\ sign(\mathbf{p}_j)\delta_{\max} & \text{no intersection,} \end{cases}$$

where $\mathbf{p}_j$ is the sensor position on the hand, and $\mathbf{p}_c$ is the intersection point on the object surface. $sign(\mathbf{p}_j)$ is positive if $\mathbf{p}_j$ is outside the object, and negative if inside. We collect values from each sensor and construct a feature vector $\mathbf{d} = \{d_0, ..., d_{103}\}$. The Proximity sensors capture the object surface details in close proximity to the hand. These distances can inform the hand where contacts are going to occur or where to avoid penetration in the intermediate future. As the sensing direction is invariant to the object shape, the proximity sensors are robust to concave features on the object.

5.3 Signed Distance Sensor

The Signed Distance sensors sample the object's signed distance field at each of the 22 finger joints. They consist of both the closest distance value and the object's surface normal direction (see Fig. 5). Specifically, features $\mathbf{s}_j$ at a hand joint j is computed as

$$\mathbf{s}_j = \left\{ sign(\mathbf{p}_j) \min\left(\left\| \mathbf{p}_j - \mathbf{p}_o \right\|, \delta_{\max} \right), \mathbf{n}_o \right\}$$

where $\mathbf{p}_j$ is the joint position, and $\mathbf{p}_o$ is its closest point on the object surface. $\mathbf{n}_o$ is the surface normal at $\mathbf{p}_o$. We collect all the features as $\mathbf{s} = \{\mathbf{s}_0, ..., \mathbf{s}_{21}\}$. The signed distance field of an object is pre-computed for efficient distance look up. However, the signed distance field is limited in resolution. To acquire more accurate results here, we use it to help find the closest surface point, then compute the exact distance and normal. Since this is more expensive, we only compute them at the hand joints.

The Signed Distance sensors encode object surface details under the influence of the hand grasp aperture in close proximity. The normal directions provide useful guidance on where to make contacts on the object. We believe surface normals are important features to consider for a physically valid grasp. We deliberately separate the distance representation from the normals, so the network can explicitly manipulate them to produce output distance in an autoregressive fashion.

6 MANIPNET FOR HAND MOTION SYNTHESIS

In this section, we introduce ManipNet for synthesizing dexterous hand manipulation. We will first explain the input and output of the network. We then describe in detail its architecture based on residual dense blocks [Zhang et al. 2018b], and finally implementation details on post-processing and training.

6.1 Network Input and Output

The network regresses a hand pose and hand-object distances from the previous pose, the hand and object trajectories, and sensor features.

Input $\mathbf{X}_t = \{\mathbf{P}_{t-1}, \mathbf{T}_{t-1}, \mathbf{S}_{t-1}\}$ at frame t is composed of the pose input $\mathbf{P}_{t-1}$, trajectory input $\mathbf{T}_{t-1}$, and sensor input $\mathbf{S}_{t-1}$, all extracted at frame $t-1$.

- **Pose input** $\mathbf{P}_{t-1} = \left\{\mathbf{j}_{t-1}^{pos}, \mathbf{j}_{t-1}^{rot}\right\}$ is composed of the joint positions $\mathbf{j}_{t-1}^{pos} \in \mathbb{R}^{3n}$ and their orientations $\mathbf{j}_{t-1}^{rot} \in \mathbb{R}^{6n}$, expressed in its wrist's coordinate at frame $t-1$. The orientations are represented by two orthogonal axes defined in the local coordinate system of each bone as in [Zhang et al. 2018a]. $n = 22$ is the number of joints in one hand (see figure on the right). As mentioned in Section 4, our network only considers the right hand pose. For the left hand, we provide its wrist trajectory for additional context as described below. This enables each hand to make predictions independent of each other. We found that including the left hand pose in the input will easily lead to overfitting to training hand poses.

- **Trajectory input** $\mathbf{T}_{t-1} = \left\{\mathbf{T}_{t-1}^{right}, \mathbf{T}_{t-1}^{left}, \mathbf{T}_{t-1}^{obj}\right\}$ is composed of trajectories of the right wrist $\mathbf{T}_{t-1}^{right}$, left wrist $\mathbf{T}_{t-1}^{left}$, and the object $\mathbf{T}_{t-1}^{obj}$. Trajectory features are commonly used as control signals for character animation [Holden et al. 2017; Zhang et al. 2018a], as they provide helpful context of the desired activity. We consider trajectories of the hands and the object(s) centered at frame $t-1$, with a total duration of one second. We uniformly sample 10 frames in the past and 10 frames in the future to arrive at 21 frames on each trajectory.
  - $\mathbf{T}_{t-1}^{right} = \left\{\tau_{t-1}^{pos}, \tau_{t-1}^{rot}\right\}$ is composed of the sampled right wrist positions $\tau_{t-1}^{pos} \in \mathbb{R}^{3\times21}$ and orientations $\tau_{t-1}^{rot} \in \mathbb{R}^{6\times21}$. They are expressed relative to the right wrist coordinate at frame $t-1$.
  - $\mathbf{T}_{t-1}^{left} = \left\{\tau_{t-1}^{dis}\right\}$, where $\tau_{t-1}^{dis} \in \mathbb{R}^{21}$ is the distance from the left hand wrist position in each sample frame to the right hand wrist position at frame $t-1$. We represent the left hand trajectory only by distance to reduce the input feature dimension. Higher input dimension will require more training data to avoid overfitting.
  - $\mathbf{T}_{t-1}^{obj} = \left\{\tau_{t-1}^{com}, \tau_{t-1}^{av}\right\}$ is composed of the positions of the object's center of mass $\tau_{t-1}^{com} \in \mathbb{R}^{3\times21}$ and the angular velocity $\tau_{t-1}^{av} \in \mathbb{R}^{3\times21}$, all expressed in the right wrist coordinate at frame $t-1$. We choose angular velocity as a feature because it can be defined consistently for an object, regardless of its local coordinate. We find it especially helpful for representing objects with symmetry.

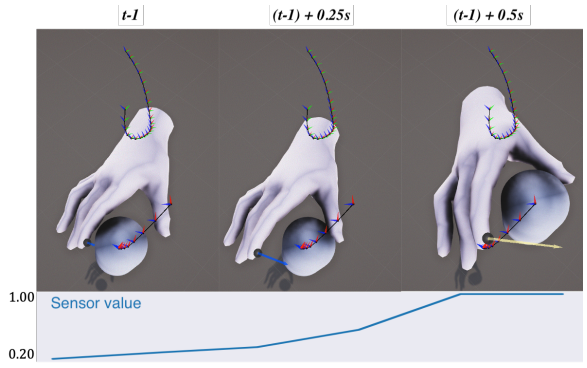


Fig. 6. We compute sensor values for future frames using the hand pose at frame $t-1$. Top: One proximity sensor is shown here at three frame samples. Bottom: Distance values from the sensor.

