DexPilot: Vision Based Teleoperation of Dexterous Robotic Hand-Arm System

Ankur Handa*† Karl Van Wyk*† Wei Yang† Jacky Liang‡ Yu-Wei Chao†
Qian Wan† Stan Birchfield† Nathan Ratliff† Dieter Fox†

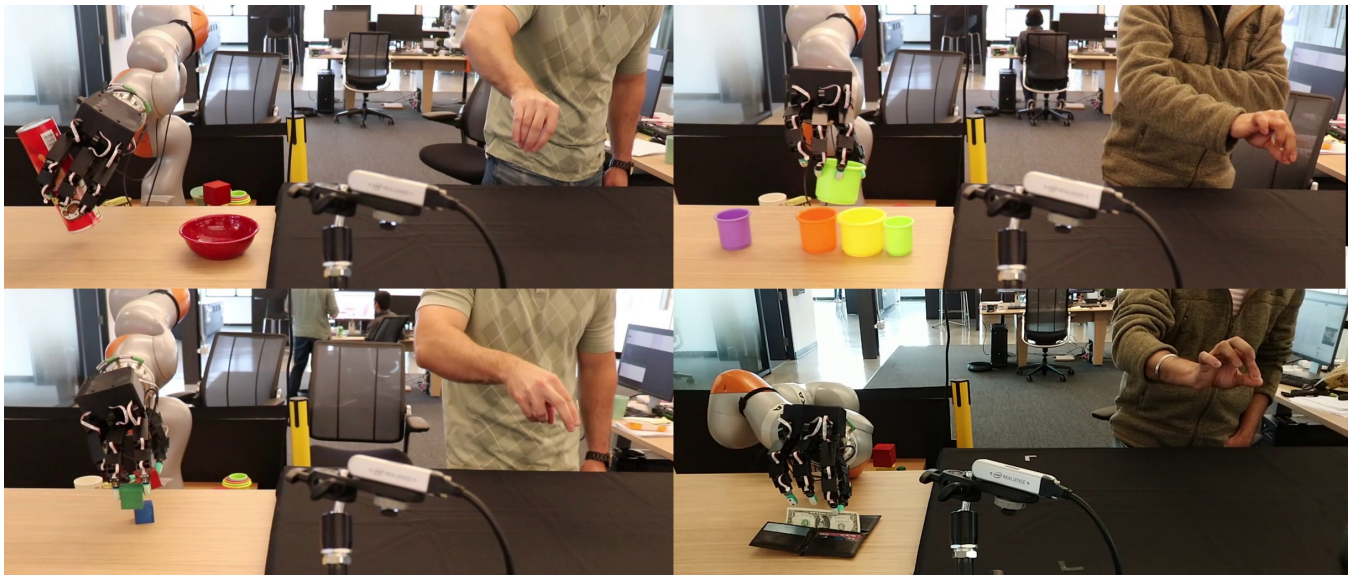

Fig. 1. DexPilot enabled teleoperation across a wide variety of tasks, *e.g.*, rectifying a Pringles can and placing it inside the red bowl (upper-left), inserting cups (upper-right), concurrently picking two cubes with four fingers (lower-left), and extracting money from a wallet (lower-right). Videos are available at `https://sites.google.com/view/dex-pilot`.

***Abstract*— Teleoperation offers the possibility of imparting robotic systems with sophisticated reasoning skills, intuition, and creativity to perform tasks. However, current teleoperation solutions for high degree-of-actuation (DoA), multi-fingered robots are generally cost-prohibitive, while low-cost offerings usually provide reduced degrees of control. Herein, a low-cost, vision based teleoperation system, DexPilot, was developed that allows for complete control over the full 23 DoA robotic system by merely observing the bare human hand. DexPilot enables operators to carry out a variety of complex manipulation tasks that go beyond simple pick-and-place operations. This allows for collection of high dimensional, multi-modality, state-action data that can be leveraged in the future to learn sensorimotor policies for challenging manipulation tasks. The system performance was measured through speed and reliability metrics across two human demonstrators on a variety of tasks. The videos of the experiments can be found at `https://sites.google.com/view/dex-pilot`.**

I. Introduction

Robotic teleoperation has been researched for decades with applications in search and rescue [1], space [2], medicine [3], prosthetics [4], and applied machine learning [5]. The primary motivation of teleoperation is to allow a robot system to perform complex tasks by harnessing the cognition, creativity, and reactivity of humans through a human-machine interface (HMI). Through the years, many advancements have been made in this research field including the incorporation of haptic feedback [6, 7] and improved human skeletal and finger tracking [8, 9, 10, 11]. Nevertheless, the cost of many high DoA teleoperation or tracking systems is prohibitive. This research bridges this gap by providing a low-cost, markerless, glove-free teleoperation solution that leverages innovations in machine vision, optimization, motion generation, and GPU compute. Despite its low-cost, the system retains the ability to capture and relay fine dexterous manipulation to drive a highly actuated robot system to solve a wide variety of grasping and manipulation tasks. Altogether, four Intel RealSense depth cameras and two NVIDIA GPUs in combination with deep learning and nonlinear optimization produced a minimal-footprint, dexterous teleoperation system. Despite the lack of tactile feedback, the system is highly capable and effective through human cognition. This result corroborates human gaze studies that indicate that humans learn to leverage vision for planning, control, and state prediction of hand actions [12] prior to accurate hand control. Therefore, the teleoperation system exploits the human ability to plan, move, and predict the consequence of their physical actions from vision alone; a sufficient condition for solving a variety of tasks. The main contributions are summarised below:

- Markerless, glove-free and entirely vision-based teleoperation system that dexterously articulates a highly-actuated robotic hand-arm system with direct imitation.
- Novel cost function and projection schemes for kinematically retargeting human hand joints to Allegro hand

* Equal Contribution
†NVIDIA, USA
‡CMU, Pittsburgh, PA, USA
@{ahanda,kvanwyk,dieterf}@nvidia.com

joints that preserve hand dexterity and feasibility of precision grasps in the presence of hand joint tracking error.
- Demonstration of teleoperation system on a wide variety of tasks particularly involving fine manipulations and dexterity, *e.g.*, pulling out paper currency from wallet and grasping two cubes with four fingers as shown in Fig. 1.
- System assessment across two trained human demonstrators — also called pilots — revealed that high task success rates can be achieved despite the lack of tactile feedback.

II. Related Work

Teleoperation via the human hand is traditionally done using either pure vision or sensorized glove-based solutions and relies on accurate hand tracking. Early vision-based solutions to hand tracknig include Dorner *et al.* [13] who used colour markers on the hand to track the 2D positions of joints and fit a skeleton model to obtain the hand pose and finger joints in 3D. Similarly, Theobalt *et al.* , [14] use colour markers to track the motion of the hand in high speed scenarios in sports. Wang *et al.* [15] propose a fully coloured glove to track the joints and pose of the hand for augmented reality (AR) and virtual reality (VR) applications. Impressive strides have also been made in the past few years in bare human hand tracking particularly with deep learning [9, 10, 11, 16, 17, 18]. Glove- and marker-free hand tracking is attractive as it produces minimal external interference with the hand and reduces the system physical footprint, but reliability and accuracy with such approaches have not yet reached desireable performance levels. Moreover, most of these works study hand tracking in isolation without evaluation of tracking performance in controlling a physical robot. Li *et al.* [19] proposed vision-based teleoperation through deep learning that calculated Shadow robotic hand joint angles from observed human hand depth images. However, this approach was not extended to a full robotic system and the mapping results had performance issues as later discussed in Section VII-A. Antotsiou *et al.* [20] show teleoperation of an anthropomorphic hand in simulation using depth based hand tracking. They focus on very simplistic grasping *e.g.* picking a ball on a flat table and do not show the level of dexterity as seen herein. In general, hand tracking with vision remains a very challenging problem due to self-similarity and self-occlusions observed in the hand.

On the other end of the spectrum, various commercial glove- or marker-based solutions offer accurate hand tracking. Additionally, approaches from CyberGlove [7] and HaptX [6] do provide the user with tactile feedback in the form of physical resistance and spatial contact, but this benefit comes with heightened cost and additional bulk to the user. Sometimes these systems still require an external hand pose estimation module as in [21] to enable a freely moving mobile hand as with DexPilot. Still, these approaches are important to investigate how relaying tactile feedback to human pilots improve teleoperation capability considering that numerous neurophysiological studies have uncovered the importance of tactile feedback in human sensorimotor control of the human hand [12, 22, 23]. Alternatively, motion capture systems provide accurate point tracking solutions, but can be expensive with multi-camera systems. Moreover, when tracking all the joints of the hands, the correspondence problem between markers on the fingers and cameras still needs to be solved. For instance, Han *et al.* [24] track hands with an array of OptiTrack cameras while solving the correspondence problem via a neural network. They show impressive tracking of two hands in VR settings. Zhuang *et al.* [4] leverage a motion capture system and myoelectric device for teleoperative control over a hand-arm system. The system worked convincingly well as a prosthetic HMI, but experiments did not reveal how well the system works for intricate in-hand manipulation tasks (as assessed herein). Overall, these solutions are viable alternatives, but are significantly more expensive and with larger physical footprints than DexPilot. Finally, some of these approaches only consider the problem of tracking hand movements, and are not complete teleoperation solutions like DexPilot.

We also note a few low-cost hardware solutions that have emerged lately. For example, [25] and [26] showed that low-cost solutions for teleoperation with a glove can be constructed though they tend to be quite bulky and have wires tethered to them.

III. Hardware

The teleoperation setup comprised of a robot system and an adjacent human pilot arena as shown in Fig. 3. The robot system was a KUKA LBR iiwa7 R800 series robot arm and a Wonik Robotics Allegro hand. The Allegro hand was retrofitted with four Syntouch Biotac tactile sensors at the fingertips and 3M TB641 grip tape applied to the inner surfaces of the phalanges and palm. The rubbery surfaces of both the Biotac sensors and 3M tape augmented the frictional properties of the hand while the Biotacs themselves produced 23 signals that can later be used to learn sensorimotor control from demonstrations. In total, the robotic system has 92 tactile signals, 23 joint position signals, and 23 joint torque actions. The human arena was a black-clothed table that housed four calibrated and time-synchronized Intel Realsense D415 RGB-D cameras spatially arranged to yield good coverage of the observation volume within the ideal camera sensing range. Ideally, depth observations should remain within 1 m for each camera; otherwise, the depth quality degenerates. The human arena is directly adjacent to the robot to improve line-of-sight and visual proximity since teleoperation is entirely based on the human vision and spatial reasoning. The teleoperation work volume is 80 $cm \times 55\ cm \times 38\ cm$.

