The International Journal of Robotics Research
2024, Vol. 43(4) 513–532
© The Author(s) 2024
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649241227559
journals.sagepub.com/home/ijr
Sage

Learning dexterity from human hand motion in internet videos

Kenneth Shaw, Shikhar Bahl, Aravind Sivakumar, Aditya Kannan and Deepak Pathak

Abstract

To build general robotic agents that can operate in many environments, it is often useful for robots to collect experience in the real world. However, unguided experience collection is often not feasible due to safety, time, and hardware restrictions. We thus propose leveraging the next best thing as real world experience: videos of humans using their hands. To utilize these videos, we develop a method that retargets any 1st person or 3rd person video of human hands and arms into the robot hand and arm trajectories. While retargeting is a difficult problem, our key insight is to rely on only internet human hand video to train it. We use this method to present results in two areas: First, we build a system that enables any human to control a robot hand and arm, simply by demonstrating motions with their own hand. The robot observes the human operator via a single RGB camera *and imitates their actions* in real-time*. This enables the robot to collect real-world experience safely using supervision. See these results at https://robotic-telekinesis.github.io. Second, we retarget in-the-wild human internet video into task-conditioned pseudo-robot trajectories to use as artificial robot experience. This learning algorithm leverages action priors from human hand actions, visual features from the images, and physical priors from dynamical systems to pretrain typical human behavior for a particular robot task. We show that by leveraging internet human hand experience, we need fewer robot demonstrations compared to many other methods. See these results at https://video-dex.github.io*

Keywords

Learning from internet video, dexterous robot hands

Received 24 February 2023; Revised 1 November 2023; Accepted 17 December 2023
Senior Editor: Jose-Luis Blanco
Associate Editor: Shoudong Huang

1. Introduction

The long-standing dream of many roboticists is to see robots autonomously perform diverse tasks in diverse environments. To build a robot that can operate anywhere, many methods rely on successful robotic interaction data to train on. However, deploying inexperienced, real-world robots to collect unrestricted real world experience may require constant supervision which is infeasible. This poses a chicken-and-egg problem for robot learning because to collect successful experience safely, the robot already needs to be experienced. How do we get around this issue?

Instead, we would like to side-step the data collection-training loop and use human hand video experience to safely control a robot hand or aid it in learning. This insight of leveraging human videos to aid robotics is not new and has seen immense attention from the vision community at large (Damen et al., 2018; Goyal et al., 2017; Grauman et al., 2022). However, most of this prior work tends to use human data as a mechanism for pretraining just the visual representation (Nair et al. 2018, 2022; Pinto et al., 2016; Sermanet et al., 2018; Xiao et al., 2022), much like how deep learning has been used as a pretraining tool in related areas of computer vision (Chen et al., 2020; He et al., 2017) and natural language processing (Brown et al., 2020; Devlin et al., 2018).

Our key idea is that we would like to watch the motion of human hands and directly use that trajectory on the robot. We call the method of detecting human motion from video and converting it into robot trajectories for downstream control or learning applications retargeting. However, retargeting human actions to robot actions is difficult because there is a large embodiment gap and extracting 3D actions

Carnegie Mellon University, Pittsburgh, PA, USA

Corresponding authors:
Kenneth Shaw, Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA.
Email: kshaw2@andrew.cmu.edu

Deepak Pathak Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA.
Email: dpathak@cs.cmu.edu

from 2D video is a severely under-constrained problem. Our key insight is to leverage the vast corpus of human hand poses from passive data on the web to train a retargeting system from human pose to robot pose as seen in Figure 1. This neural network learns to map across the large embodiment gap between human and robot as described in Handa et al. (2020) and Sivakumar et al. (2022). It uses unpaired data of humans using their hands, and uses this to learn how to control a robot hand. This method unlocks real-time control and policy training from many different types of human videos including third person or first person sources.

To experiment with this retargeting method, we first investigate using 3rd person human video information in real-time. The objective of this experiment is to enable teleoperation of a dexterous robotic hand, *in the wild*. This means our system should be low-cost, work for any untrained operator in any environment, and use only a single uncalibrated color camera. One should be able to simply look into a monocular camera of their phone or tablet and control the robot without relying on any bulky motion capture or multi-camera rigs for accurate 3D estimation. We call our system *Robotic Telekinesis* as it provides a human the ability to control a dexterous robot from a distance without any physical interaction as in Figure 2. This builds on work published at RSS 2022 by Sivakumar et al., (2022).

Second, we apply this retargeting method to translate egocentric human videos into pseudo-robot experience for pretraining robot policies. We clip internet videos where humans are performing actions similar to the desired robot actions. Next, we retarget the video into the robot hand embodiment to use as pseudo-robot experience. Finally, we pretrain policies to act sensibly for many tasks without any real robot experience, only synthetic experience generated from human video. The priors are able to use human video information to pretrain policies. To fine tune policies, we use the aforementioned 3rd person human video and teleoperation to bridge the gap between the internet human hand embodiment and the robot hand embodiment. We call this system *Videodex* as it learns dexterity from video. This builds on work published at CoRL 2022 by Shaw et al., (2023b).

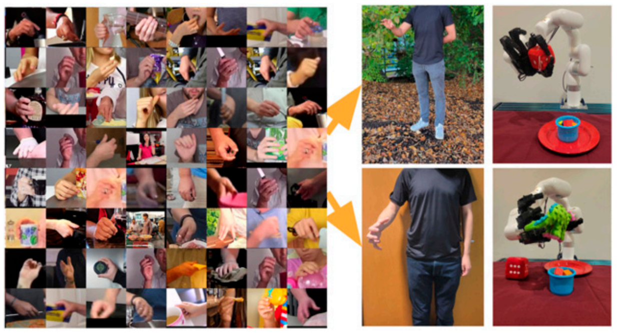
Figure 1. Our system leverages passive data from the internet to enable robotic real-time imitation of human motion. This low-cost system does not require any special gloves, mocap markers or even camera calibration and works from a single RGB camera. The left image depicts a random sample of grasps generated from internet data.

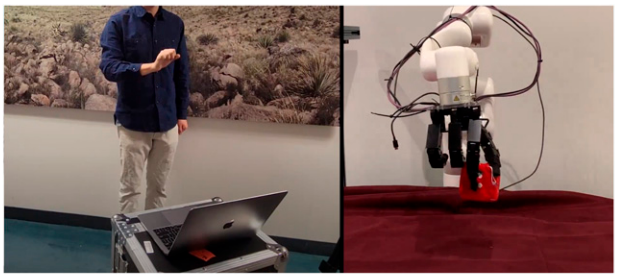
Figure 2. An operator completing a dice pickup task while watching the robot through a video conference. Video demos are at https://robotic-telekinesis.github.io/.

2. Related work

2.1. Learning action from videos

Detecting humans, estimating poses of different body parts, and understanding the dynamics and interactions related to human motion are commonly studied problems. One can model human hands using the MANO (Romero et al., 2017) parameterization as well as the human body using SMPL, SMPL-X (Loper et al., 2015; Pavlakos et al., 2019) models. There are many efforts in human pose estimation such as (Kanazawa et al., 2017; Rong et al., 2021; Wang et al., 2020). We focus on FrankMocap (Rong et al., 2021) for our project as it is robust for online videos and provides good hand-only estimation. Additionally, using these detectors and watching humans can be powerful for controlling computers. Traditionally, teleoperation approaches have employed hand markers with gloves (Fang et al., 2019) for motion capture (Han et al., 2018) or VR settings (Kumar and Todorov 2015; Mannam et al., 2023). Without gloves, Li et al. (Li et al., 2019) used depth images and a paired human-robot dataset for teleoperating the Shadow Hand, and Dexpilot (Handa et al., 2020) designed a system that mimics the functional intent of the human operator to perform object manipulation tasks.

2.2. Kinematic retargeting and visual teleoperation

Human pose tracking only solves half of the visual teleoperation problem. Mapping human poses to robot poses is itself a difficult challenge because humans and robots have very different kinematic structures. Li et al. (2019) train a deep network to map human hand depth images to joint angles in the robotic Shadow Hand, and Antotsiou et al. (2018) combine inverse kinematics and Particle Swarm Optimization to retarget human hand poses to a high-dimensional robot hand model. Our system follows the method of DexPilot Handa et al. (2020), which minimizes a cost function that captures the functional similarity between a human and a robot hand.

The general problem of kinematically retargeting motion in one morphology into another is also studied outside of robotic manipulation. Villegas et al. (2018) propose a cycle consistency objective to transform motion between animated humanoid characters of different body shapes. Peng et al. (2020) use an approach based on keypoint matching to learn robotic locomotion behaviors from demonstrations of walking dogs. Zakka et al. (2021) learn a visual reward function that allows reinforcement learning agents to learn from demonstrators with different embodiments.