- **Sensor input** $\mathbf{S}_{t-1} = \left\{\mathbf{S}_{t-1}^{amb}, \mathbf{S}_{t-1}^{pro}, \mathbf{S}_{t-1}^{sd}\right\}$ is composed of the three types of sensor features as described in Section 5. The Ambient sensor input $\mathbf{S}_{t-1}^{amb} \in \mathbb{R}^{1000}$ is the flattened object occupancy feature. For the Proximity sensor and the Signed Distance sensor, we not only compute features at frame $t-1$, but also sample 5 additional frames along the 0.5 second future trajectories. Because we don't know the future hand poses, we use the hand pose at $t-1$ instead (see Fig. 6). These future sensors reveal what would happen to the hand-object relationship if the hand pose were to remain static, as to help better prepare for the future. Therefore, the Proximity sensor input $\mathbf{S}_{t-1}^{pro} \in \mathbb{R}^{104\times6}$ consists of distance features of 6 frames in total. For the Signed Distance sensors, we compute the distance features at future frames, and the normal features only at frame $t-1$ to reduce input size, to arrive at $\mathbf{S}_{t-1}^{sd} \in \mathbb{R}^{22\times6+3\times22}$.

In summary, there are 2356 input features to the network, including $\mathbf{P}_{t-1} \in \mathbb{R}^{198}$, $\mathbf{T}_{t-1} \in \mathbb{R}^{336}$, and $\mathbf{S}_{t-1} \in \mathbb{R}^{1822}$.

Output $\mathbf{Y}_t = \{\mathbf{P}_t, \mathbf{d}_t\}$ is composed of the predicted pose $\mathbf{P}_t \in \mathbb{R}^{198}$, and the predicted distance from the 22 hand joints to the object surface, $\mathbf{d}_t \in \mathbb{R}^{22}$.

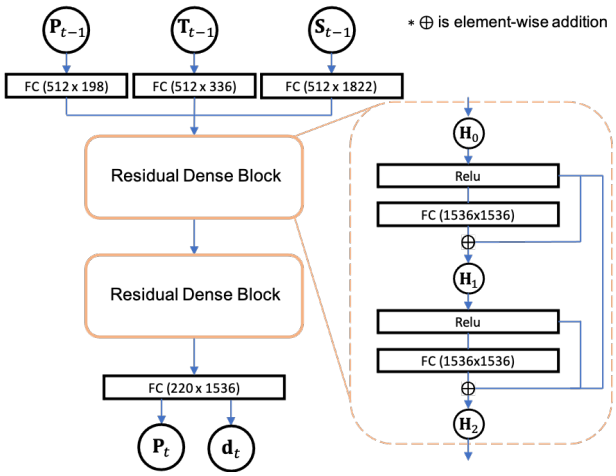


Fig. 7. The diagram of our network consisting of encoders for three categories of inputs and two residual dense blocks with the same structure. FC are fully-connected layers.

6.2 Network Architecture

An overview of the ManipNet architecture is shown in Fig. 7. The network first encodes each set of input features into a 512 dimensional vector respectively, then passes the concatenated features through two residual dense blocks, and finally decodes the resulting features into the output. Both the encoders and the decoder are fully-connected layers. The network is trained in a supervised setting by minimizing the mean squared error (MSE) between the output $\mathbf{Y}_t$ and the ground truth $\hat{\mathbf{Y}}_t$.

Residual Dense Block: We now describe the details inside each residual dense block [Zhang et al. 2018b]. The input to a residual dense block is denoted by $\mathbf{H}_0$. Two residual connection operations $\oplus$ happen inside a residual dense block. The operation of each block can be written as follows:

$$\begin{aligned} \mathbf{H}_1 &= \mathbf{W}_0\ \mathbf{RELU}(\mathbf{H}_0) + \mathbf{b}_0 + \mathbf{RELU}(\mathbf{H}_0) \\ \mathbf{H}_2 &= \mathbf{W}_1\ \mathbf{RELU}(\mathbf{H}_1) + \mathbf{b}_1 + \mathbf{RELU}(\mathbf{H}_1) + \mathbf{RELU}(\mathbf{H}_0) \end{aligned}$$

where $\mathbf{H}_1, \mathbf{H}_2$ are the outputs of the first and second dense layers and $\mathbf{W}_0 \in \mathbb{R}^{1536\times1536}$, $\mathbf{b}_0 \in \mathbb{R}^{1536}$, $\mathbf{W}_1 \in \mathbb{R}^{1536\times1536}$, and $\mathbf{b}_1 \in \mathbb{R}^{1536}$ are the parameters of each layer.

The dense residual connections are preferable for our problem because the network input and output contain the same information at two consecutive frames. Instead of synthesizing the output from scratch, it'd be easier for the network to make adjustments to the input. We compare dense residual connections with residual connections [He et al. 2016] and no residual connections qualitatively and quantitatively to demonstrate its advantage in Section 8.

6.3 Post-processing

The network predicts smooth and realistic hand poses in similar styles to the training data, especially for objects used in training. However, the hand pose may have penetration artifacts for novel objects or motions that are not seen in training. We rely on $\mathbf{d}_t$, the predicted distances between the finger joints and the object surface, to enforce hand-object contacts more explicitly. Specifically, for each joint, we first calculate a target position by combining the network-predicted joint position and distance inside the object signed distance field (see algorithm in Appendix B). These target joint positions will then be passed into an inverse kinematics process with the corresponding joint limits to edit the predicted hand poses. As a compromise of speed and output quality, we adopt a CCD-based IK solver [Aristidou and Lasenby 2011] with implementation from [Starke et al. 2019], which runs much faster than optimization-based alternatives.

6.4 Implementation details

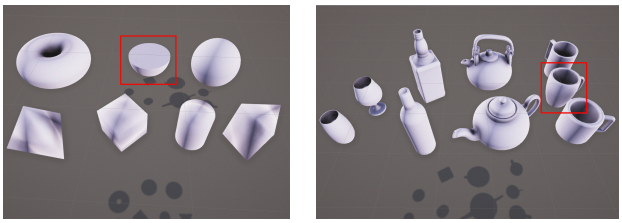
We now describe implementation details in setting up network training and inference.

During network training, we use ground truth data to compute both the training input ($\mathbf{X}_t$) and the ground truth output ($\hat{\mathbf{Y}}_t$). Sensor features are computed from the ground truth hand pose at frame $t-1$. The ground truth output distances are computed from ground truth hand pose at frame t, and clamped to $[0.0, 3.5]cm$. The predicted output distances are clamped to be non-negative. We limit the range of the distances so the network can focus on when the hands are close to the object. This training setting is per-frame based with no recurrent component. We double the training set with data mirroring, so the network does not have a handedness bias. Besides mirroring, we do not do any other data augmentation or randomization.