IV. Architecture

To produce a natural-feeling teleoperation system, an imitation-type paradigm is adopted. The bare human hand motion — pose and finger configuration — was constantly observed and measured by the visual perception module. The human hand motion is then relayed to the robot system in such a way that the copied motion is self-evident. This approach enables the human pilot to curl and arrange their fingers, form grasps, reorient and translate their palms, with the robot system following in a similar manner. DexPilot relies heavily on DART [27] which forms the backbone of tracking the pose and joint angles of the hand. In the following, we explain the main components of the overall system:

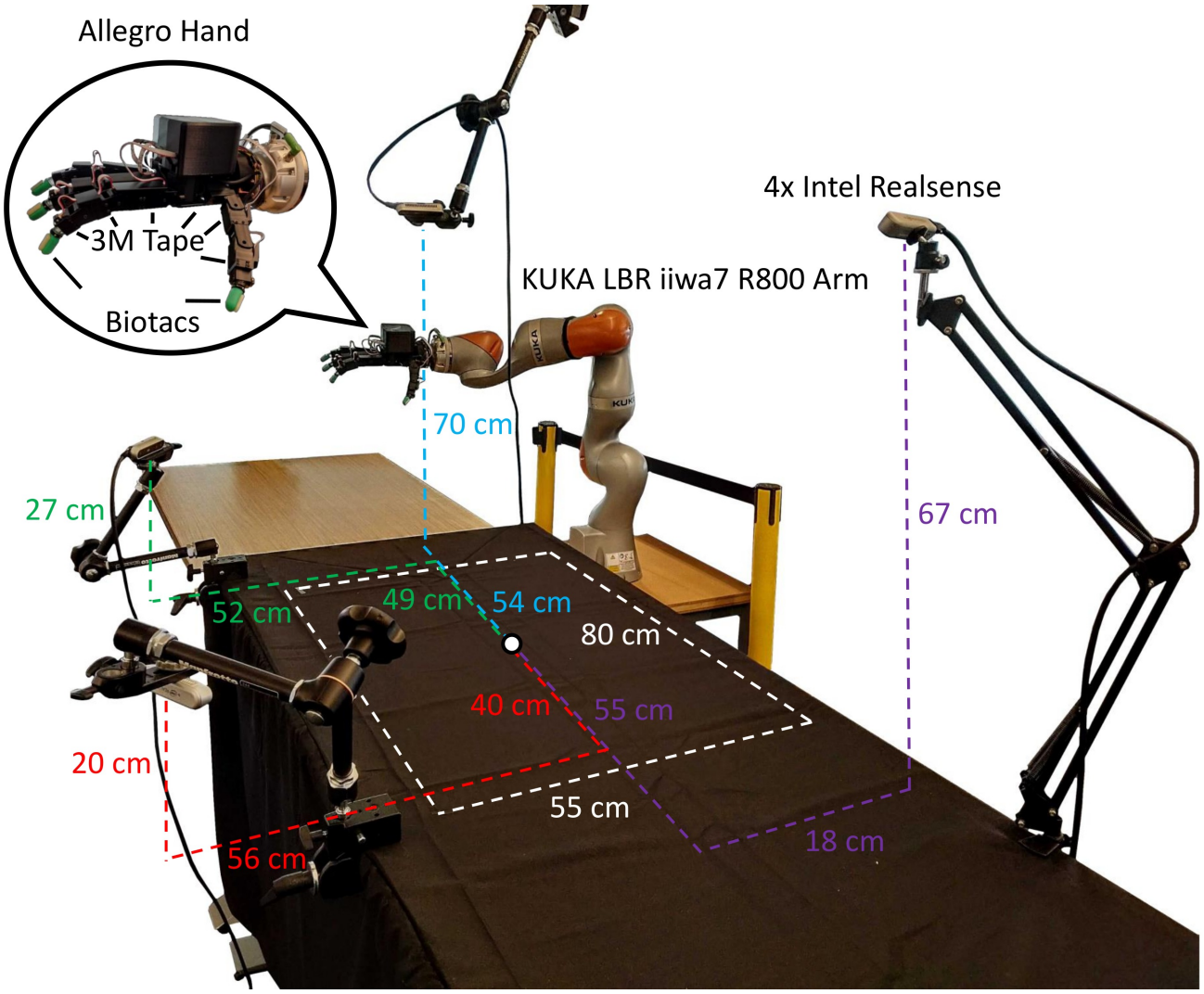

Fig. 3. Studio is composed of four cameras pointing towards the table over which the user moves their hand with the hand-arm system in close proximity to enable line of sight driven teleoperation.

1) DART for hand tracking (Section V). 2) deep neural networks for human hand state estimation and robustifying DART (Section VI). 3) human hand state refinement with DART and its conversion through nonlinear optimization to Allegro hand states (Section VII-A) 4) motion generation and control through Riemannian Motion Policies (RMPs) and torque-level impedance controllers (Section VII-B). The fully system architecture is shown in Fig. 4 where these components are daisy chained. Altogether, the system produces a latency of about one second.

V. DART

DART [27] produces continuous pose and joint angle tracking of the human hand by matching an articulated model of the hand against an input point cloud. The human hand model was obtained from [28] and turned into a single mesh model [29]. With CAD software, the fingers of the mesh model were separated into their respective proximal, medial, and distal links, and re-exported as separate meshes along with an associated XML file that described their kinematic arrangement. In total, the human hand model possessed 20 revolute joints: four joints per finger with one abduction joint and three flexion joints.

VI. Estimating Hand Pose and Joint Angles with Neural Networks

Since DART is a model-based tracker that relies on non-linear optimisation, it needs initialisation which typically comes from estimates from previous frame. If this initialisation is not within the basin of convergence, the tracker can fail catastrophically. Often, DART can match the hand model to the raw point cloud in regions of spurious local minima leading to tracking failures every few minutes. Therefore, to allow for tracking over long periods of time — as needed for teleoperation — hand pose priors and hand segmentation can be implemented to prevent the hand model from converging to incorrect local minima. One way of getting hand pose priors would to be train a neural network on a large dataset of human hand poses. Although various datasets [9, 10] exist that provide annotations for hand pose as well as segmentation, they were not suitable for this setting primarily due to different sensor characteristics and lack of hand poses

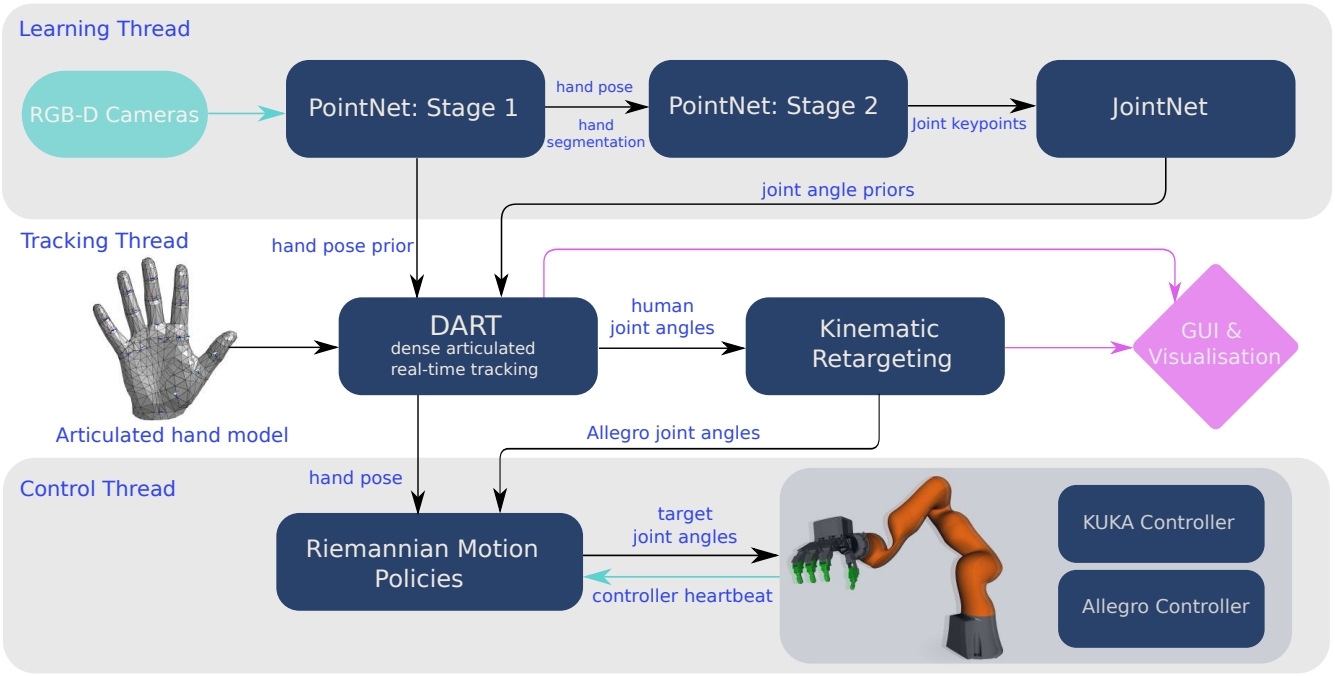

Fig. 4. The system is comprised of three threads that operate on three different computers. The learning thread provides hand pose and joint angle priors using fused input point cloud coming from four cameras from the studio. The tracking thread runs DART for hand tracking with the priors as well as kinematic retargeting needed to map human hand configuration to allegro hand. The control thread runs the Riemannian motion policies to provide the target joint commands to the KUKA and allegro hand given the hand pose and joint angles.

Fig. 5. The colour glove used in the first phase to obtain hand pose and segmentation for DART. Unique colours are printed such that annotation generation is trivial with OpenCV colour thresholding. The colours on the back of the palm can uniquely determine the pose of the hand.

required for dexterous manipulation.

Therefore, a fabric glove (shown in Fig. 5) with coloured blobs was initially used as an effective solution for obtaining a hand pose prior with a deep neural network. The data collection proceeded in two phases. In the first phase, the user wore the glove to obtain hand pose priors for DART to track human hand robustly. This process generated hand pose and joint angle annotations for raw depth maps from the RGB-D cameras for the second phase. The second phase uses these annotations and operates on raw point cloud from corresponding depth maps and frees the user from having to wear the glove. We explain differetnt phases below.