2.3. Learning from large-scale datasets

There have been many efforts to collect and curate datasets from internet human videos like FreiHand (Zimmermann et al., 2019) for hand poses, 100 Days of Hands (Shan et al., 2020) for hand-object interactions, Something-Something (Goyal et al., 2017) for semantically similar interactions, Human3.6M (Ionescu et al., 2013) and the CMU Mocap Database (Hodgins) for Human pose estimation, Epic Kitchens (Damen et al., 2018), ActivityNet datasets (Heilbron et al., 2015), or YouCook (Das et al., 2013) for action driven datasets. For dextrous manipulation, activity-based datasets contain well-labelled atomic tasks that we would like to solve.

2.4. Learning for dexterity

Human dexterity evolved with cognition in complementary fashion (Ma and Dollar 2011). This leads us to study dexterity to understand cognition in robot agents as well. To create dexterity, reinforcement learning (RL) with an engineered reward function can show good results in simulation (Kalashnikov et al., 2018; Levine et al., 2016) but requires lots of data, especially in high-dimensional dexterous manipulation. This requires simulators (Makoviychuk et al., 2021; Todorov et al., 2012), which cannot model physics (such as contact forces) or images properly, making it difficult to transfer to the real-world successfully. Agarwal et al. (2023) Imitation learning for manipulation can be safer and more sample efficient. Behavior cloning is a common approach to learning (Bojarski et al., 2016; Pomerleau 1988) and can work in the real world. DIME (Arunachalam et al., 2022) involves using nearest neighbor matching of the state or image representations of the scene with that of demonstrations to determine actions. Qin et al. (2022) propose a method for pick-and-place and opening door tasks that involves teleoperation and learning policies in simulation, followed by Sim2Real transfer. DexMV (Qin et al., 2021) uses collected human hand videos for imitation learning on a robot hand. Similarly, DexVIP and DEFT (Kannan et al., 2023; Mandikal and Grauman 2022) learn human hand-object affordance using curated internet video datasets and uses these priors as RL initialization.

2.5. Robot learning by watching humans

Recent works have leveraged human datasets to learn cost functions (Bahl et al., 2022; Chen et al., 2021; Mendonca et al., 2023; Shao et al., 2021), learn action correspondences (Schmeckpeper et al., 2020) both in a paired (Sharma et al., 2019) and unpaired manner (Smith et al., 2020). This data can also be used to extract explicit actions by leveraging structure in the collection (such as reacher-grabber tools (Young et al., 2020)) or prediction of future hand and object locations (Lee and Ryoo 2017), as well as keypoint detectors (Das et al., 2020). This can also be used to build representations for robot learning (Bahl et al., 2023; Nair et al., 2022; Pari et al., 2021). R3M (Nair et al., 2022) trains on the Ego4D Grauman et al. (2022) dataset using a temporal alignment loss between language labels and video frames. We build on top of previous efforts in this area, where we combine visual representations trained on human activity data, with *action* driven representations.

3. Learning robot motion from human video

In this section, we establish how to learn and gather 3D information of the human's trajectory from watching 2D human video. We break this retargeting process from video into two pipelines: the hand and the arm. First, how do we find robot hand finger joint configurations to match the human hand finger configurations? Our key insight is to use internet videos as training to learn how to map human hands to robot hands, without even using paired data of robot hands and human hands. Second, how do we find robot arm configurations that match the human arm/wrist configurations? Here, we rely on 3D model reconstructions of the body to find the wrist with respect to the camera or the body of the human. We generalize our method to utilize both 3rd person video and egocentric video and show experiments using both of these pipelines.

3.1. Hand pose energy function

3.1.1. 3D human hand pose estimation from 2D images. The first step in hand retargeting is to detect the operator's hand in a 2D image and infer its 3D pose. While the problem of inferring a 3D hand pose from a 2D image is inherently under-constrained, we can leverage priors from offline data and learn to accurately estimate physically plausible hand poses. Thanks to recent advances in computer vision, there are several paired 2D-3D datasets, and several methods which use these datasets to train high-quality 3D human pose estimators that operate on 2D images.

To leverage such prior methods, we need to crop the image to focus on the human hand. We first compute a "crop" around the operator's hand, based on a bounding box computed using an off-the-shelf detector derived from OpenPose (Cao et al. (2019)). This 2D body skeleton detector is run over the entire image and outputs the predicted pixel locations for each of the 18 keypoints on the skeleton. The tight bounding rectangle around the points is then computed, and a fixed padding is applied on all sides to allow a margin of error. The resulting image crop goes to a pose estimator from FrankMocap Rong et al. (2021) to obtain hand shape and pose parameters of a 3D MANO model Romero et al. (2017) of the operator's right hand. See the "OpenPose" and "FrankMocap" modules in the top row of Figure 3 for a graphical depiction of this phase of the pipeline.

A key point to note is our human hand pose estimation module works for *any human operator*, with *any uncalibrated camera* in *any environment*. By utilizing pretrained state-of-the-art neural network detectors and pose estimators, we indirectly leverage the millions of images these models were trained on. These images depict human hands in many poses against many backgrounds, and as a result, our system can be used out of the box for anyone.

3.1.2. 3D Human hand to robot hand pose. The next step is to retarget the estimated 3D human hand pose to a vector of 16 joint angles on the hand that place the robot's hand in an analogous hand pose (see the third panel on the top branch of Figure 3). This has three challenges:

- Underconstrained: The Allegro hand and the human hand have very different embodiments, and differ greatly in shape, size, and joint structure. This means the mapping between a human hand pose and a robot hand pose is not bijective; there could be multiple robot poses that can correspond to a certain human pose, and vice-versa. This issue is further magnified for high DoF dexterous hand manipulators. This makes it challenging to design a mapping from a human hand pose to a robot hand pose.
- Robustness: Our solution must work for *any human operator* trying to perform *any kind of task* in *any environment.* Notably, this means we cannot bias solutions toward any particular type of motion, or hand type.
- Efficiency: We require our solution to be real-time at inference without any lag. Empirically, we observe this means that the robot must be able to follow the human at least 15 Hz to solve tasks successfully via teleoperation.

A natural way to address these three challenges would be to train a model on a diverse dataset of paired human-robot hand pose examples (similar to how the 3D human hand estimation in Section 3.1.1 was trained on paired examples of images and ground-truth target hand poses). However, collecting such a paired dataset at large scale would be prohibitively expensive, and furthermore, a new dataset would have to be collected for each type of robot hand. To get around this issue, we train a deep human-to-robot hand retargeter network in a way that uses just the human data itself and does need any supervision for the target robot pose. The key idea is to formulate the human-to-robot mapping problem using a feasibility objective rather than a regression objective because the latter relies on ground-truth target robot poses which are not available. We define an energy optimization procedure that provides loose constraints on the robot hand poses the retargeter network should output for a given hand pose. Below, we describe the dataset used for training, and then the training procedure.

3.1.2.1. Dataset of YouTube videos of human interaction. We leverage a massive internet-scale dataset of human hand images and videos. We gather about 20 million images from the Epic Kitchens Damen et al. (2018) dataset, which captures egocentric videos of humans

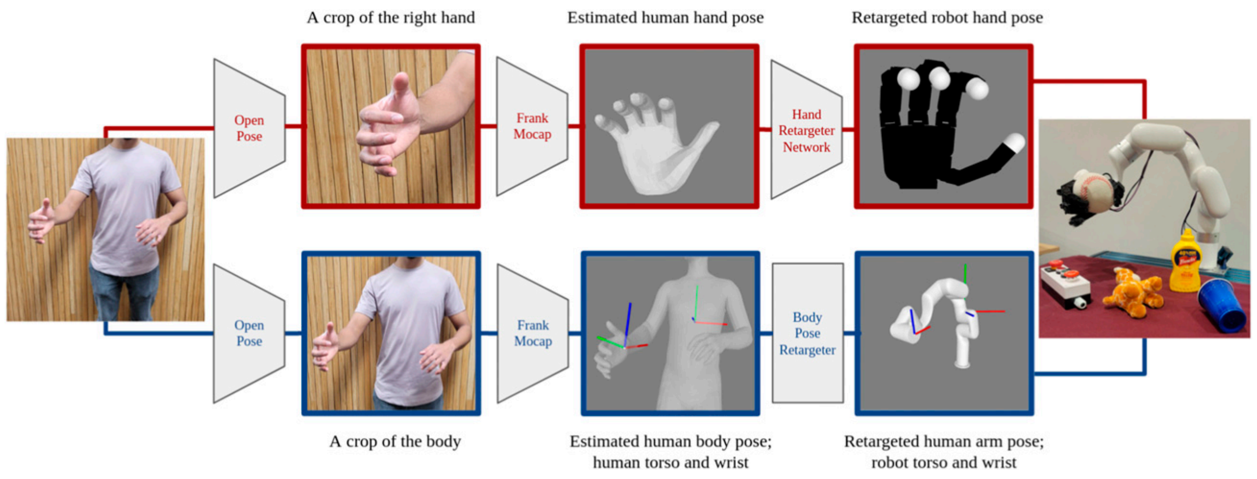
A crop of the right hand
Estimated human hand pose
Retargeted robot hand pose
Open Pose
Frank Mocap
Hand Retargeter Network
Open Pose
Frank Mocap
Body Pose Retargeter
A crop of the body
Estimated human body pose; human torso and wrist
Retargeted human arm pose; robot torso and wrist

Figure 3. A graphical description of our visual teleoperation pipeline from Robotic Telekinesis. First, a color camera captures an image of the operator. Top: to command the robot hand, a crop of the operator's hand is passed to a hand pose estimator, and the hand retargeting network maps the estimated human hand pose to a robot hand pose. Bottom: to command the robot arm, a crop of the operator's body is passed to a body pose estimator and cross-body correspondences are used to determine the desired pose of the robot's end-effector from the estimated human body pose. Finally, commands are sent to both the robot hand and arm.

performing daily household tasks, and the 100 Days of Hands Shan et al. (2020) dataset, which is a collection of YouTube videos depicting a wide variety of human hand activities. We run the hand pose estimator from Rong et al. (2021) (the same one we use at deployment time in our pipeline) to estimate human hand poses for each image frame in these videos. We augment this massive noisy dataset of estimated human hand poses with the small and clean FreiHand dataset Zimmermann et al. (2019), which contains ground-truth human hand poses for a diverse collection of realistic hand configurations.