During inference, we assume the trajectory input is obtained from some motion tracking solution, and the pose computed at frame $t-1$ is used to compute network input at frame t. The output pose is updated by post-processing before feeding back to the network as input. We initiate the hand motion from the neutral pose, and start inference before the hands come into contact with the objects. In a live setting, we start pose prediction after a 0.5 second delay to collect future input trajectories. Notably, even without data randomization in training, we do not observe any drift or regress-to-mean behavior at inference time.

7 EXPERIMENTAL RESULTS

In this section, we first describe the data collection process and details of the dataset. We next describe the training process and present experimental results. Please refer to the supplementary video for visual validation.



(a) Primitive objects (b) Dining table objects

Fig. 8. Objects in our dataset with held-out objects highlighted.

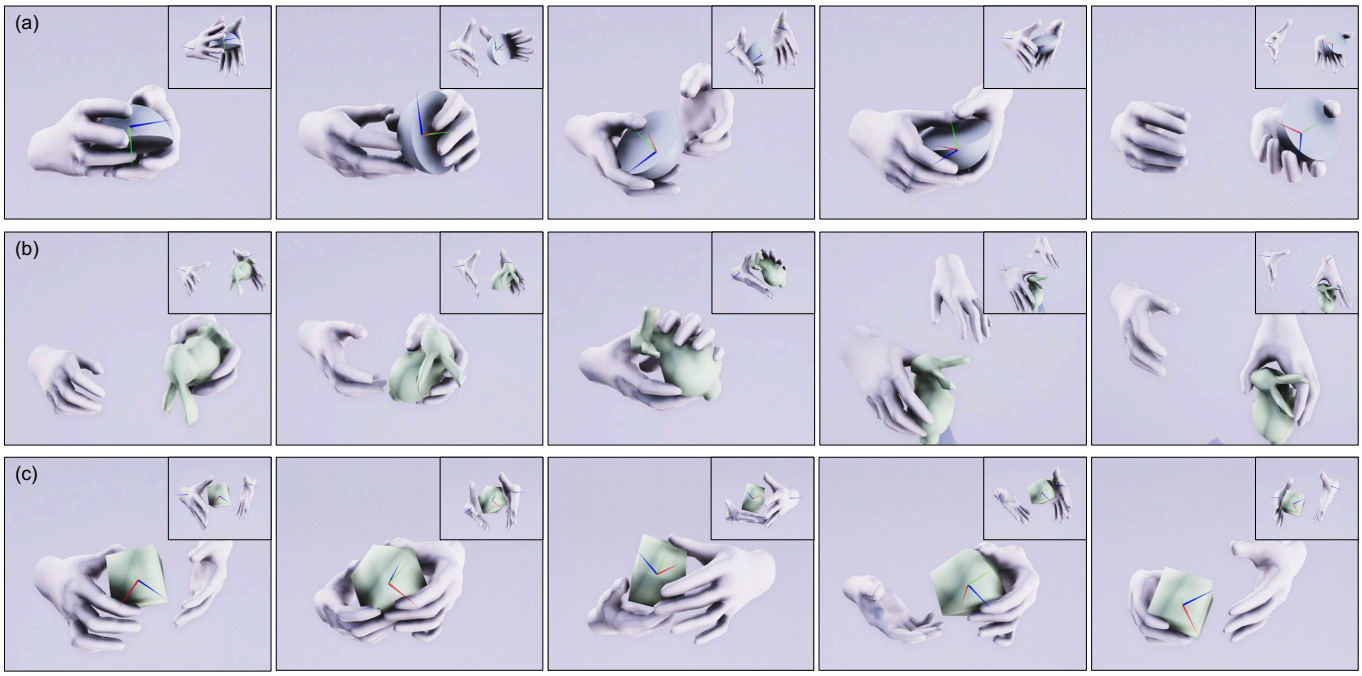


Fig. 9. Results of the proposed method with (a) hemisphere, (b) bunny and (c) cube objects.

Motion Capture Setup. We capture our dataset using the deep label system [Han et al. 2018]. We used an *Optitrack* motion capture system with 16 cameras at 60 frames per second, and gloves with 19 markers per hand for finger tracking. Objects are tracked as rigid bodies using marker clusters. To have an accurate correspondence between the virtual and physical objects, we 3D-printed the object models and manually calibrated the marker clusters attached on each object. We capture both single-handed and two-handed manipulation motions.

Object Manipulation Dataset. Our dataset includes dexterous manipulations of two types of objects: primitive shapes and dining table drink ware. The motions are all performed by a single subject.

The primitive object dataset consists of basic shapes shown in Fig. 8(a). We use 58, 300 frames for training, and leave out the hemisphere object and a total of 28, 000 frames for testing. Our goal is to capture various free-form dexterous manipulation behaviors including grasping, spinning, re-grasping, passing between hands and putting down, as well as transitions between these behaviors. The hand motions are mostly constrained by the object geometry. We expect a model trained from these primitive shapes can generalize to more complex shapes, because of their commonalities in local geometric details.

The dining table object dataset consists of drinkware as shown in Fig. 8(b). We include motions such as pouring from teapots or wine bottles, drinking from a cup, as well as casually fiddling with these objects. 90, 880 frames are used for training, and 32, 000 frames for testing, including all 8, 000 frames where each hand is manipulating a different object. The goal of this dataset is to demonstrate functional manipulations of more complex shapes. We expect to see generalization from this dataset to richer behaviors on everyday objects, especially those with handles and concavities.

Training. The network is trained in Tensorflow [Abadi et al. 2016] using the Adam optimizer [Kingma and Ba 2014], with a batch size of 32 and a learning rate of 0.0001. Dropout [Srivastava et al. 2014] is applied with a retention probability of 0.7 to avoid overfitting. Training is performed for 500 epochs for the primitive object dataset and takes around 1.5 hours with GeForce GTX 2080 GPU. The dining table object dataset takes around 2.5 hours with the same setting.

Results. We now present experimental results from our system. We refer readers to the supplementary video for visual evaluation.