*a) **First Phase**:* The color glove is inspired by that of [13, 15] who used it for hand tracking. It is important to clarify that hand tracking includes both the hand pose and the joint angles of the fingers. In our set-up, the user moves their hand over a table in a multi-camera studio with four Intel RealSense D415 RGB-D cameras pointing downwards to the table. The problem of hand pose estimation is formulated with the glove via keypoint localisation: ResNet-50 [30] with spatial-softmax is used to regress from an RGB image to the 2D locations of the centers of the coloured blobs on the glove. We call this network GloveNet. The coloured blobs at finger-tips are also regressed but were found to be not helpful in full hand tracking in the end and therefore we only use the predictions of the blobs on the back of the palm for hand pose estimation. We explain that in detail in the appendix.

The hand pose can be estimated by three unique keypoints as indicated by three different coloured blobs at the back of the palm of glove. To obtain annotations for the centers of the blobs, HSV thresholding in OpenCV is used to generate segmentations and compute the centeroids of these segmented coloured blobs. To aid segmentation for high quality annotations, the user wears a black glove with coloured blobs and moves the hand over a table also covered with black cloth. The pose of the hand can be obtained via predicted 2D locations of the blobs from all four cameras: the 2D keypoints are converted to their corresponding 3D locations using the depth values resulting in each blob having four 3D predictions in total from four cameras. These 3D locations are filtered and temporally smoothed to obtain the hand pose. Hand segmentation is also obtained by removing the 3D points that fall outside the bounding volume of the hand. The dimensions of this volume were obtained heuristically from the hand pose obtained from the neural network predictions. Crucially, DART now only optimises on the segmented hand points, preventing the hand model from sliding out to points on the arm as often observed when a full point cloud is used. It is worthwhile remembering that DART does not use RGB images — the glove only provided pose priors

and aided hand segmentation — and therefore the result of DART with hand pose priors and segmentation in the first phase is generating annotations for raw point cloud captured with the cameras for second phase which can operate on the bare human hand.

*b) **Second Phase**:* It is desirable to free the user from having to wear glove in future for any teleoperation. While the first phase operates on RGB image, the second phase operates directly on fused point cloud of bare human hand obtained by back-projecting four depth maps from extrinsically calibrated cameras into a global reference frame with annotations provided by first phase. Since the fused point cloud contains both the points on table as well as human body and arm it becomes imperative to first localise the hand. Points on the table are removed by fitting a plane and the remaining points containing the arm and human body are fed to a PointNet++ based [31] architecture that localises the hand as well as provides the hand pose. Our network is based on [11] who estimate hand pose via a voting based regression to the 3D positions of specified keypoints on the hand, a technique reminiscent of spatial-softmax often used in 2D keypoint localisation. It is trained to predict 3D coordinates of 23 keypoints specified on the hand — 4 joint keypoints each on 5 fingers and 3 at the back of the palm for hand pose estimation. The loss function is standard Euclidean loss between predicted and the ground truth keypoints together with the voting loss inspired by [11]. An auxiliary segmentation loss is also added to obtain hand segmentation. For efficiency reasons, any input point cloud of size $N \times 3$ is sub-sampled uniformly to a fixed 8192×3 size before feeding to our network.

While reasonable hand pose estimation and segmentation is achieved, getting high quality predictions for the 20 joint keypoints on the fingers remains difficult with this network. The uniform sub-sampling used at the input means that points on the fingers are not densely sampled and therefore a second stage refinement is needed which resamples points on the hand from the original raw point cloud given the pose and segmentation of the first stage. The overall network architecture is shown in Fig. 6. The second stage is trained on only the loss functions pertaining to the keypoints and no segmentation is needed. It uses the points sampled on the hand instead and predicts accurately the 23 keypoints. To enable robustness to any inaccuracies in the hand pose from the first stage, additional randomization is added to the hand pose for second stage. The Fig. 7 shows how the second stage refinement improves the system. Overall, both stages of our network are trained on 100K point clouds collected over a batch of 30-45 minutes each for 7-8 hours in total by running DART with priors from glove. Together they provide annotations for keypoints, joint angles and segmentation. The training takes 15 hours in total on a single NVIDIA TitanXp GPU.

While keypoints are a natural representation for Euclidean space as used in PointNet++ architectures, most articulated models use joints as a natural parameterisation. Therefore, it is desirable to have output in joint space which can serve as joint priors to DART. A third neural network is trained that maps 23 keypoint locations predicted by our PointNet++ inspired architecture to corresponding joint angles. This neural network, called JointNet, is a two-layer fully connected network that takes input of size 23×3 and predicts 20-dimensional vector of joint angles for fingers.

The neural networks are trained on data collected within the limits of the studio work volume across multiple human hands, ensuring accurate pose fits for this application and enabling sensible priors for DART. Qualitatively, the hand tracker worked well for hands geometrically close to the DART human hand model. Overall, average keypoint error on a validation set of seven thousand images of differing hand poses and finger configurations was 9.7 mm (comparable to results reported in [11], but on a different dataset) and joint error was 1.33 degrees per joint.

VII. Robot Motion Generation

A. Kinematic Retargeting

Teleoperation of a robotic hand that is kinematically disparate from the human hand required a module that can map the observed human hand joints to the Allegro joints. There are many different approaches to kinematic retargeting in the literature. For instance, a BioIK solver was used to match (between the human and Shadow hand) the positions from palm to the fingertips and medial joints, and the directionality of proximal phalanges and thumb distal phalange in [19]. The optimized mapping was used to label human depth images to learn end-to-end a deep network that can ingest a depth image and output joint angles for the Shadow hand. Although interesting, the result produced retargeting results that are not useful for precision grasps (e.g., pinching) where gaps between fingertips need to be small or zero. Motion retargeting is also present in the animation field. For instance, a deep recurrent neural network was unsupervised trained to retarget motion between skeletons [32]. Although the synthesized motion look compelling, it is unclear whether these approaches work well enough for teleoperative manipulation where the important task spaces for capturing grasping and manipulation behavior are likely not equivalent to those that capture visual consistencies. The approach herein prioritized fingertip task-space metrics because distal regions are of highest priority in grasping and manipulation tasks as measured by their contact prevalence [33], degree of innervation [12], and heightened controllability for fine, in-hand manipulation skill [34]. Moreover, the joint axes and locations between the two hands are strikingly different, and therefore, no metrics directly comparing joint angles between the two hands are used. To capture and optimize for the positioning of fingertips, both distance and direction among fingertips were considered. Specifically, the cost function for kinematic retargeting was chosen as

$$\mathcal{C}(q_h, q_a) = \frac{1}{2} \sum_{i=0}^{\mathsf{N}} s(d_i) ||r_i(q_a) - f(d_i)\hat{r}_i(q_h)||^2 + \gamma ||q_a||^2,$$

where q_h, q_a are the angles of the human hand model and Allegro hand, respectively, $r_i \in \mathbb{R}^3$ is the vector pointing from the origin of one coordinate system to another, expressed in the origin coordinate system (see Fig. 8). Furthermore, $d_i = ||r_i(q_h)||$ and $\hat{r}_i(q_h) = \frac{r_i(q_h)}{||r_i(q_h)||}$. The switching weight function $s(d_i)$ is defined as

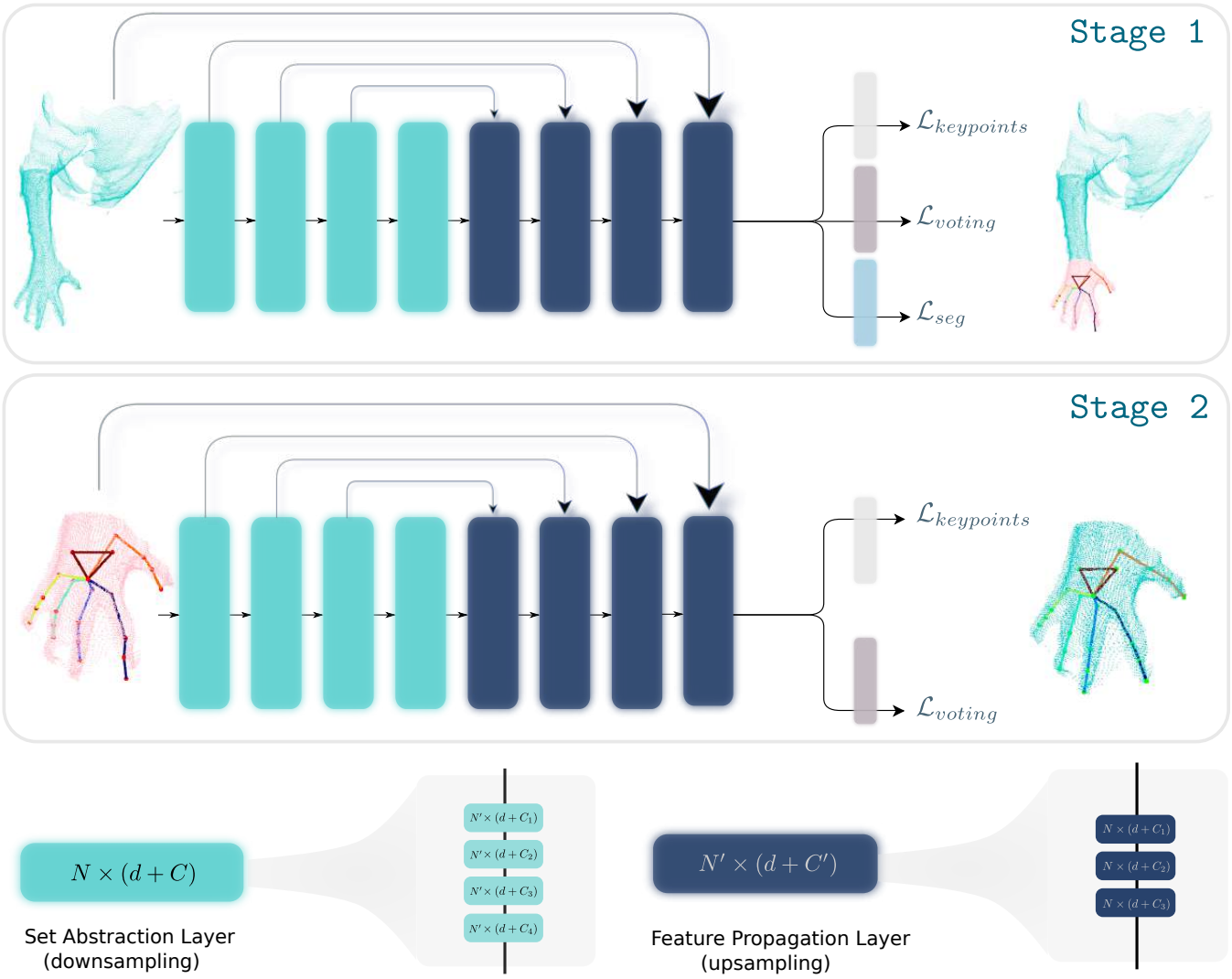

Fig. 6. The PointNet++ inspired architecture operates in two stages. The first stage segments the hand (as shown in pink colour) as well as provides a rough hand pose. The second stage refines the hand pose given the hand segmentation and pose from the first stage. The loss functions include the segmentation loss, the Euclidean loss between the predicted keypoints and ground truth keypoints, and the voting loss as used in [11]. Since the second stage refines keypoints, the segmentation loss is not needed. The set abstraction takes an input of size $N \times (d + C)$ and outputs $N' \times (d + C_4)$ while the feature propagation layer takes $N' \times (d + C')$ input and outputs a tensor of size $N' \times (d + C_3)$. Together these two form the backbone of the network. MLPs are used to map the embeddings of PointNet++ backbone to the corresponding desired outputs. More details of the network are in the Appendix.