3.1.2.2. Retargeter network. Our hand retargeter network is a Multi-Layer Perceptron (MLP), f (.), with two hidden layers. It takes as input a human hand pose (a vector $x \in \mathbb{R}^{55}$ that denotes the MANO hand shape and pose parameters) and outputs an Allegro hand pose $y \in \mathbb{R}^{16}$ (a vector of Allegro joint angles). At inference time, we simply pass the estimated hand shape/pose vector to the network, which directly outputs Allegro joint angles that we can command to the robot as shown in Figure 4. A key benefit of using a neural network is speed: the network's forward pass takes about 3 ms (333 Hz)—this is critical for smooth real-time teleoperation.

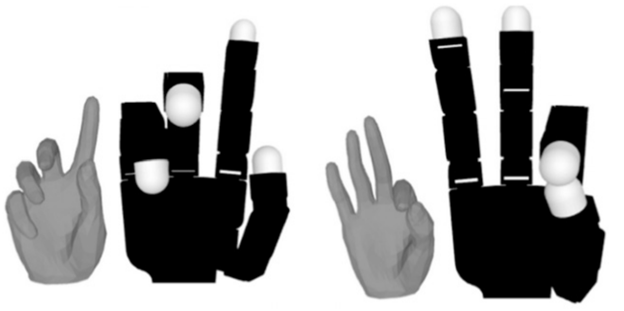
Figure 4. Human-to-robot Translations. The inputs and outputs of our hand retargeting network. Each of the pairs depicts a human hand pose and the retargeted Allegro hand pose.

3.1.2.3. Optimization via energy minimization. Our dataset contains millions of human hand poses, but no ground-truth target robot poses to regress onto. In most neural network setups, at training time, the network has access to *paired* examples $(x \in \mathcal{X}, y \in \mathcal{Y})$, where $\mathcal{X}$ is the source domain, and $\mathcal{Y}$ is the target domain. In our case, the source domain $\mathcal{X}$ is the set of all human hand poses and the target domain $\mathcal{Y}$ is the set of all robot hand poses, *but we only have training data from the source domain.* Hence, we cannot perform a direct regression. To get around this issue, we define an energy function $E\ (x, y)$ that captures the dissimilarity between the input 3D human hand pose x and the network's predicted robot hand pose y. Per convention, a high energy means the two poses are highly *dis*similar. Assuming that we have designed such an energy function, the neural network optimizes the following objective

$$\underset{f}{\operatorname{argmin}}\ \mathbb{E}_{x \in \mathcal{X}}\ [E(x, f(x))] \tag{1}$$

3.1.2.4. Energy function formulation. Our energy function is designed to capture the notion that for any given human hand pose, the optimal corresponding robot hand pose is the one that best mimics the *functional intent* of the human as visualized in Figures 5 and 6. For example, consider a human performing a pinch grasp with the tips of their thumb and index finger 1 cm apart, while simultaneously pressing a button with their outstretched ring fingertip positioned 15 cm away from the center of their palm. In order for the robot hand to effectively mimic this action, the robot's thumb and index fingertips should also be 1 cm apart, and the robot's ring fingertip should be 15 cm away from its palm center as seen in Figure 7.

This notion of functional intent is formalized by the energy function, which captures the degree of (dis)-similarity between a human hand pose and a robot hand pose. Following Handa et al. (2020), we define a set of five hand keypoints: the tips of the index, middle, ring and thumb

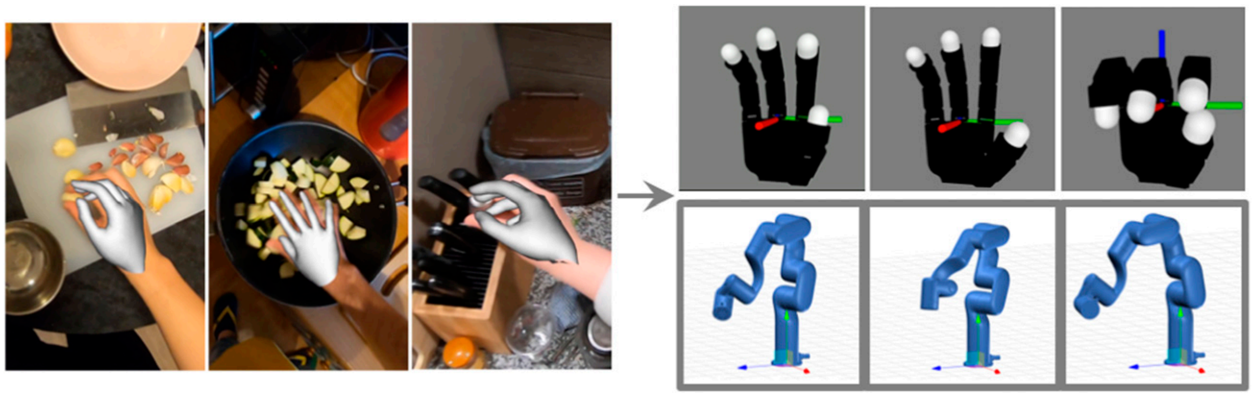
Figure 5. To use internet videos as pseudo-robot experience, we retarget human hand detections from the 3D MANO model (Romero et al., 2017) to 16 DoF robotic hand (LEAP) embodiment and we retarget the wrist from the moving camera to the xArm6 UFactory embodiment. First person video examples are at https://video-dex.github.io.

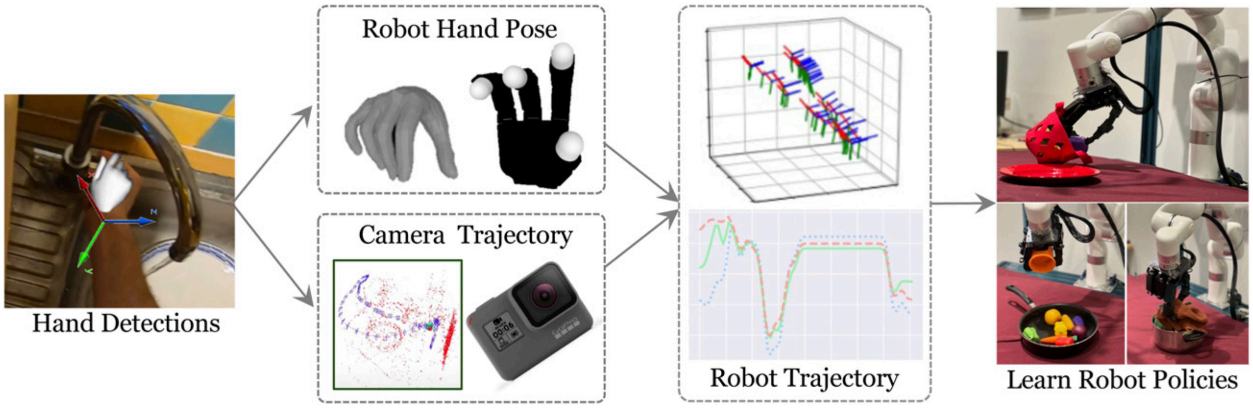

Figure 6. To use first or third person human videos as an action prior for training policies, we retarget them to the robot embodiment. The detected human fingers are converted to the robot fingers using a learned energy function. The wrist pose is retargeted using the hand detections and camera trajectory and transformed to the robot arm.

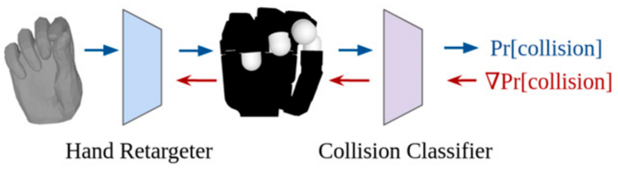

Figure 7. A trained self-collision classifier is used as an adversarial loss that penalizes self-colliding joint configurations. The blue arrows denote the forward pass, and the red arrows denote the flow of gradients during the backward pass.

fingers, and the center of the palm. We define a set of ten keyvectors, each of which connects a pair of keypoints. Each keyvector has one endpoint designated as the origin, and the other as the tip, and the vector is expressed in the coordinate system of the origin keypoint. Our energy function is a weighted sum of ten cost terms, where the i-th cost term measures how much the i-th keyvector on the posed human hand differs from the i-th key vector on the posed robot hand (for more details, see appendix and Figure 8 in Handa et al. (2020)).