Separate networks are trained on each of the two datasets, denoted as ManipNet_P and ManipNet_D. We first apply ManipNet_P on a held-out trajectory of the torus (seen in training) and on the held-out object hemisphere. We do not expect the output to replicate the captured finger motions on these novel input, but expect visually plausible manipulations in agreement with the object geometries and motions. Indeed, when turning a large torus in hand (see Fig. 1(b)), we see non-trivial finger gaits with rewinds and substitutions [Han and Trinkle 1998; Hong et al. 1990], carefully orchestrated across the torus' surface. The doughnut hole is opportunistically utilized to stabilize the grip. When passing the hemisphere between hands (see Fig. 9(a)), all ten fingers naturally conform to either the flat side or the round side with coordinated movements, even though this combination of flat and round surfaces is novel. We also test scaled versions of the cylinder (seen in training) on a held-out trajectory (see Fig. 10). The hands have no problem adapting to this novel task, where the pregrasping apertures are properly adjusted for the

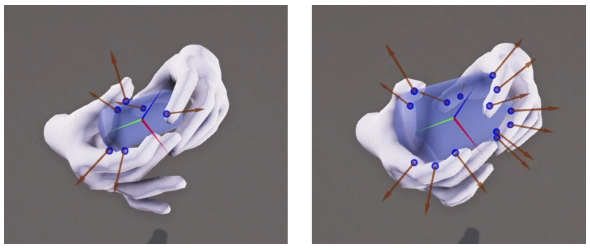


Fig. 10. The held-out trajectory applied to cylinders of different scales.

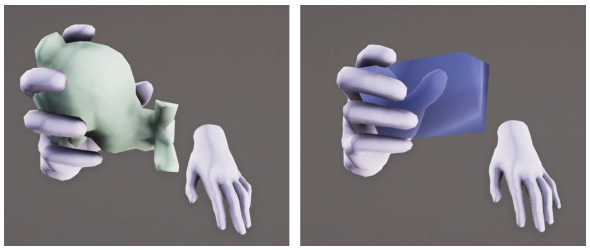


Fig. 11. The same input applied to two different shapes.

different sizes. We can also apply the same unseen trajectory to drastically different object shapes, such as a bunny (see Fig. 9(b)) and a triangular prism, and observe distinct motions consistent with the respective object (see Fig. 11). Finally, an example of manipulating a cube is produced with a held-out trajectory, where the system produces a sequence of rotation and regrasping motion (see Fig. 9(c)).

We conduct similar experiments with ManipNet$_D$. We successfully generate a new tea pouring motion that requires the hands to manipulate their own objects, grasp on handles, and make small pregrasping adjustments (see Fig. 1(a)). We also test new teapot and cup models not seen in training, as variations in tea sets are practically countless. The resulting motions include functional activities such as pouring (see Fig. 12) and drinking, as well as transitioning between hands and collaborating for support and finger adjustments. The fingers are able to grasp a cup on its body by going through the handle, or directly grabbing the handle.

Finally, we show a live demo where a user manipulates an object not seen during training in an interactive motion capture session. Motions of the wrists and the object are tracked. The finger motions are synthesized in real time with a 0.5 second delay due to the requirement of future trajectory input. We can substitute the real object with different novel virtual objects on-the-fly, such as a hammer, a wine bottle, a bunny, a pig, and a dolphin etc. (see Fig. 13). The user can manipulate the real object to accomplish purposeful tasks, or just to fiddle with it for fun. Our system automatically synthesizes natural and realistic finger movements appropriate for the corresponding virtual objects.

Failure Cases. Our feature representation is limited in spatial resolution as both the voxel grid and the distance sensors are sparse. It is therefore easy to miss small features on the object. For example, as shown in the right, the thumb penetrates the thin handle of the cup because it is too small. As a mitigation, we could adopt an adaptive-resolution representation to balance between run-time and accuracy. Other common failures are due to inaccurate predictions of the hand pose and distances. They manifest as either penetrations or the lack of desirable contacts.

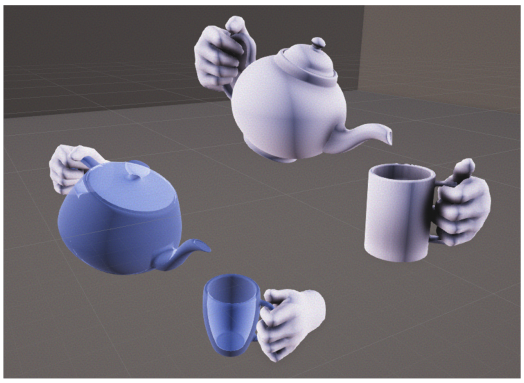


Fig. 12. A snapshot of a scene where the Utah teapot is used to pour into a novel mug cup. The original scene is shown for reference.

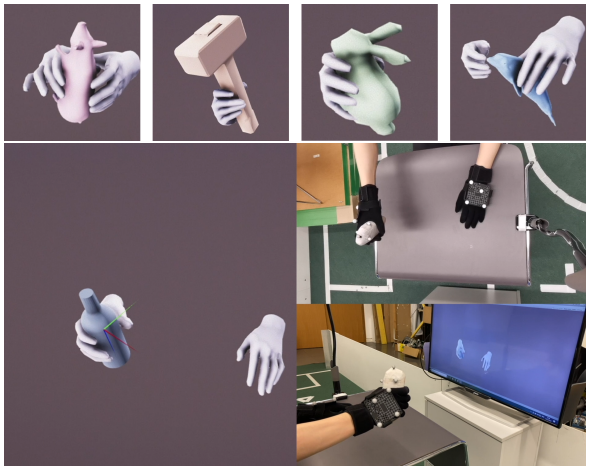


Fig. 13. (top) Scenes where various 3D objects manipulated at the online demo. (bottom) The live demo setup.

8 EVALUATION

We first describe three evaluation metrics, then present ablation studies against them to validate our design.

Evaluation Metrics. The evaluation of our system is inherently visual since we are generating new animations not seen in any data. And because our network is a regression function (rather than sampling from a distribution as in common generative models), the output motion naturally follows the same style as the training data. In our case, artifacts reveal themselves as finger-object penetrations

Fig. 14. The snapshots of the ablation study where the proximity sensor (left), both the signed distance and proximity sensors (middle) and the future sensing (right) are disabled.

Fig. 15. The snapshots of the ablation study where the architecture is switched to MLP (left) and Resnet (right).

or inconsistent finger movements with the object motion. In addition, post-processing could introduce noise and jitter. We therefore use finger *penetration*, *physical realism* of the manipulation, and finger motion *smoothness* as quantitative evaluation metrics.

- **Penetration** We approximate the hand mesh by a collection of capsules for collision detection, and compute the depth of penetration between the hand and the object at every frame. We expect a small penetration at the site of contact even in the ground truth data, due to soft finger deformations. But large penetrations (ie. $\geq 1.5cm$) indicate a non-physical pose.
- **Physical realism** We evaluate whether the object's motion can be explained by frictional contacts from the hand as a metric of physical realism. We solve for contact forces applied to the object as explained in Appendix A, on frames without large penetrations. We then count the percentage of frames where this inverse dynamics problem is feasible.
- **Smoothness** The acceleration of the 22 finger joint positions is computed as a metric for smoothness. It indicates whether the motion is jittery, as may be induced by noisy network output or post-processing. Meanwhile, the smoothness can illustrate the temporal coherence achieved under different settings by comparing against the ground truth data.