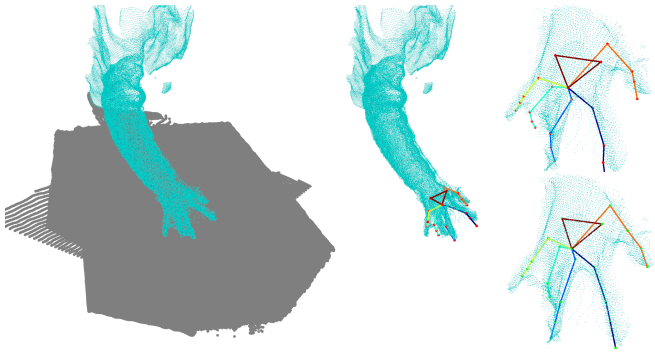

Fig. 7. The input point cloud has points both from the table as well as the human body and arm. A plane was fit to remove points on the table and the remaining points are input to the first stage of our network that recovers the pose of the hand. The second stage refines the pose and provides a more accurate result. The hand images on the right show the result from stage 1 (above) and stage 2 (below).

$$s(d_i) = \begin{cases} 1, & d_i > \epsilon \\ 200, & d_i \leq \epsilon \ \wedge \ r_i(q_h) \in \mathcal{S}_1 \\ 400, & d_i \leq \epsilon \ \wedge \ r_i(q_h) \in \mathcal{S}_2, \end{cases}$$

where $\mathcal{S}_1$ and $\mathcal{S}_2$ are vector sets defined in Table I, The distancing function, $f(d_i) \in \mathbb{R}$, is defined as

$$f(d_i) = \begin{cases} \beta\, d_i, & d_i > \epsilon \\ \eta_1, & d_i \leq \epsilon \ \wedge \ r_i(q_h) \in \mathcal{S}_1 \\ \eta_2, & d_i \leq \epsilon \ \wedge \ r_i(q_h) \in \mathcal{S}_2, \end{cases}$$

where $\beta = 1.6$ is a scaling factor, $\eta_1 = 1 \times 10^{-4}\, m$ closes the distance between a primary finger and the thumb, and $\eta_2 = 3 \times 10^{-2}\, m$ forces a minimum separation distance between two primary fingers when both primary fingers are being projected with the thumb. These projections ensure that

TABLE I. Description of vector sets used in kinematic retargeting.

Set	Description
$\mathcal{S}_1$	Vectors that originate from a primary finger (index, middle, ring) and point to the thumb.
$\mathcal{S}_2$	Vectors between two primary fingers when both primary fingers have associated vectors $\in \mathcal{S}_1$, *e.g.*, both primary fingers are being projected with the thumb.

contact between primary fingers and the thumb are close without inducing primary finger collisions in a precision grasp. This was found to be particularly useful in the presence of visual finger tracking inaccuracies. Importantly, the vectors r_i not only capture distance and direction from one task space to another, but their expression in local coordinates further contains information on how the coordinate systems, and thereby fingertips, are oriented with one another. The coordinate systems of the human hand model must therefore have equivalent coordinate systems on the Allegro model with similarity in orientation and placement. The vectors shown in Fig. 8 were a minimal set that produced the desired retargeting behavior. Finally, $\gamma = 2.5 \times 10^{-3}$ is a weight on regularizing the Allegro angles to zero (equivalent to fully opened the hand). This term helped greatly with reducing redundancy in solution and ensured that the hand never entered strange minima that was difficult to recover from (e.g., the fingers embedding themselves into the palm). Finally, to further reduce redundancy, the search space, and to emulate the human hand, the distal joints for the primary (index, middle, and ring) fingers of the Allegro hand were constrained to equal their medial joints. Various mappings from human hand to Allegro as produced by our kinematic retargeting are show in Fig. 9.

For implementation, the above cost function was minimized in real-time using the Sequential Least-Squares Quadratic Programming (SLSQP) algorithm [35, 36] of the NLopt library [37]. The routine was initiated with Allegro joint angles set to zero, and every solution thereafter was initiated with the preceding solution. Moreover, the forward kinematic calculations between the various coordinate systems of both the human hand model and Allegro hand were found using the Orocos Kinematics and Dynamics library [38]. Finally, a first-order low-pass filter was applied to the raw retargeted joint angles in order to remove high-frequency noise present in tracking the human hand and to smooth discrete events like the projection algorithm inducing step-response changes in retargeted angles.

B. Riemannian Motion Policies

Riemannian Motion Policies (RMPs) are real-time motion generation methods that calculate acceleration fields from potential function gradients and corresponding Riemannian metrics [39, 40]. RMPs combines the generation of multi-priority Cartesian trajectories and collision avoidance behaviors together in one cohesive framework. They are used to control the Cartesian pose of the Allegro palm given the observed human hand pose while avoiding arm-palm collisions with the table or operator using collision planes. Given these objectives, the RMPs generated target arm joint trajectories that were sent to the arm's torque-level impedance controller at 200 Hz. The kinematically retargeted

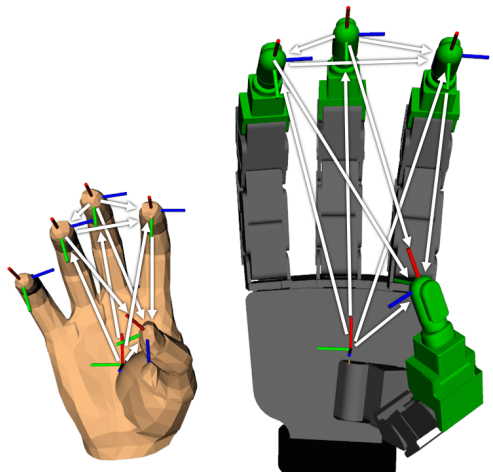

Fig. 8. Task space vectors between fingertips and palm for both the human hand model and Allegro hand used for retargeting optimization.

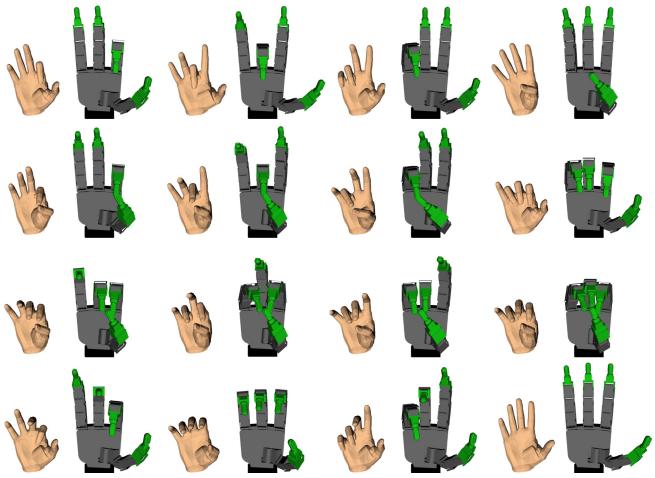

Fig. 9. Canonical kinematic retargeting results between the human hand model and the Allegro hand model.

Allegro angles were sent to the torque-level joint controller at 30 Hz. One final calibration detail involves registering human hand pose movements with the robot system. This was accomplished by finding the transformation from the robot coordinate system to the camera coordinate system. This transformation was calculated using the initial view of the human hand and an assumed initial pose of the robot hand. To facilitate spatial reasoning of the pilot, the desired initial hand pose of the pilot is a fully open hand with the palm parallel to the table and fingers pointing forwards. The assumed initial pose of the robot mimics this pose. In this way, the robot moves in the same direction as the pilot's hand, enabling intuitive spatial reasoning.

VIII. Experiments

The DexPilot system was systematically tested across a wide range of physical tasks that test precision and power grasps, prehensile and non-prehensile manipulation, and finger gaiting (see Fig. 10 for test objects and Figs. 11, 12, 13 for teleoperative manipulation). The description of the tasks are provided in Table II.

Before benchmarking, the pilots went through a warm-up training phase where they tried to solve the task with 3-5 non-

Fig. 10. Objects used for teleoperation. Various tasks performed with these objects can be viewed at `https://sites.google.com/view/dex-pilot`.

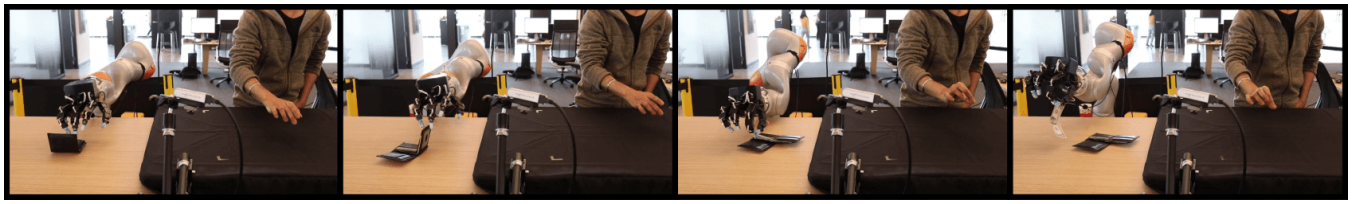

Fig. 11. Extracting money from a wallet. The pilot has to open the wallet first and move it to a particular vantage location in order to pull out paper currency. Importantly, the hand is able to keep the paper by pinching fingers against each other.