3.1.2.5. Computing the keyvectors on the human hand. We now describe how to compute the keyvectors on the human hand, given the SMPL-X model parameters (β_h, θ_h). The first step is to use the SMPL-X model to generate a full posed 3D mesh of the human hand. Given β_h and θ_h (and a template hand mesh), the SMPL-X model generates a 3D mesh that correctly captures the shape and pose of the human hand. The next step is to transform these vertices into a canonical coordinate frame, centered at the palm center, with the positive x axis pointing out of the hand, the positive y axis pointing toward the thumb, and the positive z axis pointing toward the middle fingertip. This is done by applying a hand-coded transformation between the SMPL-X coordinate frame and the canonical coordinate frame. The next step is to compute the transformation between each of the keypoint coordinate frames and the canonical coordinate frame. This is done using the Kabsch–Umeyama Algorithm Umeyama (1991) for estimating the transformation that best aligns corresponding pairs of points. Concretely, for each keypoint, we manually determine four vertices on the template hand mesh: (1) the keypoint itself, (2) a vertex located along 0.05 m along the positive x axis from the keypoint, (3) a vertex located 0.05 m along the positive y axis from the keypoint, and (4) a vertex located 0.05 m along the positive z axis from the keypoint. This is precomputed once, up front. At runtime, for a given posed human hand mesh, we gather the 3D coordinates for each of these three points in the canonical coordinate frame. We define a corresponding set of four points: {[0, 0, 0], [0.05, 0, 0], [0, 0.5, 0], [0, 0, 0.5]}, which denote the coordinates of these points in the coordinate frame of the keypoint. Given these four correspondences, the Kabsch–Umeyama computes the transformation between the keypoint coordinate frame and the canonical coordinate frame that best aligns these corresponding point pairs.

3.1.2.6. Computing the keyvectors on the Allegro hand. Here, we describe how to compute the keyvectors on the Allegro hand, given a vector of Allegro hand joint angles. The key idea here is to exploit forward kinematics. The URDF of the Allegro hand defines the kinematic skeleton of the Allegro hand. The forward kinematics map takes as input a joint angle vector and outputs the transformation between each link's coordinate frame and the root coordinate frame. Each of our keypoints conveniently corresponds to a particular link on the Allegro hand, so the keypoint coordinate frames can simply be read off from the forward kinematics result.

Formally, the energy function between a human hand pose (parameterized by the tuple (β_h, θ_h)) and an Allegro hand pose (parameterized by the joint angles q_a) is computed as follows. First, for each $i \in \{1, ..., 10\}$, the i-th keyvector is computed on the human hand (call it $\mathbf{v_i^h}$) and the Allegro hand (call it $\mathbf{v_i^a}$). Then, each Allegro hand keyvector $\mathbf{v_i^a}$ is scaled by a constant c_i. The i-th term in the

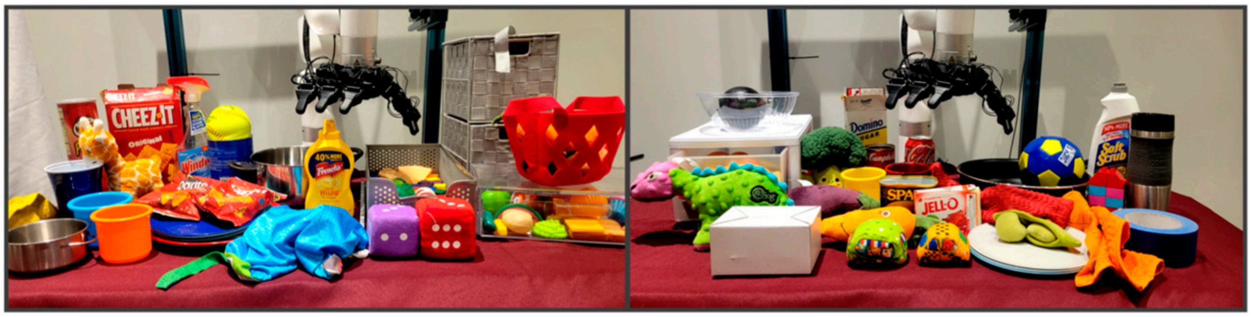

Figure 8. The collection of train objects (left) and test objects (right) used for experimentation in Videodex. Videodex utilizes retargeting from 1st person internet videos and 3rd person video for teleoperation and demonstration collection.

energy function is the Euclidean difference between $\mathbf{v_i^h}$ and $c_i \cdot \mathbf{v_i^a}$

$$E(\ (\beta_h, \theta_h), q_a\ ) = \sum_{i=1}^{10} \left\| \mathbf{v_i^h} - \left(c_i \cdot \mathbf{v_i^a}\right) \right\|_2^2 \tag{2}$$

where the scaling constants $\{c_i\}$ are hyperparameters. Critically, this energy function is a fully differentiable function of the Allegro joint angles because of the differentiability of the forward kinematics operation (which is a chain of sin/cos and matrix multiplications). This allows us to train the hand retargeter network f via gradient descent, using the energy function as a loss function. One should set each c_i to around 0.625 if the goal is to produce aesthetically appealing retargeted Allegro hand poses. If the goal is to maximize functional similarity, in theory, one should set each c_i to 1, to encourage perfect matching of each keyvector. In practice, we use a scaling constant of 0.8 for each of the finger-to-thumb and finger-to-finger keyvectors, and a scaling constant of 0.5 for each of the finger-to-palm keyvectors. This means that in order to ensure a stable grasp, operators must squeeze their fingers closer together than they normally would when grasping, but through our human-subject study, we find that novice operators quickly realize this and naturally adjust.

This energy optimization formulation has parallels to the classical Inverse Kinematics (IK). However, while the general IK problem is widely studied, our instantiation has a key difference. The prototypical IK problem solves for joint angles that achieve target fingertip poses relative to a fixed point in the palm. However, our end-effector constraints are all relative to each other, which makes it difficult to adopt traditional powerful IK solvers such as (e.g., Carpentier et al. (2019) or Buss and Kim (2005a)).

3.1.3. Collision avoidance via adversarial training. Using a neural network to perform human-to-robot hand retargeting has another subtler advantage over an online optimization approach: we can add terms that are slow to compute to our energy function. During training time of the neural network, we run expensive operations in order to compute the loss at each iteration. Then we can absorb the computation cost during training instead of incurring it at deployment time. This allows us to use any energy function, as long as it is differentiable.

We exploit this idea to address the problem of self-collisions. Minimizing the keyvector-similarity energy function described above can sometimes yield robot hand joint configurations that place the hand such that fingers collide with each other or with the palm. It is difficult to add a term to the energy function that penalizes such configurations as self-collision-ness is not a differentiable function of the robot's joint angles.

To address this, we first train a different neural network, a classifier that takes in an Allegro joint angle vector, and tries to classify whether or not the joint configuration yields a self-collision. This classifier is an MLP, and we generate its training data programmatically by repeatedly sampling a joint angle vector within the legal joint limits, and querying a (non-differentiable) self-collision checker to generate a ground-truth binary self-collision label.

Once our self-collision classifier is trained, we use it as a "discriminator" to train our retargeter network. At every training iteration, we pass the predicted robot joint angle vectors to the self-collision classifier. Intuitively, we want the predicted self-collision score to be as low as possible, and therefore we use it as a term in the loss function for the retargeter network. *The gradient of the self-collision score from the collision network is backpropagated through the self-collision classifier, and used to update the weights of the retargeter network*, as shown in Figure 7. This leads the retargeter network to avoid outputting Allegro joint angle configurations that the self-collision classifier believes to be illegal. Our retargeter network and self-collision classifier are akin to the generator and discriminator, respectively, in a Generative Adversarial Network (GAN), though in our case, we pretrain and freeze the self-collision classifier so we don't suffer the instability of jointly optimizing a discriminator and a generator, notorious in GAN training.

3.2. 3rd person video: Wrist pose

A hand that can flex its fingers but does not have the mobility of an arm will not be able to solve many useful tasks. The second branch of our retargeting pipeline therefore focuses on computing the correct pose for the robot arm from human images. In 3rd person video, we are able to see the human torso and can use that as an anchor point, which is useful in retargeting compared to in 1st person video where this information is not available.

Because we aim to build a system that operates from a single color camera, there are two main problems that arise. (1) Without a depth sensor or camera intrinsics, we cannot accurately estimate how far from the camera the human's wrist is. (2) Without camera extrinsics, there is no known transformation between the camera, robot, and human. (If there were, we could simply mount a camera with a fixed known transformation relative to the robot's base frame, localize the position and orientation of the operator's wrist in the 3D camera coordinate frame, and use the known camera transformation extrinsics to determine how the robot's wrist should be positioned and oriented).