These three metrics need to be jointly considered to evaluate the quality of a motion.

Ablation Study. We conducted ablation studies of each component of the system and present the statistics in Table 1. We compare the physical realism and penetration metrics with and without inverse kinematics (IK) in post-process. Without IK, we can better compare the network output between different training settings. With the help of distance prediction, IK is extremely useful in fixing severe penetration artifacts and making contact, therefore greatly improves the physical realism of the result. More importantly, the processed pose is fed back to the network as input, so errors do not accumulate over time. However, large edits to the output pose will lead to unpleasant jittery motion. We therefore also present the smoothness metric of IK results to demonstrate this trade-off.

We experimented with ManipNet$_P$ in two scenarios: generalization to a new trajectory unseen in training with a known object (cylinder), and generalization to a held-out object (hemisphere). The new object presents a greater challenge to the network as its performance degrades quite significantly. Thanks to the distance prediction and IK, we can still produce high quality results.

We study the effect of virtual sensors by removing them from the system. If we remove all sensor information and only use the trajectories as input, we see about a 45% drop in physical realism without IK ("no sensors" in Table 1). In this case, the network has no knowledge of the object shape and won't be able to adapt the finger pose accordingly. IK is not always successful in improving penetration without introducing a lot of noise in the motion. We also test removing only the ambient sensor ("no ambient" in Table 1) or removing both the proximity sensor and the signed distance sensor ("no distance" in Table 1). The former deprives the network of global shape information so the grasping behavior is only influenced by the closest shape details. The latter only supplies the network with coarse shape information, so it cannot handle fine details in the

Table 1. Results of the ablation study when each module/sensor is removed from the system, tested with objects seen during training (top) and unseen during training (bottom). The physics show the ratio of frames that the interaction is physically plausible, with/without IK. The penetration is the average penetration per frame and collider, with/without IK. When all geometry sensors are turned off, the system predicts the finger motions only from the wrist and object trajectories. Smoothness is the acceleration of the finger joints per frame.

Seen object (cylinder), unseen trajectory

	with IK phy.(%) / pen.(cm)	without IK phy.(%) / pen.(cm)	smoothness (cm/s^2)
raw data	95.63 / 0.2680	- / -	109.5
ours	**94.75** / **0.3485**	**88.00** / **0.4678**	**109.8**
no sensors	89.50 / 0.4016	49.75 / 0.7372	143.5
no ambient	91.75 / 0.3644	64.50 / 0.5550	117.3
no distance	89.75 / 0.4023	81.75 / 0.5050	131.2
short traj.	94.25 / 0.3607	79.25 / 0.5254	119.2
ResNet	81.75 / 0.4100	77.00 / 0.5014	126.8
MLP	83.25 / 0.4283	46.25 / 0.7301	200.2

Unseen object (hemisphere), unseen trajectory

	with IK phy.(%) / pen.(cm)	without IK phy.(%) / pen.(cm)	smoothness (cm/s^2)
raw data	92.19 / 0.2957	- / -	78.22
ours	**93.75** / **0.4175**	**66.50** / **0.6150**	**74.25**
no sensors	74.75 / 0.4682	45.00 / 0.7206	104.3
no ambient	93.25 / 0.4497	51.00 / 0.7509	84.12
no distance	92.00 / 0.4292	47.50 / 0.6887	93.09
short traj.	90.25 / 0.4348	61.75 / 0.6240	88.65
ResNet	91.50 / 0.4723	61.75 / 0.6599	89.83
MLP	85.50 / 0.4919	59.25 / 0.6171	170.5

geometry. We see degradation of metrics in both cases either with or without IK. Visually, we observe more penetrations or absence of desirable contacts when we leave out any type of sensors, including distances from future frames (see Fig. 14). Each virtual sensor provides unique information in different scenarios, so we get the best results by combining them.

Future trajectories are important for pre-grasp shaping and planning of finger gaits, at the expense of long delays in real time applications. This is a trade-off each application has to consider. For reference, we show the metrics of using a shorter future trajectory ("short traj." in Table 1), where it is shortened from 0.5 second to 0.1 second, corresponding to two sample frames of trajectories and one sample frame of future distances.

Finally, we study the effect of dense residual connections, by comparing with residual connections only (i.e., ResNet, removing the outer residual connection inside the block in Fig. 7) and no residual connections (i.e., MLP). As expected, more residual connections lead to better results, as more information is carried from the input to the output (see Fig. 15). Training curves for different network structures are shown in Fig. 16.

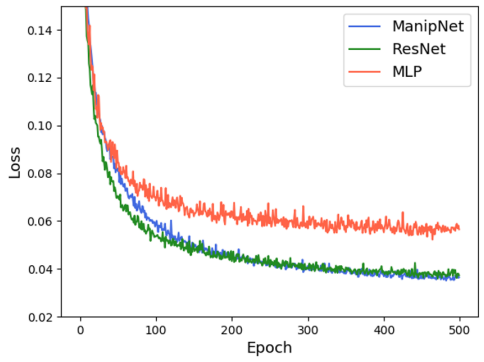


Fig. 16. Training curves of ManipNet, ResNet and MLP.

9 DISCUSSIONS

We presented a neural network formulation to synthesize finger motions of object manipulation using both hands. Our key contribution is features that represent the spatial relation between the hand and the object in a manipulation. They proved to be crucial for the network to generalize from a small dataset to new shapes and new motions, as supported by quantitative ablation studies. We also demonstrated their efficacy in a live demo of casually fiddling with different virtual toy animals, and in activities such as serving tea from tea sets of various shapes and sizes.

Our network can serve as an autoregressive model even though it is trained from ground truth data without any special training routines or data randomization. We believe this can be attributed both to the IK process that prevents error accumulation, and to our choice of input features that effectively discriminate the control signals without overfitting. IK also plays an important role to improve physical realism in novel scenarios. Nonetheless, better network predictions are always valuable as they provide IK with a better initial pose and more accurate distance targets. Notably, our results also affirm the finding from Ye and Liu [2012] that the object shape and the relative motion between the hand and the object together provide sufficient information about the manipulation. We observe that even the relative motion alone encodes distinctive features of the object's affordance. It enables generalization to new motions of the same object to a certain degree, and motions of an object doesn't directly apply to arbitrary objects. We hope our findings above will inspire new ideas in related areas.