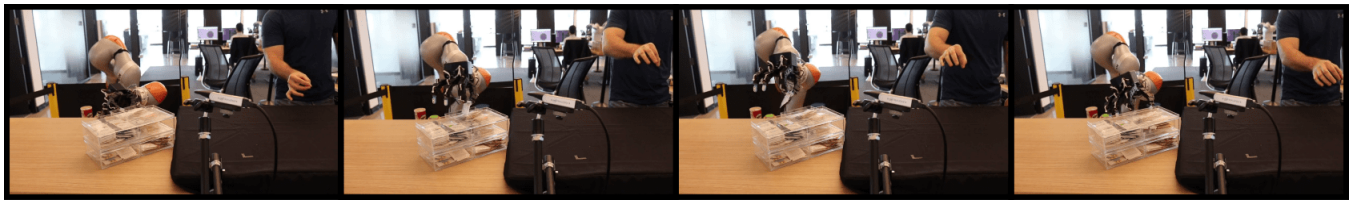

Fig. 12. Opening a tea drawer, extracting a tea bag, and closing the drawer. This is a somewhat long horizon task and requires dexterity in opening the drawer and holding on to the tea bag.

Fig. 13. Opening a peanut jar. The task needs rotating the cap multiple times in order to open while maintaing the contacts.

TABLE II. The test suite consists of 15 different tasks of varying complexity ranging from classic pick and place to multi-step, long horizon tasks. Each of these tasks is operated with 5 consecutive trials to avoid preferential selection and success rate is reported accordingly. If the object falls out of the workspace volume the trial is considered a failure. The last column represents the skills needed for teleoperation as the hand changes its state over time.

Task	Description	Required Skills
Pick and Place • Foam brick • Pringles can • Spam box	Pick object on the table and place it in a red bowl.	grasping, releasing
Block Stacking • Large (L) (6.3cm) • Medium(M) (3.8cm) • Small (S) (2.3cm)	Stacking three blocks on top of each other.	precision grasping, precision releasing
Pouring Beads	Pour beads from a cup into a bowl.	grasping, pouring
Opening Jar	Open peanut jar and place lid on table.	finger gaiting, grasping, releasing
Brick Gaiting	Pick up and in-hand rotate brick 180 degrees and place back down.	grasping, in-hand manipulation, releasing
Container	Open plastic container, extract and open cardboard box.	twisting, pulling, pushing, grasping, in-hand manipulation
Cup Insertion	Inserting concentric cups inside each other.	grasping, releasing
Tea Drawer	Pull open tea drawer, extra single bag of tea and place on table, close tea drawer.	precision grasping, pulling, pushing, releasing
Card Sliding	Slide a card along the box and pick it up with two fingers.	sliding, precision grasping, releasing
Wallet	Open the wallet and pull out paper money.	precision grasping, pulling, pushing, in-hand manipulation
Box Flip	Flip the box by 90 degrees and place it on the designated goal.	pushing, grasping, releasing

consecutive attempts. Later, five consecutive test trials were conducted by each pilot for each task to avoid preferential selection of results and pilots were graded based on their performance. The performance metrics for these tasks include mean completion time (CT) and success rate which capture speed and reliability of the teleoperation system. The system was tested with two pilots and the performance measures are reported in Fig. 14 (see more videos at `https://sites.google.com/view/dex-pilot`). Overall, the system can be reliably used to solve a variety of tasks with a range of difficulty. Differences in mean CT across tasks indicate the effects of task complexity and horizon scale. Discrepancies in mean CT across pilots per task indicate that there does exist a dependency on pilot behavior. Effects include fatigue, training, creativity, and motivation. The ability to solve these tasks reveal that the teleoperation system has the dexterity to exhibit precision and power grasps, multi-fingered prehensile and non-prehensile manipulation, in-hand finger gaiting, and compound in-hand manipulation (*e.g.*, grasping with two fingers while simultaneously manipulating with the remaining fingers). Note, certain tasks, *e.g.* Container and Wallet, take a particularly long time to teleoperate largely due to the fact that these tasks are multi-stage tasks. On the other hand, the task requiring picking small cubes is particularly challenging because the behavior of releasing the grasps on these objects with the projection scheme used in kinematic retargeting can be unpredictable. Nevertheless, such a rich exhibition of dexterous skill transferred solely through the observation of the bare human hand provides empirical evidence that the approach and architecture herein works well. An important aspect that is worth highlighting is that although the full teleoperation process for a particular task may not be perfect (*e.g.* the pilot may lose an object in hand but fetches it again to accomplish the task), the data collected is still equally valuable in helping robot learn to recover from failures. Additionally, in the spirit of [41],

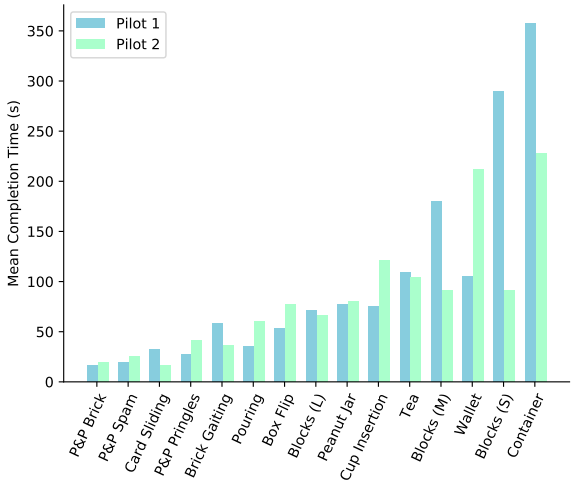

Fig. 14. Mean completion time of teleoperation tasks across two pilots run over five successive trials without reset.

the data can be regarded as play data which is useful to learn long range planning. Visualization of a sensorimotor solution to the `Brick Gaiting` task can be seen in Fig. 16. As shown, discrete events like intermittent finger-object contacts can be observed in the tactile signals. Undulations in these state-action signals reveal the rich, complex behavior evoked in the system through this embodied setting. Force estimates can also be obtained as in [42]. This data can now be generated on-demand for a particular task with the hope that functional sensorimotor patterns may be gleaned and imparted to the system in an autonomous setting.

IX. Discussion

DexPilot enabled a highly-actuated hand-arm system to find a motor solution to a variety of manipulation tasks by translating observed human hand and finger motion to

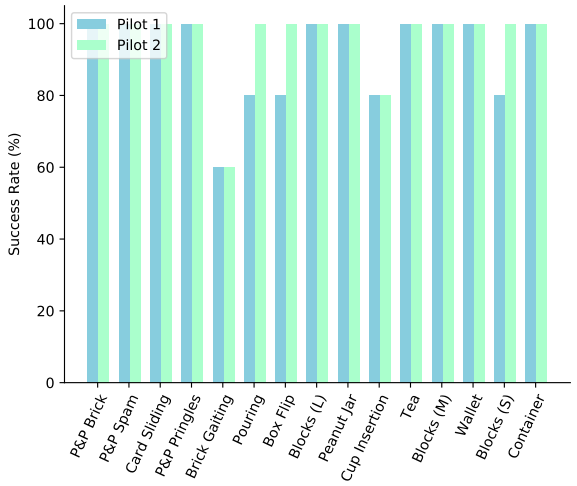

Fig. 15. Success rate of teleoperation tasks across two pilots run over five successive trials without reset.

robot arm and finger motion. Importantly, several tasks like extracting money from a wallet and opening a cardboard box within a plastic container were so complex that hand-engineering a robot solution or applying learning methods directly are likely intractable. Solving these tasks and the others through the embodied robotic system revealed that a sensorimotor solution did exist and that these solutions can be generated on-demand for many demonstrations. Furthermore, creating these solutions on the system itself allows for the reading, access, and storage of the 92 tactile signals in the robot's fingertips, 23 commanded and measured joint position and velocity signals through the hand and arm, 23 torque commands throughout the system, and any camera feeds associated with the system. This rich source of data will be critical for the application of learning methods that may hopefully learn to solve complex, multi-stage, long horizon tasks. Applying these learning methods is a future direction for this enabling work. Moreover, the current DexPilot system will be improved in a variety of ways in the future as well. For instance, human hand tracking accuracy could be improved with advancements in deep learning architectures, inclusion of RGB data, larger data sets, and changes to imaging hardware. Ideally, human hand tracking accuracy should be improved enough to greatly reduce the projection distance in the kinematic retargeting approach, enhancing fine finger control and manipulation over small objects and multi-fingered precision grasps. Grasp and manipulation control algorithms [34] could be implemented on the hand that automates force modulating control to reduce the control burden on the user and minimize unintentional part drops from the application of incorrect grip forces. Finally, intent recognition schemes could be implemented that enables the robot to predict human intention and deploy automated solutions, *e.g.*, the system recognizes the human's intent to grasp an object and the system automatically acquires the grasp. Such a co-dependent system would allow a human to direct the robot with full knowledge of a task and its solution strategy, while the robot system controls the finer details of the solution implementation.

X. Limitations

Overall, DexPilot is a viable, low-cost solution for teleoperating a high DoA robotic system; however, there are areas that could be improved with the current implementation. For instance, the observable work volume of the pilot could be enlarged to allow for tasks that cover greater distances with better RGB-D cameras. The projection schemes in kinematic retargeting enabled successful manipulation of small objects, but can interfere with finger gaiting tasks and timely releasing grasps on small objects as shown in Fig. 18. Fortunately, this feature can be turned off when desired, but ultimately, this interference should be non-existent. This issue could be solved entirely with hand tracking that can accurately resolve situations where the human hand fingertips are making contact. Human hand tracking could also be further improved with enhanced robustness across size and shape of the pilot's hand. The lack of tactile feedback makes precision tasks difficult to complete. To compensate, building in autonomous control features could alleviate some of the control burden on the pilot. Furthermore, the system latency could be reduced and the responsiveness of the RMP motion generator could be tuned for faster reactions. Finally, high-precision tasks like slip-fit peg-in-hole insertions pose a challenge to the current system. Peg-in-hole insertions on the NIST task board [43, 44] were attempted with DexPilot, but results were mostly unsatisfactory. Fig. 17 shows one attempt where the pilot managed to successfully guide the system to insert a 16 mm $\times$ 10 mm $\times$ 49.5 mm peg with a 0.1 mm hole clearance. Encouragingly, these tasks which have not been successfully solved by robotic systems that use machine vision and robotic hands are now made possible by such systems. However, success rates are typically within 10 % when under specific conditions on the initial placement of the NIST task board and parts (close to the user). The difficulty of completing such tasks could be significantly reduced with improved hand tracking performance, automated precision grip control on the assembly objects, and improved sight to the small parts and insertion locations.