To circumvent these issues, at each timestep, we *estimate the relative transformation between the human wrist, and an anchor point on the human's body*, and use this to our advantage. We define the human's torso as the origin of an anchor coordinate frame and choose a suitable point to serve as the "robot's torso." (We manually chose a point 20 cm directly above the robot's base). We posit that the relative transformation between the human's right hand wrist and torso should be the same as the relative transformation between the robot's wrist link and the robot's torso. If the operator's hand were, say, 10 cm in front of their torso, the robot's hand should be 10 cm in front of its "torso." If the operator rotated their wrist so that the palm faced upward, then the robot should rotate its wrist to make its palm face upward.

Concretely, at each timestep, upon capturing an image of the operator, we first compute a crop of the operator's body using a bounding box detector derived from OpenPose Cao et al. (2019), then pass the crop to the body pose estimator from FrankMocap Rong et al. (2021). We model the human body using the parametric SMPL-X model, and the body pose estimator predicts the 3D positions of the joints on the human kinematic chain. By traversing the kinematic chain from the torso joint to the right hand wrist joint, we compute the relative position and orientation between the human's right hand wrist and torso (see Figure 3). The bottom row in Figure 3 depicts this pipeline visually. This simple correspondence trick works surprisingly well in practice and provides a natural user experience for the operator.

We then use an IK solver Buss and Kim (2005b) to compute arm joint angles that place the robot's end-effector at the correct relative transformation, relative to the "robot torso" coordinate frame. To handle minor errors in human body pose estimation and ensure smooth motion, we reject outliers, and apply a low-pass filter on the stream of estimated wrist poses. This low-pass filter is an Exponentially Moving Average (EMA) of end-effector poses. This helps ensure smooth motion in the presence of noise in the pose estimation and retargeting modules. The following update rule is used to update running average P_{EMA} to incorporate the new target pose P_{new}

$$P_{\text{EMA}} = \alpha \cdot P_{\text{new}} + (1 - \alpha) \cdot P_{\text{EMA}} \tag{3}$$

We find $\alpha = 0.25$ works well to create smooth trajectories. We note that a lower value of α can introduce lag, but we find that because our system runs at such a high frequency, this is not an issue in practice. We finally send the smoothed target joint angles to the xArm controller. See the bottom row of Figure 9 for a depiction of our control stack.

3.3. Egocentric human videos: Wrist pose

While using the human's torso as the origin of an anchor coordinate frame is a useful trick, it does not work when the torso is not visible such as in 1st person egocentric video. To get around this problem, we instead must track the wrist moving around in the camera and transform it to M_{Robot}^{Wrist}, the pose of the wrist in a fixed, upright robot frame.

To find the transformation we are after, we break it into a few parts that are composed together. First, to calculate $M_{C_t}^{Wrist}$, where C_t is the camera frame at timestep t, we leverage the Perspective-n-point algorithm (Fischler and Bolles 1981). This takes 2D keypoint outputs (u_i, v_i) by the hand detection model and 3D keypoints from the hand model (x_i, y_i, z_i) and computes $M_{C_t}^{Wrist}$. Specifically, we are given two sets of points, 16 points in 3D on each of the hand model's joints (MCP, PIP, DIP for each of the five fingers plus a wrist) in the model's frame $[X_w, Y_w, Z_w, 1]^t$ and another set of 16 2D points in image frame $[u, v, 1,]^t$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

We use the OpenCV3 solvePnPRANSAC to complete this calculation. This implementation ensures that the process is resilient to erroneous detections. To accurately obtain camera intrinsics for PnP, COLMAP is used (Schönberger et al., 2016).

In human egocentric video datasets, the position of the camera is not always fixed because it is attached to the human's head. Thus, VideoDex deals with moving camera poses as well. Specifically, we compute the transformation between the camera pose in the first frame C_1 and all other frames in the trajectory, C_t. We call this transform $M_{C_1}^{C_t}$. To this end, we run monocular SLAM, specifically ORBSLAM3 (Campos et al., 2021).

Computing poses in the first frame of the camera is important but this is still not in the robot frame because the robot is always upright but the camera is not. It is possible to use camera data (accelerometer) to find this desired transform which we call $M_{World}^{C_1}$. Many off-the-shelf cameras with image stabilization save their acceleration data in the metadata and this can be used to compute a transformation between C_1 and the upright world frame, $M_{World}^{C_1}$. For example, camera's acceleration data will return (0, 0, 9.8) m/s when upright. We initially used the camera data to obtain the

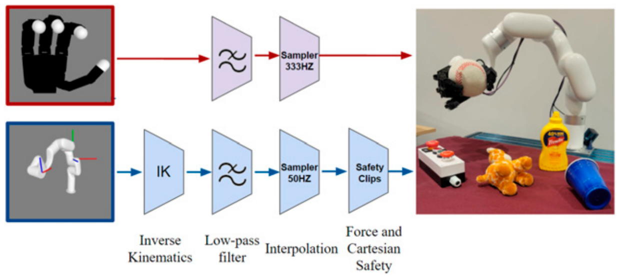

Figure 9. Control Pipeline. Teleoperation from human video requires a control pipeline that takes poses from the visual teleoperation pipeline and controls the robot. Inverse kinematics, low-pass filtering, sampling, and safety clipping are performed. The output controls the robot.

vector α of pitch and roll that parameterizes $M_{World}^{C_1}$, using the following equations

$$\text{pitch} = \tan^{-1}\left(x_{Acc} \Big/ \sqrt{y_{Acc}^2 + z_{Acc}^2}\right) \tag{4}$$

$$\text{roll} = \tan^{-1}\left(y_{Acc} \Big/ \sqrt{x_{Acc}^2 + z_{Acc}^2}\right) \tag{5}$$

However, using sensor data is not a general solution, applicable to arbitrary human egocentric videos. Thus, we develop an approach **that does not rely on camera metadata**. We recover object segmentations for surfaces that are parallel to the floor such as tables, floors, counters, and similar synonyms using a state-of-the-art object detector Detic Zhou et al. (2022). We then compute an estimated depth map from RGB frames only using Adabins Bhat et al. (2021). This does not rely on the long-term contiguity of a video like most SLAM approaches. We use depth map portions that correspond to the relevant objects and calculate a surface normal vector. We estimate α using this normal vector and the previously aforementioned equations. We also remove the dependency on gyroscope data in SLAM by assuming that the scaling factor is 1.0. It is important to note that this wrist retargeting approach **uses only 2D images, which can be obtained from human videos**. We provided detailed ablations on the parameterization of α in Section 5.2.

The robot frame has significant workspace limits that the human does not have. Even if the human arm is smaller than the robot's, the human can walk around, whereas the robot arm cannot move from the middle of the table. Because of this, we must scale the size of the trajectory and center the human starting pose on the robot so that the trajectory is physically possible on the robot. We heuristically compute T_{Robot}^{World} which rescales and rotates the human trajectory in the world frame τ_W^{wrist} into the robot trajectory τ_R^{wrist}.

We rescale each dimension of the arm trajectory as

$$T_{Robot}^{World} = M_{World}^{Wrist_N} + \text{robotWorkspaceCenter} - \frac{\max\left(M_{World}^{Wrist_{1..N}}\right) + \min\left(M_{World}^{Wrist_{1..N}}\right)}{2} \tag{6}$$

We would like to generate more similar trajectories to use in possible data augmentation. The naive method is to add Gaussian noise to the trajectory. While this can be valid, it adds noise to an already noisy system. Instead, we leverage the coordinate frames to create more accurate trajectories. We randomize the workspace scaling that is used by 10%. Additionally, we create a rotation M_{World}^{World} that rotates the initial world frame by up to 10° in each fixed axis in roll pitch yaw convention. This perturbs the direction that the robot moves in its frame.

We interpolate the length of the trajectories using RBF basis functions. All trajectories from the internet data are rescaled to 200 datapoints. This uniformity enables efficient batch training and was used for all of the results.

The final function to obtain M_{Robot}^{Wrist} can be described as the composition of these four transformations as summarized in Table 1

$$M_{Robot}^{Wrist} = T_{Robot}^{World} \cdot M_{World}^{C_1} \cdot M_{C_1}^{C_t} \cdot M_{C_t}^{wrist} \tag{7}$$

Algorithm 1 Procedure for VideoDex

```
Require: Human videos $V_{1:K}^H$ (length $T$), policy $\pi_\theta$, demonstrations $\mathcal{D}_{1:N}$. Human detection $f_{\text{human}}$ (Rong et al. 2021).
  for $k = 1...K$ do
    for $t = 1...T$ do
      Pose parameters $\theta_t, \beta_t = f_{\text{human}}(I_t)$
      Get wrist pose $w_t$ from 4, 5 and 7,
      Hand pose $h_t = H(\theta_t, \beta_t)$
    end for
    Store all $h_t, w_t$ into robot trajectory $\tau_R^k$
    $\hat{\tau}_R^k = \pi_\theta(I_1^k, h_1^k, w_1^k)$
    Optimize $\mathcal{L}_\theta = ||\tau_R^k - \hat{\tau}_R^k||_1$
  end for
  Store policy weights $\theta_h$ to initialize $\pi_\theta$
  while not converged do
    for $n = 1...N$ do
      $\tau_n, I_{1:T}^n = \mathcal{D}_n$
      $\hat{\tau}_n = \pi_\theta(I_1^n, h_1^n, w_1^n)$
      Optimize $\mathcal{L}_\theta = ||\tau_n - \hat{\tau}_n||_1$
    end for
  end while
```

4. Experiment descriptions

We study two applications of our novel retargeting scheme presented in the previous section. First, how do we control the robot hand arm system without relying on any bulky motion capture or multi-camera rigs for accurate 3D estimation. We use the retargeting in a system called *Robotic Telekinesis*, as it provides a human the ability to control a dexterous robot from a distance without any physical interaction. Second, how do we generate robot data from internet videos. Here we use the retargeting in a system called *VideoDex* as it enables us to learn dexterous tasks from internet videos.