10 LIMITATIONS AND FUTURE WORK

Data variation. We are encouraged by what the network can learn from a small dataset, but this by no means implies our model would not benefit from a large variety of high quality hand-object interaction data, such as the GRAB dataset [Taheri et al. 2020]. It will allow us to expand support to multiple hand models and different motion styles of performing the same task. Especially, the lack of certain type of objects in our dataset hinders our model to generalize to tiny or intricate objects. It could be interesting to explore more precise motion capture for such scenarios. Another potential direction is to condition the motion generation on a parametric hand model such as MANO [Romero et al. 2017]. Similarly, conditional generative models like MoGlow [Henter et al. 2020] are promising for stochastic and stylized motions [Alexanderson et al. 2020].

Physics plausibility. Although our results look natural in general, physical realism only comes from training data distribution and is not strictly enforced. It would be interesting to explicitly encode physical constraints into the neural network formulation. We can also post-process our results in a physical simulation, for example as a reference to DeepMimic-type [Peng et al. 2018] of motion controllers. Moreover, our single hand based coordinate cannot handle an intense interactions between the two hands, such as interleaving fingers when holding a cup. A fruitful future direction is to unify single-hand, two-hand, and hand-object interaction synthesis under the same framework, by investigating more general and learnable representation of geometric features [Park et al. 2019a].

Synthetic data capture. Our system can be an interesting source of synthetic data for training neural networks, as realistic hand-object interaction data is still expensive to collect. Our system can also be a useful alternative to finger motion capture for VFX or game production, with the convenience of substituting different objects in post-production, and without the additional hassle of finger tracking hardware. Animators can use our system to iterate quickly on complex finger animations involving objects. A helpful addition to our system would then be to take hand-animated trajectories as input and output modified trajectories similar to the training data.

Real-world application. Alternatively, capturing the required input trajectories are actually quite accessible at home. Many commercial or open source solutions are available to track rigid object trajectories and hand trajectories from a smart phone [Apple 2021; Facebook 2021; Google 2021]. Moreover, as AR/VR hardware is rising in the consumer market, incorporating our system with their builtin object tracking [Oculus 2021; Vive 2021] and hand tracking [Han et al. 2020; Leap 2021; Microsoft 2021; Ultraleap 2021] solutions will

open up many creative opportunities to new interactive content. Unfortunately, we need to mitigate the latency from future trajectories to deliver a satisfactory experience. A potential solution is to further infer the future trajectories based on high level goals and the past motion, as done in [Starke et al. 2019], or to combine with a motion planer learned from video or real-world demos [Wang et al. 2019]. As our system requires the object geometry as input, on-line mesh reconstruction from the camera data can improve the applicability of our system.

Geometry representation. Geometric features play an important role when humans interact with unseen objects. Given the recent progress in object saliency [Lau et al. 2016] and functionality [Hu et al. 2018, 2020; Pirk et al. 2017] analysis, one compelling future direction can be to use features extracted by such models as input to our system to plan the movements based on the object's saliency and functionality.

Full body interaction. Finally, it will be interesting to integrate our system as a module in an embodied AI agent to synthesize complex full body interaction actions in a realistic environment, such as arranging items on a shelf, retrieving toys from a toybox, or cooking food by adding and mixing ingredients [Batra et al. 2020].

ACKNOWLEDGMENTS

Taku Komura is partly supported by a start-up fund by The University of Hong Kong (182DRTAKU, 187FRTAKU, 230DRTAKU). This project is funded by Facebook Reality Labs (Project 5961133: Oculus - Hand-Object Interaction).

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Savannah, GA, USA) *(OSDI'16)*. USENIX Association, Berkeley, CA, USA, 265–283. http://dl.acm.org/citation.cfm?id=3026877.3026899

Simon Alexanderson, Gustav Eje Henter, Taras Kucherenko, and Jonas Beskow. 2020. Style-Controllable Speech-Driven Gesture Synthesis Using Normalising Flows. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 487–496.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* 39, 1 (2020), 3–20.

Apple. [Online; accessed 27-January-2021]. Augmented Reality: Introducing ARKit 4. https://developer.apple.com/augmented-reality/arkit/.

Andreas Aristidou and Joan Lasenby. 2011. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. *Graphical Models* 73, 5 (2011), 243–260.

Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision*. Springer, 640–653.

David Barraff. 1997. An introduction to physically based modeling: Rigid body simulation I - unconstrained rigid body dynamics. In *ACM SIGGRAPH Courses*.

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–11.

Samarth Brahmbhatt, Ankur Handa, James Hays, and Dieter Fox. 2019. Contactgrasp: Functional multi-finger grasp synthesis from contact. *arXiv preprint arXiv:1904.03754* (2019).

Samarth Brahmbhatt, Chengcheng Tang, Christopher D Twigg, Charles C Kemp, and James Hays. 2020. ContactPose: A dataset of grasps with object contact and hand pose. *arXiv preprint arXiv:2007.09545* (2020).

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. 2019. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 8973–8979.

Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. 2017. Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Facebook. [Online; accessed 27-January-2021]. Spark AR Studio. https://sparkar.facebook.com/ar-studio/.

Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 2019. 3d hand shape and pose estimation from a single rgb image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10833–10842.

Google. [Online; accessed 27-January-2021]. MediaPipe. https://google.github.io/mediapipe/.

Li Han and Jeffrey C Trinkle. 1998. Dextrous manipulation by rolling and finger gaiting. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, Vol. 1. IEEE, 730–735.

Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D. Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, Asaf Nitzan, Gang Dong, Yuting Ye, Lingling Tao, Chengde Wan, and Robert Wang. 2020. MEgATrack: Monochrome Egocentric Articulated Hand-Tracking for Virtual Reality. *ACM Transactions on Graphics (TOG)* 39, 4 (2020).

Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D Twigg, and Kenrick Kin. 2018. Online optical marker-based hand tracking with deep labels. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10.

Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. 2019. Learning Joint Reconstruction of Hands and Manipulated Objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. 2020. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–14.

Daniel Holden. 2018. Robust solving of optical motion capture data by denoising. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Jiawei Hong, Gerardo Lafferriere, Bhubaneswar Mishra, and Xiaonan Tan. 1990. Fine manipulation with multifinger hands. In *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, 1568–1573.

Ruizhen Hu, Manolis Savva, and Oliver van Kaick. 2018. Functionality representations and applications for shape analysis. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 603–624.

Ruizhen Hu, Zihao Yan, Jingwen Zhang, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. 2020. Predictive and generative neural networks for object functionality. *arXiv preprint arXiv:2006.15520* (2020).

Nathanaël Jarrassé, Adriano Tacilo Ribeiro, Anis Sahbani, Wael Bachta, and Agnes Roby-Brami. 2014. Analysis of hand synergies in healthy subjects during bimanual manipulation of various objects. *Journal of neuroengineering and rehabilitation* 11, 1 (2014), 1–11.

Sophie Jörg, Jessica Hodgins, and Alla Safonova. 2012. Data-driven finger motion synthesis for gesturing characters. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–7.