XI. Acknowledgements

We would like to thank Adithya Murali, Jonathan Tremblay, Balakumar Sundaralingam, Clemens Eppner, Chris Paxton, Yashraj Narang, Alexander Lambert, Timothy Lee, Michelle Lee, Adam Fishman, Yunzhu Li, Ajay Mandlekar, Muhammad Asif Rana, Carlos Florensa, Krishna Murthy Jatavallabhula, Jonathan Tompson, Joseph Xu, Kendall Lowrey, Lerrel Pinto, Ian Abraham, Jan Czarnowski, Raluca Scona, Tucker Hermans, Svetoslav Kolev, Umar Iqbal, Pavlo Molchanov, Emo Todorov, Artem Molchanov, Yevgen Chebotar, Viktor Makoviychuk, Visak Chadalavada and James Davidson for useful discussions and feedback during the development of this work.

References

[1] A. Norton, W. Ober, L. Baraniecki, E. McCann, J. Scholtz, D. Shane, A. Skinner, R. Watson, and H. Yanco, "Analysis of human–robot interaction at the darpa robotics challenge finals," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 483–513, 2017.

[2] M. Diftler, T. Ahlstrom, R. Ambrose, N. Radford, C. Joyce, N. De La Pena, A. Parsons, and A. Noblitt, "Robonaut

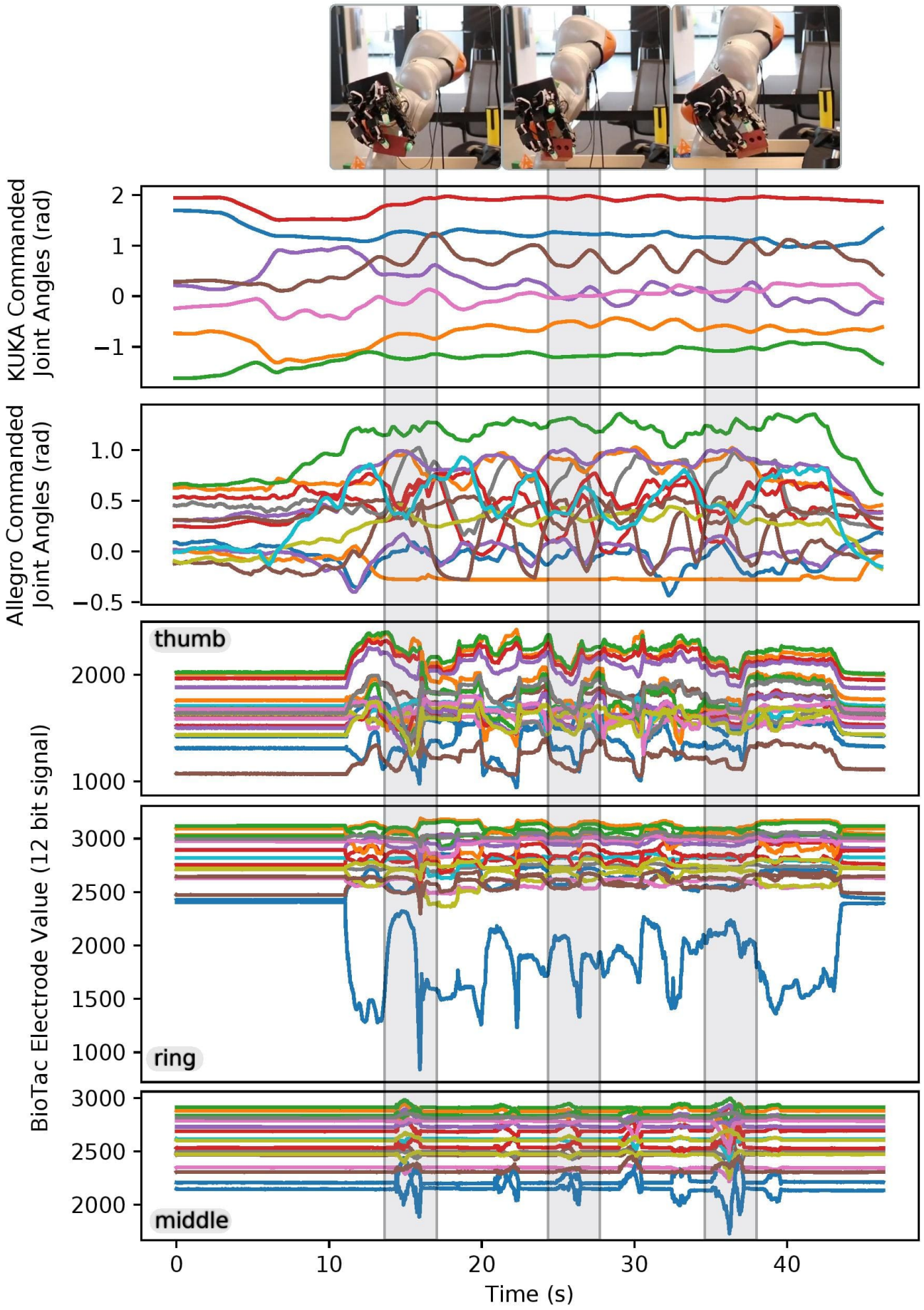

Fig. 16. BioTac tactile signals and robot joint position commands during the brick gaiting task where the middle finger makes the contact with the brick a total of 7 times over a 40 second duration in order to rotate it by 180 degrees. The 7 contacts made are also evident in the BioTac signals of the middle finger. The thumb and the ring finger remain pinched in order to hold the brick in hand.

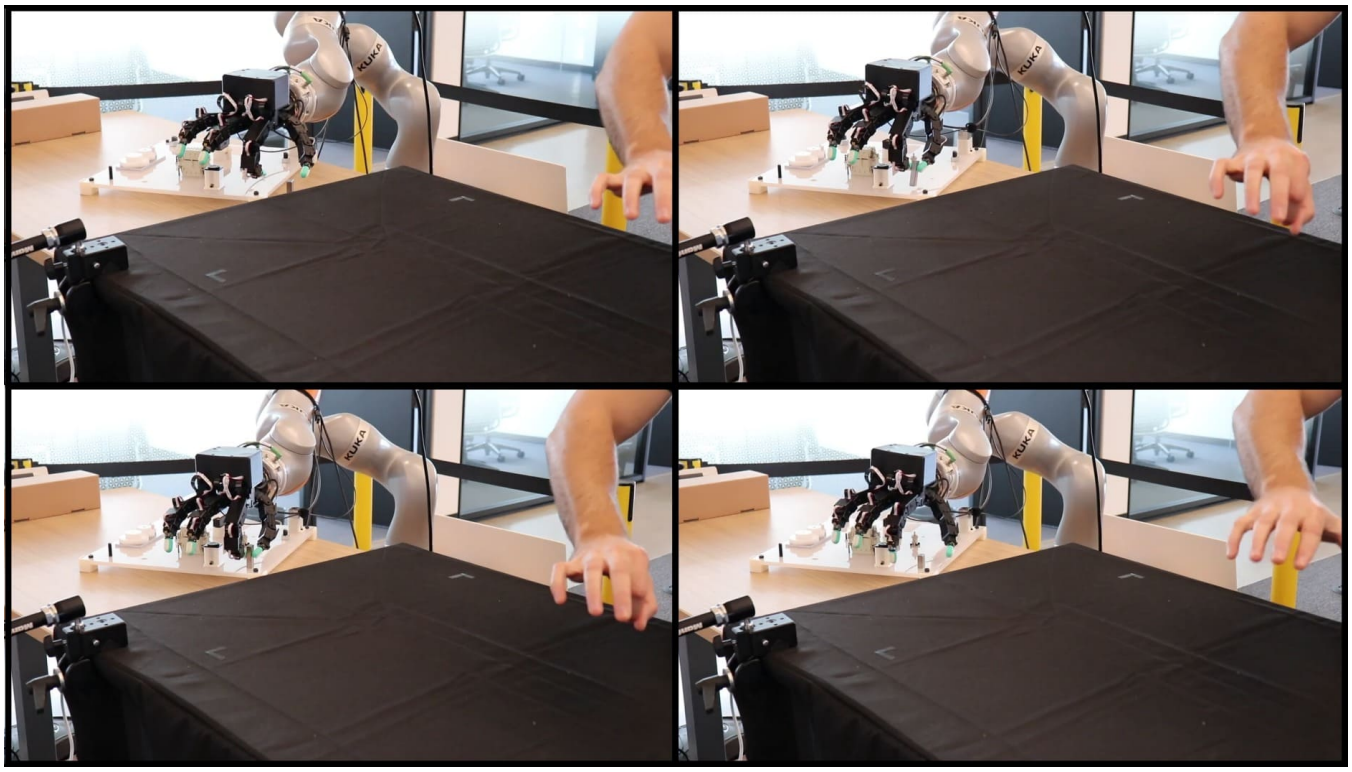

Fig. 17. NIST task board peg-in-hole insertion. The peg dimensions are 16 mm × 10 mm × 49.5 mm with the hole clearance of 0.1 mm.

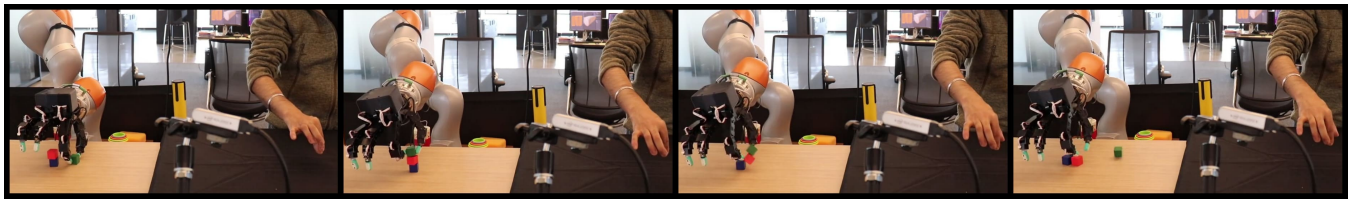

Fig. 18. The projection scheme used in the kinematic retargeting that compensates for inaccuracies in the hand tracking can make releasing small objects from the fingers difficult leading to losing objects often.