Because of the retargeting scheme from Section 3, our system is low-cost, glove-free, and marker-free, and it requires only a single uncalibrated color camera with

Table 1. Transformations required to calculate wrist in robot frame from passive videos to use in learning. M denotes a transformation matrix, where T is a general transformation.

Transforms	Description	Method
$M_{C_t}^{Wrist}$	Wrist in each camera	FrankMocap + PnP
$M_{C_1}^{C_t}$	Track moving camera	IMU/ORBSLAM
$M_{World}^{C_1}$	Make camera parallel to ground	IMU/Stabilization sensor
T_{Robot}^{World}	Rescale and reorient for robot	Heuristic
M_{Robot}^{Wrist}	$T_{Robot}^{World} \cdot M_{World}^{C_1} \cdot M_{C_1}^{C_t} \cdot M_{C_t}^{wrist}$	

which to view the operator from a 3rd person view. It allows any operator to control a four-finger 16-DoF Allegro hand, mounted on a robotic arm, simply by moving their own hand and arm, as illustrated in Figure 1. This system, because it is trained on internet videos, can be used in real-time to control the robot arm and hand system. The operator can even control the robot remotely. For example, Figure 2 illustrates an operator solving a grasping task while monitoring the robot through a video conference feed.

We demonstrate the usability and versatility of our system on 10 challenging dexterous manipulation tasks. We further demonstrate the generality and robustness of our system by performing a systematic study across ten previously untrained human operators.

4.1. Learning priors from 1st person human video

In this experiment, we would like to learn general-purpose manipulation by utilizing large scale human hand action data from first person egocentric video available on the web as seen in Figure 10. We would like to utilize these videos as pseudo-robot experience in the robot embodiment. To do this, we use the *re-targeting* system from Section 3 to re-target the human data to the robot's point of view.

By pretraining policies with this pseudo-robot data, we learn *action* priors of the human hand encoded in the network as an action representation. This differs from many other representations such as Nair et al. (2022) that only train visual priors of the scene. However, it's notoriously difficult to leverage these noisy human video detections. Therefore, we must also employ a policy with *physical* priors to learn smooth and robust policies that do not overfit to noise. By putting this together, we leverage important aspects of the human hand's motion, intent, and interaction. Another difficulty is that human arms and hands and robot arms and hands have a very different shape and embodiment.

Specifically, an open-loop policy π learns first from the retargeted human trajectories (the action prior) and then from real robot trajectories collected in teleoperation. Naively, training a neural network policy on τ_R will lead to overfitting to noisy hand detections. To circumvent this, we first use visual priors from the visual ResNet-based He et al. (2015) encoder provided by Nair et al. (2022), E_ϕ. Then, we introduce *physical priors* to the network backbone, the physically inspired NDPs (Bahl et al., 2020, 2021).

We construct π with the following setup. We firstly process the first scene image I with the visual encoder E_ϕ. Then the extracted features $E_\phi(I)$ are used to cfondition an NDP for the wrist and hand separately, f_{wrist} and f_{hand}. Concretely, each NDP operates by processing the input features with a small MLP which outputs w, g that are the trajectory shape and goal parameters. The forward integrator of the NDP outputs an open-loop trajectory for the hand and the wrist, $\hat{\tau}_R$. We use the following loss function

$$\mathcal{L} = \sum_k \text{Loss}_{L1}\left(\tau_R - \left[f_{\text{hand}}(E_\phi(I_k)), f_{\text{wrist}}(E_\phi(I_k))\right]\right)$$

4.1.1. *Training methodology for priors.*

We collect between 500 and 3000 video clips of humans completing the same task as the robot will from the Epic Kitchens dataset (Damen et al., 2018). For example, in pick, there are close to 3000 video clips of humans picking items. These are retargeted to the robot domain and used to pretrain the network with the human action prior of the pick task. Then, the final policy π is trained on a few teleoperated demonstrations of pick on the real robot. The full training takes about 10 h on a single 2080Ti GPU. More training details can be found in the appendix and in Algorithm 1. Our network consists of the R3M (Nair et al., 2022) initialized ResNet-18 (He et al., 2015). We process these features with a 3 layer MLP with a hidden layer size of 512, which are then processed by 2 NDP (Bahl et al., 2020) networks.

4.1.2. *Task setup for Videodex.*

We pretrain action priors on retargeted Epic Kitchens data for seven robot tasks. Then, we collect about 120–175 demonstrations for each of these tasks on our setup to train the policy. (see Table 2) In pick, the goal is to pick up an object. In rotate, the agent grasps and rotates the object in place. In cover and uncover, the goal is to cover or uncover a pan/plate with a soft cloth object. Push involves flicking/poking an object with the fingers. In place, the robot has to pick up an object and place it into a plate, pan, or pot. In open, we open three different

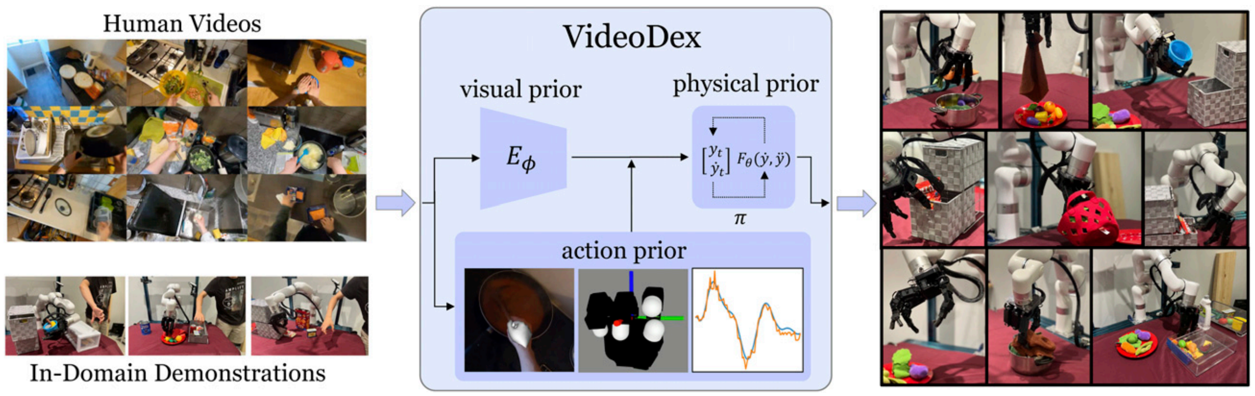

Figure 10. In Videodex, we retarget human videos as an action prior, use pretrainined embeddings as a visual prior, and use Neural Dynamical Policies (NDPs) Bahl et al. (2020) as a physical prior to complete many different tasks on a robotic hand.

drawers. Our testing procedure consists of unseen locations and objects. Details on the tasks and objects are in the supplemental.

While robot hands can provide great dexterity, we also investigate whether 2-finger grippers can benefit from action priors. The internet data is converted to where the closed human hand is a closed 2-finger gripper, and the open human hand is an open 2-finger gripper. We collect separate demonstrations on the real robot using the 2-finger gripper from xArm UFactory. Separate action priors are trained for the 16 DoF LEAP Hand (Shaw et al., 2023a) and the 2-finger gripper.

4.2. Hardware setup

Our hardware setup consists of a LEAP 16 DOF Hand and an XArm 6 manipulator (from Ufactory). The hand is mounted on the wrist of the xArm. We use Intel Realsense D415 cameras to collect human teleoperation and robot videos. We use four NVIDIA RTX 2080TI's for training the policy and two Nvidia RTX 3080s for the teleoperated system.

In our experiments, we tried using the Allegro Hand Lee et al. (2017) and LEAP Hand Shaw et al. (2023a). We found that the Allegro had higher inaccuracy in control and more hardware failures as compared to LEAP Hand. LEAP Hand outperformed the Allegro Hand $7 - 12\%$ on average in all experiments, thus, we use it for our setup Shaw et al. (2023a).

5. Results

We show results from the two applications of this retargeting methodology. First, we teleoperate the robot arm and hand system using a single monocular camera in Robotic Telekinesis. Second, we retarget videos from the internet videos to use as pseudo-robot experience for policy pretraining in Videodex.