Korrawe Karunratanakul, Jinlong Yang, Yan Zhang, Michael Black, Krikamol Muandet, and Siyu Tang. 2020. Grasping Field: Learning Implicit Representations for Human Grasps. In *International Conference on 3D Vision (3DV)*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Paul G Kry and Dinesh K Pai. 2006. Interaction capture and synthesis. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 872–880.

Nikolaos Kyriazis and Antonis Argyros. 2014. Scalable 3d tracking of multiple interacting objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3430–3437.

Manfred Lau, Kapil Dev, Weiqi Shi, Julie Dorsey, and Holly Rushmeier. 2016. Tactile mesh saliency. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

Magic Leap. [Online; accessed 27-January-2021]. Hand Tracking. https://developer.magicleap.com/en-us/learn/guides/lumin-sdk-handtracking.

Gilwoo Lee, Zhiwei Deng, Shugao Ma, Takaaki Shiratori, Siddhartha S. Srinivasa, and Yaser Sheikh. 2019. Talking With Hands 16.2M: A Large-Scale Dataset of Synchronized Body-Finger Motion and Audio for Conversational Motion Analysis and Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Ian Lenz, Honglak Lee, and Ashutosh Saxena. 2015. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research* 34, 4-5 (2015), 705–724.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research* 37, 4-5 (2018), 421–436.

Ying Li, Jiaxin L Fu, and Nancy S Pollard. 2007. Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Transactions on Visualization and Computer Graphics* 13, 4 (2007), 732–747.

C Karen Liu. 2009. Dextrous manipulation from a grasping pose. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–6.

Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. 2019. Generating Grasp Poses for a High-DOF Gripper Using Neural Networks.

Anna Majkowska, Victor Zordan, and Petros Faloutsos. 2006. Automatic splicing for hand and body animations. In *ACM SIGGRAPH 2006 Sketches*. 32–es.

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. 2020. Catch & Carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 39–1.

Microsoft. [Online; accessed 27-January-2021]. Microsoft Mixed Reality Toolkit: Hand Tracking. https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/Input/HandTracking.html.

Gyeongsik Moon, Shoou-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. 2020. InterHand2.6M: A Dataset and Baseline for 3D Interacting Hand Pose Estimation from a Single RGB Image. *arXiv preprint arXiv:2008.09309* (2020).

Igor Mordatch, Zoran Popović, and Emanuel Todorov. 2012. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*. 137–144.

Franziska Mueller, Micah Davis, Florian Bernard, Oleksandr Sotnychenko, Mickeal Verschoor, Miguel A. Otaduy, Dan Casas, and Christian Theobalt. 2019. Real-time Pose and Shape Reconstruction of Two Interacting Hands With a Single Depth Camera. *ACM Transactions on Graphics (TOG)* 38, 4 (2019).

Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2017. Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Oculus. [Online; accessed 27-January-2021]. Oculus Touch Controllers. https://developer.oculus.com/documentation/native/pc/dg-input-touch-overview/.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019a. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 165–174.

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019b. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–11.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Sören Pirk, Vojtech Krs, Kaimo Hu, Suren Deepak Rajasekaran, Hao Kang, Yusuke Yoshiyasu, Bedrich Benes, and Leonidas J Guibas. 2017. Understanding and exploiting object interaction landscapes. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 1–14.

Nancy S Pollard and Victor Brian Zordan. 2005. Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 311–318.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*. 5099–5108.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087* (2017).

Javier Romero, Dimitrios Tzionas, and Michael J. Black. 2017. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36, 6 (Nov. 2017), 245:1–245:17. http://doi.acm.org/10.1145/3130800.3130883

M Santello, M Flanders, and J F Soechting. 1998. Postural hand synergies for tool use. *J Neurosci* 18, 23 (Dec 1998), 10105–10115.

M Santello, M Flanders, and J F Soechting. 2002. Patterns of hand motion during grasping and the influence of sensory guidance. *J Neurosci* 22, 4 (Feb 2002), 1426–35.

Gerrit Schoettler, Ashvin Nair, Jianlan Luo, Shikhar Bahl, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. 2019. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. *arXiv preprint arXiv:1906.05841* (2019).

Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. 2011. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*. Ieee, 1297–1304.

Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. 2017. Hand Keypoint Detection in Single Images Using Multiview Bootstrapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. 2017. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1746–1754.

Srinath Sridhar, Franziska Mueller, Michael Zollhoefer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. 2016. Real-time Joint Tracking of a Hand Manipulating an Object from RGB-D Input. In *Proceedings of European Conference on Computer Vision (ECCV)*. 17. http://handtracker.mpi-inf.mpg.de/projects/RealtimeHO/

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958. http://dl.acm.org/citation.cfm?id=2627435.2670313

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019), 209–1.

Omid Taheri, Nima Ghorbani, Michael J. Black, and Dimitrios Tzionas. 2020. GRAB: A Dataset of Whole-Body Human Grasping of Objects. In *European Conference on Computer Vision (ECCV)*. https://grab.is.tue.mpg.de

Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. 2014. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)* 33, 5 (2014), 1–10.

François Touvet, Agnès Roby-Brami, Marc A Maier, and Selim Eskiizmirliler. 2014. Grasp: combined contribution of object properties and task constraints on hand and finger posture. *Experimental brain research* 232, 10 (2014), 3055–3067.

Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. 2016. Capturing Hands in Action using Discriminative Salient Points and Physics Simulation. *International Journal of Computer Vision* 118, 2 (2016), 172–193.

Ultraleap. [Online; accessed 27-January-2021]. Gemini: Fifth-generation hand tracking platform". https://www.ultraleap.com/tracking/gemini-hand-tracking-platform/.

Vive. [Online; accessed 27-January-2021]. VIVE TRACKER: GO BEYOND VR CONTROLLERS. https://www.vive.com/us/accessory/vive-tracker/.

He Wang, Sören Pirk, Ersin Yumer, Vladimir G Kim, Ozan Sener, Srinath Sridhar, and Leonidas J Guibas. 2019. Learning a Generative Model for Multi-Step Human-Object Interactions from Videos. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 367–378.

Jiayi Wang, Franziska Mueller, Florian Bernard, Suzanne Sorli, Oleksandr Sotnychenko, Neng Qian, Miguel A. Otaduy, Dan Casas, and Christian Theobalt. 2020. RGB2Hands: Real-Time Tracking of 3D Hand Interactions from Monocular RGB Video. *ACM Transactions on Graphics (TOG)* 39, 6 (12 2020).

Nkenge Wheatland, Yingying Wang, Huaguang Song, Michael Neff, Victor Zordan, and Sophie Jörg. 2015. State of the Art in Hand and Finger Modeling and Animation. *Computer Graphics Forum* 34, 2 (2015), 735–760.