2initial activities on-board the iss," in *2012 IEEE Aerospace Conference*. IEEE, 2012, pp. 1–12.

[3] J. R. Sterbis, E. J. Hanly, B. C. Herman, M. R. Marohn, T. J. Broderick, S. P. Shih, B. Harnett, C. Doarn, and N. S. Schenkman, "Transcontinental telesurgical nephrectomy using the da vinci robot in a porcine model," *Urology*, vol. 71, no. 5, pp. 971–973, 2008.

[4] K. Z. Zhuang, N. Sommer, V. Mendez, S. Aryan, E. Formento, E. DAnna, F. Artoni, F. Petrini, G. Granata, G. Cannaviello, W. Raffoul, A. Billard, and S. Micera, "Shared human–robot proportional control of a dexterous myoelectric prosthesis," *Nature Machine Intelligence*, 2019.

[5] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *ICRA*. IEEE, 2018, pp. 1–8.

[6] "HaptX," https://haptx.com/.

[7] "CyberGlove Systems," www.cyberglovesystems.com.

[8] B. Stenger, A. Thayananthan, P. H. Torr, and R. Cipolla, "Model-based hand tracking using a hierarchical bayesian filter," *PAMI*, 2006.

[9] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM ToG*, 2014.

[10] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, "BigHand2.2m benchmark: Hand pose dataset and state of the art analysis," in *CVPR*, 2017.

[11] L. Ge, Z. Ren, and J. Yuan, "Point-to-point regression pointnet for 3d hand pose estimation," in *ECCV*, 2018.

[12] R. S. Johansson and J. R. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," *Nature Reviews Neuroscience*, vol. 10, no. 5, p. 345, 2009.

[13] B. Dorner, "Chasing the colour glove: Visual hand tracking," Ph.D. dissertation, Simon Fraser University, 1994.

[14] C. Theobalt, I. Albrecht, J. Haber, M. Magnor, and H.-P. Seidel, "Pitching a baseball: tracking high-speed motion with multi-exposure images," in *ACM TOG*, 2004.

[15] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM ToG*, 2009.

[16] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands deep in deep learning for hand pose estimation," *arXiv preprint arXiv:1502.06807*, 2015.

[17] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1812.08008*, 2018.

[18] M. Oberweger, P. Wohlhart, and V. Lepetit, "Generalized feedback loop for joint hand-object pose estimation," *IEEE transactions on pattern analysis and machine intelligence*,

2019.

[19] S. Li, X. Ma, H. Liang, M. Görner, P. Ruppel, B. Fang, F. Sun, and J. Zhang, "Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network," in *ICRA*, 2019.

[20] D. Antotsiou, G. Garcia-Hernando, and T.-K. Kim, "Task-oriented hand motion retargeting for dexterous manipulation imitation," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshop*, 2018.

[21] V. Kumar and E. Todorov, "MuJoCo HAPTIX: A virtual reality system for hand manipulation," in *Humanoids*, 2015.

[22] R. S. Johansson and G. Westling, "Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects," *Experimental brain research*, vol. 56, no. 3, pp. 550–564, 1984.

[23] R. S. Johansson and Å. B. Vallbo, "Tactile sensory coding in the glabrous skin of the human hand," *Trends in neurosciences*, vol. 6, pp. 27–32, 1983.

[24] S. Han, B. Liu, R. Wang, Y. Ye, C. D. Twigg, and K. Kin, "Online optical marker-based hand tracking with deep labels," *ACM TOG*, 2018.

[25] H. Liu, Z. Zhang, X. Xie, Y. Zhu, Y. Liu, Y. Wang, and S.-C. Zhu, "High-fidelity grasping in virtual reality using a glove-based system," in *ICRA*, 2019.

[26] H. Liu, X. Xie, M. Millar, M. Edmonds, F. Gao, Y. Zhu, V. J. Santos, B. Rothrock, and S.-C. Zhu, "A glove-based system for studying hand-object manipulation via joint pose and force sensing," in *IROS*, 2017.

[27] T. Schmidt, R. A. Newcombe, and D. Fox, "Dart: Dense articulated real-time tracking." in *RSS*. IEEE, 2014.

[28] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *ACM SIGGRAPH Asia*, 2017.

[29] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid, "Learning joint reconstruction of hands and manipulated objects," in *CVPR*, 2019.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *CoRR*, vol. abs/1706.02413, 2017.

[32] R. Villegas, J. Yang, D. Ceylan, and H. Lee, "Neural kinematic networks for unsupervised motion retargetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8639–8648.

[33] S. Sundaram, P. Kellnhofer, Y. Li, J.-Y. Zhu, A. Torralba, and W. Matusik, "Learning the signatures of the human grasp using a scalable tactile glove," *Nature*, vol. 569, no. 7758, p. 698, 2019.

[34] K. Van Wyk, M. Culleton, J. Falco, and K. Kelly, "Comparative peg-in-hole testing of a force-based manipulation controlled robotic hand," *IEEE Transactions on Robotics*, 2018.

[35] D. Kraft, "A software package for sequential quadratic programming," *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[36] ——, "Algorithm 733: Tomp–fortran modules for optimal control calculations," *TOMS*, 1994.

[37] S. G. Johnson, "The nlopt nonlinear-optimization package," http://github.com/stevengj/nlopt.

[38] "Orocos kinematics and dynamics library," https://github.com/orocos/orocos_kinematics_dynamics.

[39] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv preprint arXiv:1801.02854*, 2018.

[40] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "Rmpflow: A computational graph for automatic motion policy generation," *arXiv preprint arXiv:1811.07049*, 2018.

[41] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," *arXiv preprint arXiv:1903.01973*, 2019.

[42] B. Sundaralingam, A. S. Lambert, A. Handa, B. Boots, T. Hermans, S. Birchfield, N. Ratliff, and D. Fox, "Robust learning of tactile force estimation through robot interaction," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[43] K. Van Wyk, J. Falco, and E. Messina, "Robotic grasping and manipulation competition: Future tasks to support the development of assembly robotics," in *Robotic Grasping and Manipulation Challenge*. Springer, 2016, pp. 190–200.

[44] N. I. of Standards and Technology, "Robotic grasping and manipulation for assembly," https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assembly.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[46] R. Zhang, "Making convolutional networks shift-invariant again," *ICML*, 2019.

[47] "imgaug," https://github.com/aleju/imgaug.

[48] Y. Vardi and C.-H. Zhang, "The multivariate l1-median and associated data depth," *Proceedings of the National Academy of Sciences*, 2000.

XII. Appendix

A. GloveNet: Hand Tracking with Colour Glove

Hand tracking with glove is done via keypoint detection with neural networks. The user wears a black glove with coloured blobs and moves the hand on a table covered with black cloth *i.e.* the scene is instrumented in a way that aids hand tracking. Since the colours are distinct and that most of the background is black, we can use OpenCV HSV colour thresholding to generate annotations for these coloured blobs. The HSV thresholds vary with the time of the day and therefore we collect data across days to build a big dataset of 50K images. We use a neural network to fit this data which makes the whole process robust to lighting changes and bad annotations and avoids the burden on the user to find the appropriate thresholds at test time. The network, called GloveNet, uses 4 layers of ResNet-50 [45] with spatial-softmax at the end to regress to the 2D locations of finger-tips. We choose the recently proposed anti-aliased ResNet-50 from [46] for accurate and consistent predictions. We explain various stages of the pipeline below.

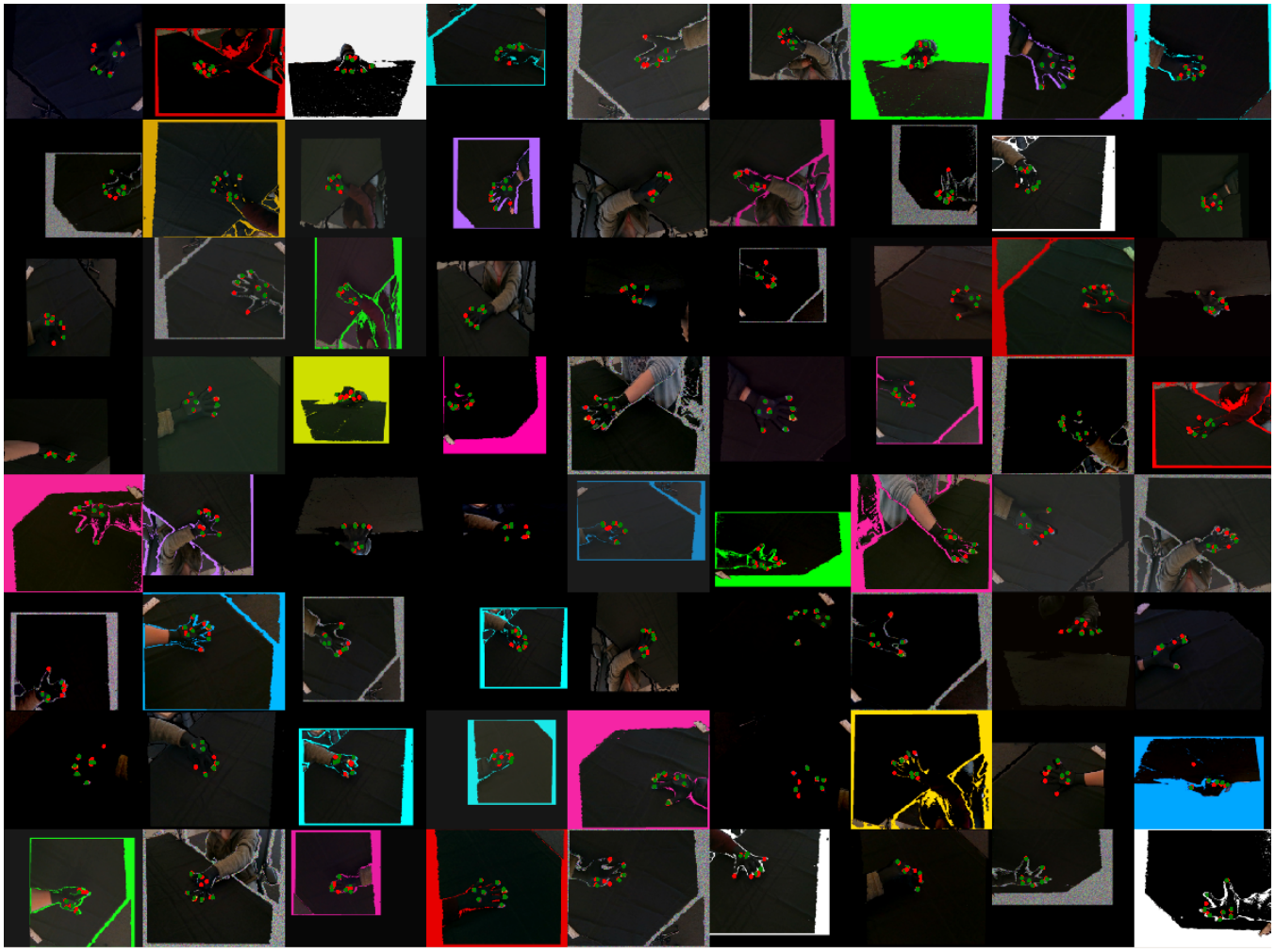

Fig. 19. GloveNet is trained with enough data augmentation to allow for accurate and stable keypoint detection of the coloured blobs. The green points represent the predicted while the red keypoints represent the ground truth keypoints. This snapshot is taken while the network is still training. The network is trained on images with annotations equal to or more than 4. Some ground truth annotations are missing due to occlusions or depth inconsistency check across the 4 views.