Table 2. Left: Number of trajectories we used for each task. Robot data is collected locally using teleportation. Most of these trajectories are 5–15 s in length and capture the motion trajectory of the task and visual data. Right: The number of different objects we used for each task's data collection. In our testing, we show generalization outside of this set of objects.

Task	Robot demos	Objects
Pick	125	8
Rotate	140	8
Open	120	4
Cover	124	12
Uncover	145	12
Place	175	10
Push	136	14

5.1. Robotic telekinesis: Teleoperation from human video

We evaluate the strengths and limitations of our system through experiments on a diverse suite of dexterous manipulation tasks with an expert operator. We also demonstrate the usability and robustness of the system through a smaller set of tasks on a group of ten previously untrained operators. Videos can be found at https://robotic-telekinesis.github.io/.

Baseline Our hand retargeter neural network is compared to an *online optimization* procedure that runs online gradient descent to minimize the energy function between the human and robot hand. We call this baseline DexPilot-Monocular*: the use of online optimization for retargeting is modeled after DexPilot Handa et al. (2020), but the overall system (including the single-camera setup) is held constant between the baseline and our method. At each timestep, given an estimated human hand pose x, a solver iteratively searches for the robot pose y^* that minimizes the energy (cost) function $\mathcal{L}$ with respect to x, that is

$$y^* = \underset{y}{\operatorname{argmin}} \ \mathcal{L}(x, y) \tag{8}$$

The code for DexPilot's Handa et al. (2020) kinematic retargeting module is not available, so we implement their online optimization solver using the Jax GPU-accelerated auto-differentiation engine Bradbury et al. (2018).

We do not compare our system to the full DexPilot system. DexPilot is designed for use in a specific multi-camera rig, but our system is designed to run anywhere. The DexPilot-Monocular* baseline is meant to enable analysis of the tradeoffs between online optimization and neural networks for kinematic retargeting, within a single-camera setup. It uses the retargeting module from DexPilot Handa et al. (2020), but is otherwise identical to our system.

5.1.1. Success rate: Trained operator study. A trained operator attempted a diverse set of tasks to test the capabilities of our system and the DexPilot-Monocular* baseline. These tasks are shown in Figure 11. They span a diverse spectrum of arm and hand motions and involved interacting with a variety of different objects. Each of the ten tasks was run for ten trials with a timeout period of 1 minute. This rigorously tested the system's capabilities and limitations. These tasks are described in Table 3. The operator achieved good success on all tasks—our system outperformed the baseline on 7 out of 10 tasks and performed similarly on the other 3 tasks. Grasping plush objects proved easy as these grasps do not require much precision, but we observed that fine-grained grasps of smaller, more slippery objects like plastic cups occasionally proved difficult. See videos of the trained operator completing these tasks at: https://robotic-telekinesis.github.io/.

During experiments, the expert found that our system was easier to use and performed better than DexPilot-Monocular*. The online gradient descent solver in the baseline occasionally stayed stuck in local minima because it would use the previous pose as a seed. This meant that the hand would often output unnatural poses with the fingers digging into the palm, an issue that the authors of DexPilot also noted. Our method, because it was trained on YouTube data, learned to always output natural hand poses which was useful for operators to use. Since it is not seeded, our method did not get stuck in minima. This data also masked the ambiguities and errors from our single camera-constrained setup. Our method also produced occasional errors on uncommon hand poses unseen in the training set, but these one-off errors did not propagate forward through time. Additionally, the baseline ran at a slower rate and felt delayed to the operator's movements as benchmarked in Table 4. This was jarring and hard to compensate for when trying to complete dexterous tasks. Our system maintained fluidity and felt very responsive when opening and closing the hand.

5.1.2. Usability: Human-subject study. To test usability and generality, we conducted a human-subject study in which 10 previously untrained operators each completed a set of 3 tasks, 7 times each. The first task was a plush dice pickup task (30 s timeout), the second was drawer opening (30 s timeout), and last was to place a cup onto a plate (60 s timeout). The total time for one human subject to learn about the system and complete all tasks took approximately 15 min. Figure 12 reports the completion times of each operator on each of the three tasks.

Although the underlying technology is complex, the user interface was easy to understand and use for all operators. Each operator differed in their style of motion, stances, and appearances, but there were no noticeable discrepancies in the behavior of the system or the distribution of results.

We found that subjects often struggled during the first few trials. However, all subjects found it easy to adjust and learn how to use the system very quickly. Our system was

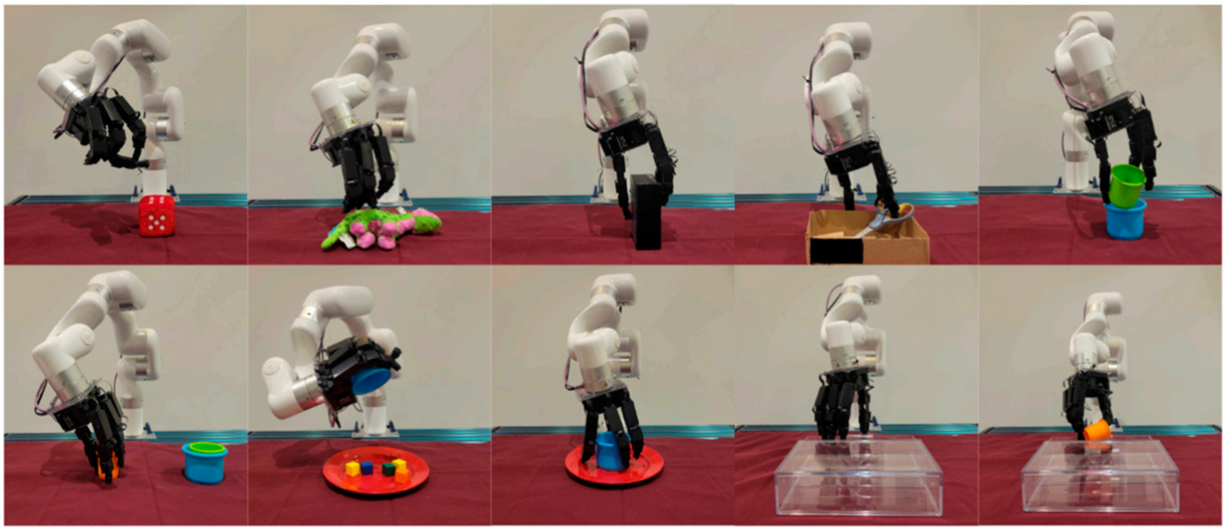

Figure 11. Ten different teleoperation tasks. Top row, left to right: Pickup Dice Toy, Pickup Dinosaur Doll, Box Rotation, Scissor Pickup, Cup Stack. Bottom row, left to right: two cup stacking, pouring cubes onto plate, cup into plate, open drawer and open drawer and pickup cup. Please see videos at https://robotic-telekinesis.github.io/.

Table 3. Success rate and completion time (Mean and standard deviation) of a trained operator completing a variety of tasks using two different methods. The DexPilot-Monocular* baseline is nearly identical to our system, but uses online gradient descent for hand pose retargeting (Inspired by DexPilot Handa et al. (2020)). Our system, which uses a neural network retargeter, outperforms the baseline in 7 out of 10 tasks.

	Success (rate)		Completion time (sec)		
Task	Ours	DexPilot-Mono*	Ours	DexPilot-Mono*	Description
Pickup dice toy	**0.9**	0.7	**8.6 (2.65)**	13.5 (5.47)	Pickup plush dice from table.
Pickup dinosaur doll	**0.9**	0.6	**8.2 (3.49)**	11.00 (3.95)	Pickup plush dinosaur from table.
Box rotation	**0.6**	0.3	37.2 (12.6)	**16.33 (10.69)**	Rotate box 90° onto the smaller side.
Scissor pickup	**0.7**	0.5	28.6 (9.4)	**27.66 (11.09)**	Remove scissors from the box using fingers.
Cup stack	0.6	**0.7**	**21.5 (7.6)**	22.85 (16.57)	Smaller cup must be placed inside the large cup.
Two cup stacking	**0.3**	0.1	**27.3 (11.0)**	45.00 (0.0)	Small cup placed into medium cup into large cup.
Pouring cubes onto plate	**0.7**	0.5	36.80 (17.7)	**13.8 (4.02)**	Five cubes in a cup must be poured onto a plate.
Cup into plate	**0.8**	0.7	**10.6 (4.4)**	13.71 (5.44)	Place cup on the plate.
Open drawer	**0.9**	**0.9**	23.6 (12.3)	**14.88 (4.40)**	Open clear drawer.
Open drawer and pickup cup	0.6	**0.7**	33.7 (8.1)	**28.14 (11.48)**	Open clear drawer and pickup cup inside.

often complimented on its responsiveness and fluidity: subjects did not notice any lag or jitter in the robot's imitation. Subjects enjoyed participating in the study, and some said that teleoperation of the robot was similar to a video game. Additionally, subjects noted they felt safe and comfortable during teleoperation.