Jungdam Won and Jehee Lee. 2019. Learning body shape variation in physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

Yuting Ye and C Karen Liu. 2012. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–10.

Shanxin Yuan, Guillermo Garcia-Hernando, Björn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, Junsong Yuan, Xinghao Chen, Guijin Wang, Fan Yang, Kai Akiyama, Yang Wu, Qingfu Wan, Meysam Madadi, Sergio Escalera, Shile Li, Dongheui Lee, Iason Oikonomidis, Antonis Argyros, and Tae-Kyun Kim. 2018. Depth-Based 3D Hand Pose Estimation: From Current Achievements to Future Goals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2636–2645.

Hao Zhang, Zi-Hao Bo, Jun-Hai Yong, and Feng Xu. 2019. InteractionFusion: Real-Time Reconstruction of Hand Poses and Deformable Objects in Hand-Object Interactions. *ACM Transactions on Graphics* 38, 4 (2019).

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018a. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. 2018b. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2472–2481.
Wenping Zhao, Jianjie Zhang, Jianyuan Min, and Jinxiang Chai. 2013. Robust realtime physics-based motion control for human grasping. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–12.
Christian Zimmermann and Thomas Brox. 2017. Learning to Estimate 3D Hand Pose From Single RGB Images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.
Paula Zuccotti. 2015. *Every Thing We Touch: A 24-Hour Inventory of Our Lives.* Viking.

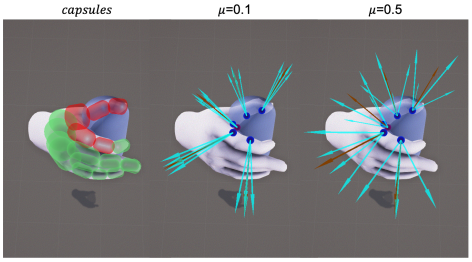


Fig. 17. Left: collision geometries. Colliding capsules highlighted in red. Middle: hand contact basis with a friction coefficient 0.1. Right: hand contact basis with a friction coefficient 0.5. Brown arrows are contact normals.

A PHYSICS EVALUATION

We evaluate the physical validity of the hand motion by looking at the rigid body dynamics of the object during in-hand manipulation. We want to make sure the input object motion can be fully supported by contact forces from the hands. Since we are dealing with casual activities with lightweight objects, we do not evaluate hand joint torques.

- **Dynamics** We follow the Barraff tutorial [Barraff 1997] to compute the change of linear momentum $\dot{P}$ and angular momentum $\dot{L}$ of the object at every frame. The linear and angular velocities (v, ω) and accelerations ($\dot{v}$, $\dot{\omega}$) are computed using finite differences from the input object trajectory.

$$P(t) = Mv(t) \qquad (linear\ momentum) \qquad (1)$$
$$\dot{P}(t) = M\dot{v}(t) \qquad (force) \qquad (2)$$
$$L(t) = I(t)\omega(t) \qquad (angular\ momentum) \qquad (3)$$
$$\dot{L}(t) = \dot{I}(t)\omega(t) + I(t)\dot{\omega}(t) \qquad (torque) \qquad (4)$$
$$I(t) = R(t)I_oR(t)^T \qquad (world\ space\ object\ inertia) \qquad (5)$$
$$\dot{I}(t) = \dot{R}(t)I_oR(t)^T + R(t)I_o\dot{R}(t)^T \qquad (6)$$
$$\dot{R}(t)^T = [\omega(t)]R(t) \qquad ([]\ is\ a\ skewsymmetric\ matrix) \qquad (7)$$

  We don't know the ground truth mass M of the objects. Since it only scales the forces, we simply set it to 1 for all objects. The object space inertia matrix I_o is computed from the 3D mesh of the objects. We stack $\dot{P}$ and $\dot{L}$ into a vector $\mathbf{b}$.
- **Contacts and forces** Hand-object contacts are detected based on the finger capsule colliders and the object mesh collider in Unity (see Fig. 17 left). For simplicity, each capsule collider can only detect at most one contact point. To model frictional contact, we use four basis to approximate Coulomb's friction cone for static friction. If we detect a sliding contact from the motion, one basis vector is used instead to model sliding friction. Fig. 17 shows the detected contacts and their force basis with two different friction coefficients from a grasping pose. We use 0.35 as the friction coefficient in our evaluation. Specially, we compute the force and torque at each frame by summing up the contributions from each contact location c_i as follows:

$$F(t) = \sum f_i(t) + Mg \qquad (force) \qquad (8)$$
$$\tau(t) = \sum [c_i(t) - o(t)]f_i(t) \qquad (torque) \qquad (9)$$
$$f_i(t) = V_i(t)x_i(t) \qquad (10)$$

  where $o(t)$ denotes the object center of mass, $V_i(t)$ denotes the linear force basis at each contact point, and x_i is the corresponding non-negative coefficients. We stack Eq. (8) and Eq. (9) into a linear system $\mathbf{Ax}$, where $\mathbf{x} = [x_0^T, x_1^T, ...]^T$.
- **Optimization with Non-Negative Least Square** We measure the violation of physics as the objective value of the following optimization:

$$\arg\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2, \ s.t.\ \mathbf{x} \geqslant \mathbf{0}$$

  We use the non-negative least square solver in Accord.NET Framework[2] with a maximum of 100 iterations. If a frame is physically valid, the objective will be optimized to zero. If the optimized objective is nonzero, it means no valid contact forces can balance the object motion. We threshold the value at 0.01 and use the ratio of success frames as our metric. We also evaluate the physicality of our training data this way as a baseline.

B POST-PROCESSING ALGORITHM

ALGORITHM 1: Calculate the target joint position for CCD-based IK given the joint position and distance predicted by the network.

Function `FindTargetJointPosition`(pos_{net}, dis_{net}):
  Input : Network-predicted joint position, distance: pos_{net}, dis_{net}
  Output: Target joint position: pos_{target}
  if dis_{net} <= 2.8cm **then**
    /* Get the nearest object surface point, the surface normal direction, the signed distance value for joint at pos_{net} from the object signed distance field. */
    $point_{sdf}, dir_{sdf}, dis_{sdf} \leftarrow$ SDFNNSearch(pos_{net});
    if $dis_{sdf} < 0$ **then**
      /* If penetrating, rely more on the predicted distance. */
      $dis_{final} \leftarrow 0.8 * dis_{net} + 0.2 * dis_{sdf}$;
    else
      /* If no penetration, rely more on the predicted pose. */
      $dis_{final} \leftarrow 0.2 * dis_{net} + 0.8 * dis_{sdf}$;
    end
    $pos_{target} \leftarrow point_{sdf} + dis_{final} * dir_{sdf}$;
  else
    $pos_{target} \leftarrow pos_{net}$;
  end
  return pos_{target}

[2]http://accord-framework.net