a) Data Augmentation: We use `imgaug` [47] and apply various data augmentations while training. Additionally, because we want to focus on the hand moving on the table, for each training image we set the colour values of all pixels with depth beyond a threshold to zero. At training time, we either fill these zeroed out values with any of the colours on the glove or leave the image unchanged based on random number generator. We also replace these zeroed out values random noise based on some probability. All this ensures that the network learns to ignore the colours in the background that look similar to the colours on the glove.

b) Confidence Estimation of Predictions: We also obtain confidence for each predicted finger-tip location using test-time augmentation (TTA). We generate new images by shifting the original image by random shifts and pass them all through the network in one batch. We then subtract the applied random shifts from the predicted positions for each image to bring

them into the reference frame of the original image and average them out to obtain the mean and standard deviation. We use the standard deviation as our confidence measure. We also use this to clean up the ground truth data that is noisy.

c) Outlier Rejection: At test time, we generate four randomly shifted images per camera image and a combined total of 16 images from all four cameras. We compute the predicted finger-tip locations and their confidence measures and discard those that have low confidence. Of the confident ones we compute the euclidean distances d_i between them and the previous finger-tip locations and turn them into probabilities p_i via softmax:

$$\mathsf{p}_i = \frac{\exp(-\alpha(\mathsf{d}_i - \min_i \mathsf{d}_i))}{\sum_{i=0}^{\mathsf{N}} \exp(-\alpha(\mathsf{d}_i - \min_i \mathsf{d}_i))}$$

We then push the predicted locations that have probability $\mathsf{p}_i > 0.2$ in a rolling buffer and compute the geometric median [48] to obtain the final predicted location of the finger-tip in 3D. The hyper-parameter $\alpha = 500$.

layer name	output size	parameters
input	320×240	-
conv1	160×120	7×7, 64, stride 2 3×3 max pool, stride 2
conv2	80×60	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix}\times3$
conv3	40×30	$\begin{bmatrix} 1\times1,\ 128 \\ 3\times3,\ 128 \\ 1\times1,\ 512 \end{bmatrix}\times4$
conv_transpose	80×60	3×3, 8
spatial_softmax	8×2	β=50

TABLE III. Architectures for GloveNet. Downsampling in conv2 and conv3 is performed with a stride of 2. We regress to 8 keypoint locations — 5 keypoints for fingers and 3 on the back of the palm to obtain the hand pose. We scale the predicted keypoint locations by a factor of 4 to obtain the results for 320×240 resolution image. The pre-trained weights come from anti-aliased version of ResNet-50 as done in [46]. The β is softmax temperature.

We found that while the predictions of the blobs at the back of the palm were stable, the predictions of finger-tips blobs tended to be quite inconsistent across time. Since the annotations were generated by computing the center of mass (CoM) of the segmented blob using the HSV colour thresholding in OpenCV, the CoM of the finger-tip were somewhat inconsistent across frames due to occlusions. Therefore, we relied only on the hand pose estimate provided by the blobs at the back of the palm.

B. Architecture for Hand Pose Estimation with PointNet++

The PointNet++ implementation we used in this paper is from `https://github.com/sshaoshuai/Pointnet2.PyTorch`.

layer name	mlp features	radius	num points
SA_1	[3, 64, 64, 128]	0.2	2048
SA_2	[128, 128, 128, 256]	0.4	1024
SA_3	[256, 128, 128, 256]	0.8	512
SA_4	[256, 128, 128, 256]	1.2	256
FP_4	[256+256, 256, 256]		512
FP_3	[256+256, 256, 256]		1024
FP_2	[256+128, 256, 256]		2048
FP_1	[256+3, 256, 256]		8192

TABLE IV. The architecture is composed of 4 set abstraction layers, SA_i and 4 feature propagation layers, FP_j. The set abstraction layer sub-samples the points while the feature propagation layer interpolates features at a higher resolution.

a) Set Abstraction: A set abstraction level takes $N \times (d + C)$ as input of N points with d-dim coordinates and C-dim point feature. It outputs tensor of $N' \times (d + C')$ where N' sub-sampled points with d-dim coordinates and new C'-dim feature vectors summarise local context.

b) Feature Propagation: In a feature propagation level, point features are propagated from $N_i \times (d + C)$ points to N_{i-1} points where N_{i-1} and N_i (with $N_i \leq N_{i-1}$) are point set size of input and output of set abstraction level i. It is achieved by interpolating feature values of N_i points at coordinates of the N_{i-1} points. The interpolated features on N_{i-1} points are then concatenated with skip linked point features from the set abstraction level.

c) Predicting Keypoint Locations: The backbone of the hand pose estimation is PointNet++ architecture which returns an embedding, f, of size $N \times C$. Different MLPs are used to map this embedding to the corresponding desired outputs.

$$
\begin{aligned}
z &= \texttt{mlp\_layer1}(f) \\
\delta_{xyz} &= \texttt{voting}(z) \\
\texttt{coords} &= \texttt{input}_{xyz} + \delta_{xyz}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{JointMask}_{xyz} &= \texttt{sigmoid}(\texttt{seg}(z)) \\
\texttt{HandSeg}_{xyz} &= \texttt{cls}(z) \\
\texttt{HandSegProb}_{xyz} &= \texttt{sigmoid}(\texttt{HandSeg}_{xyz})
\end{aligned}
$$

$$
\begin{aligned}
\texttt{weights} &= \texttt{HandSegProb}_{xyz} \cdot \texttt{JointMask}_{xyz} \\
\texttt{Keypoints} &= \frac{\sum \texttt{weights} \cdot \texttt{coords}}{\sum \texttt{weights}}
\end{aligned}
$$

layer name	parameters
mlp_layer1	[256, 256, 256]
voting	[256, 23×3]
seg	[256, 23]
cls	[256, 2]

TABLE V. Various MLPs used to map embedding to the corresponding outputs.

The `voting` layer obtains the relative positions of the 23 keypoints with respect to each point. The `seg` layer obtains the masks for each keypoint *i.e.* the neighbourhood of points that contribute to the location of a keypoint. The `HandSeg` layer segments hand from the background. We use Euclidean losses for both `voting` as well as `Keypoints` while a sigmoid cross-entropy is used for `HandSeg`.

C. Architecture for JointNet

The 23×3 keypoint locations are unrolled to a 69-dimensional vector before feeding to the JointNet which returns a 20-dimensional vector of joint angles. Of all the hand-designed architectures we tried, we found this particular architecture to be an optimal trade-off between accuracy and efficiency.

layer name	parameters
linear1	69×128
linear2	128×256
linear3	256×20

TABLE VI. The JointNet architecture is comprised of three layers. The layers linear1 and linear2 also use BatchNorm1d and ReLU.

D. Completion Times Over 5 Consecutive Trials

We show the completion times for the 5 consecutive trails for each of the tasks. The failed trial is denoted by F. Note that the for most of the trials, the pilot only used 3-4 training trails to warm up. These 5 consecutive trails allow for testing both the ability to carry out a certain task without getting tired as well as showcasing that the tracking works without failures. Admittedly, the performace can vary depending on the pilot and how they are feeling on a given day but our experiments have revealed that the performance is in general quite similar.

Task	Pilots	Completion Times for 5 Consecutive Trials(s)					Mean	Std.
Pick and Place: Brick	Pilot 1	19	16	17	11	18	16	3.11
	Pilot 2	22	22	19	16	14	19	3.57
Pick and Place: Spam	Pilot 1	28	14	15	16	23	19	6.05
	Pilot 2	23	23	28	29	20	25	3.78
Card Sliding	Pilot 1	27	26	32	38	35	32	5.12
	Pilot 2	18	12	18	15	17	16	2.54
Pick and Place: Pringles	Pilot 1	50	18	20	29	18	27	13.6
	Pilot 2	25	53	29	36	63	41	16.22
Brick Gaiting	Pilot 1	48	67	F	F	58	58	9.50
	Pilot 2	37	44	F	F	28	36	8.02
Pouring	Pilot 1	38	42	32	F	28	35	6.21
	Pilot 2	73	56	62	50	57	60	8.61
Box Flip	Pilot 1	51	39	45	F	77	53	16.73
	Pilot 2	174	26	90	30	67	77	60.18
Blocks (L)	Pilot 1	41	49	54	45	165	71	52.87
	Pilot 2	53	93	79	43	61	66	20.12
Peanut Jar	Pilot 1	89	66	79	77	75	77	8.25
	Pilot 2	68	105	84	87	57	80	18.45
Cup Insertion	Pilot 1	64	94	70	F	71	75	13.2
	Pilot 2	125	F	124	124	112	121	6.18
Tea	Pilot 1	48	115	170	58	154	109	55.00
	Pilot 2	54	48	99	105	213	104	66.22
Blocks (M)	Pilot 1	179	278	64	80	298	180	108.37
	Pilot 2	99	48	82	75	152	91	38.63
Wallet	Pilot 1	105	66	195	96	63	105	61.82
	Pilot 2	321	92	328	100	218	212	114.36
Blocks (S)	Pilot 1	136	371	169	F	484	290	165.88
	Pilot 2	113	89	69	117	67	91	23.57
Container	Pilot 1	442	271	375	297	405	358	72.18
	Pilot 2	189	212	258	238	243	228	27.39