The largest challenges with the system were periodic errors in the retargeting of the human fingers to the Allegro robot hand. Many subjects noted instances when they were attempting complicated hand poses, but our system failed to accurately imitate them. In particular, we noticed systematic errors of our system in handling the flexion of the thumb. The shape and joint axes of the Allegro hand thumb are particularly different from that of the human thumb, and we suspect that our energy function does not place enough weight on accurate thumb retargeting. Some subjects observed that the system was worse at tracking their hand when it was all the way open with their palm parallel to the camera, this is a particular issue that we cannot get around with a single camera setup.

5.2. VideoDex: Learning priors from 1st person human video

We perform thorough real world experiments on manipulation tasks, specifically many tasks that require dexterity as in Figure 13. See our webpage for result videos. We aim to answer the following questions. (1) Is VideoDex able to perform general-purpose open-loop manipulation? (2) How much does the action prior of VideoDex help? (3) How much does the physical prior of the NDPs in VideoDex help? (4) What important design choices are there (visual priors, physical priors, or training setup)?

First, we evaluate the need for initialization with the action priors obtained from the human internet videos. The baseline without internet pretraining is called BC-NDP. It uses the same physical prior and visual network initialization, without the initialization from θ_h. We also compare the effect of the action prior on 2-finger gripper policies. Second, we compare against two standard open-loop behavior cloning approaches introduced in recent benchmarks (Dasari et al., 2021). BC-open uses a 2 layer MLP instead of the NDP network. BC-RNN, uses an RNN to pre-process the visual features and then a two-stream, 2 layer MLP for wrist and hand trajectories. We try an offline RL ablation CQL (Kumar et al., 2020), where we use the demonstrations as a sparse reward. We train a behavior cloning policy with the action prior from human videos without the physical prior of the NDP. We call this VideoDex-BC-Open. We ablate the type of visual representation and prior use by trying an initialization using the VGG16 network (Simonyan and Zisserman 2014) (VideoDex-VGG) and the MVP network (He et al., 2022; Xiao et al., 2022) (VideoDex-MVP) based representation trained for robot learning. We ablate the need for a two-stream policy, instead training a single NDP for both hand and wrist. (VideoDex-Single) To see if VideoDex works with fewer

Table 4. Runtime of each stage of our pipeline. Our hand retargeter NN runs at 24 Hz (the online gradient-descent baseline runs at 10 Hz). Both systems use an AMD 3960x CPU and two 3080 Ti GPUs.

Pipeline stage	Ours (Hz)
Open pose body (input from camera)	29
Open pose hand (input from camera)	29
Frank Mocap body	16
Frank Mocap hand	27
Body pose retargeter (output to robot)	16
Hand retargeter (output to robot)	24

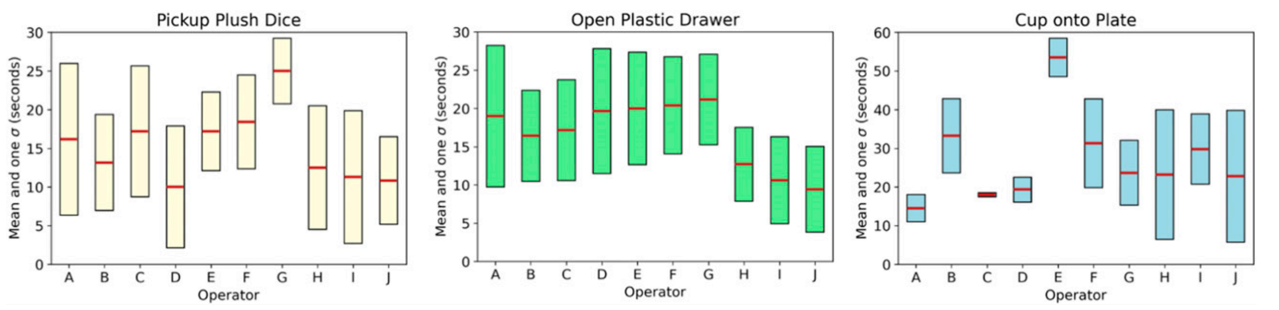

Figure 12. Ten novice operators were asked to complete tasks: (1) picking up a plush dice, (2) opening a plastic drawer, and (3) placing a cup onto a plate. For each task, the mean and standard deviation completion times were computed over seven trials.

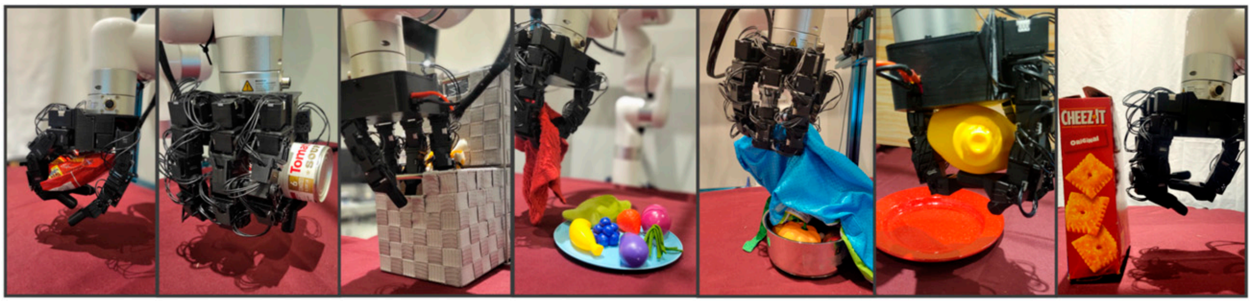

Figure 13. Tasks used in experiments. From left to right: pick, rotate, open, cover, uncover, place and push. See https://video-dex.github.io for videos of these tasks.

demonstrations (around 50 demonstrations, 5–7 per variant only), we train a policy called VideoDex-Constrained.

We analyze the results of our experiments and the guiding questions discussed. We present the results of our findings as a 0–1 success rate in Table 6 and the result of the ablations we ran on the place task in Table 5.

Table 5. We present the results of the ablations discussed in section 5.2. These are all performed on the place task.

	Train	Test
Baselines:		
BC-NDP (Bahl et al., 2021)	0.70	0.35
BC-Open (Dasari et al., 2021)	0.40	0.25
BC-RNN (Dasari et al., 2021)	0.70	0.50
CQL (Kumar et al., 2020)	0.40	0.20
No Physical Prior:		
VideoDex-BC-open	0.50	0.50
VideoDex-single	0.50	0.30
Visual Prior Ablation:		
VideoDex-VGG	0.20	0.20
VideoDex-MVP	0.40	0.20
Constrained Data:		
VideoDex-const-5	0.80	0.60
VideoDex-const-10	0.50	0.30
VideoDex (ours)	**0.90**	**0.70**

5.2.1. Effect of action priors. We firstly compare VideoDex against methods that do not employ an action prior trained on human data, as explained in Section 4.1. For almost all of the tasks, VideoDex either outperforms baselines or has a similar performance, especially for held-out objects/instances. We believe that one of the key aspects of VideoDex generalizing to test objects is the action prior pretraining on human videos. This can be seen in Figure 14. Without ever training on the robot demonstrations, the trajectories initialized using the action prior pretrained network θ_h (left) are much closer to the ground-truth trajectories of a network that is initialized using only a visual prior such as the encoder from Nair et al. (2022) (right). From the results, we see that VideoDex-BC-Open with action priors (Table 5) outperforms BC-Open. Having a physical prior added (BC-NDP) tends to help, but it is not the case for every task. We suspect that some tasks require smoother behavior than others. Additionally, in Table 5 our offline RL baseline, CQL (Kumar et al., 2020) does not perform as well as the rest of the approaches, even underperforming the Behavior Cloning setup. Qualitatively, we see a much less smooth and less safe execution with this method, thus, we only perform it on one task (place). Note that we use the same visual prior for this as well.

5.2.2. Effect of visual priors. We compared using our approach with MVP (VideoDex-MVP) (Xiao et al., 2022) and VGG (VideoDex-VGG) (Simonyan and Zisserman 2014) and their performance was below VideoDex using Nair et al. (2022). This is likely because both encoders are much larger than the ResNet18 (He et al., 2015) we use and require a lot more training time than feasible on human videos. However, VideoDex-MVP still performs better than VideoDex-VGG, which indicates that using a visual prior trained on human data does in fact help, as Xiao et al. (2022) trained the representation in self-supervised fashion on videos and use the embeddings to perform robotics tasks in simulation. We see in Table 6, that while visual priors are important, action priors are more impactful.

5.2.3. Effect of physical priors and architectural choices. We compare different types of physical priors in Table 6 and in Table 5. In general, (BC-NDP) tends to outperform baselines without a physical prior, except for BC-RNN in a couple of tasks. BC-RNN performs less aggressive behavior, which allowed it to efficiently grasp more objects. In Table 5, it's shown that an important physical prior is to treat the wrist and the hand in a more disentangled manner, as the performance for VideoDex-Single tends to drop compared to BC-NDP and VideoDex-BC-Open (Behavior Cloning with our action prior pre-training). The two-stream architecture aids in learning, as it allows the policy to disentangle the actions of the wrist and the hand. This is important as the same grasp might be used for picking objects in many different locations, and similarly, it is possible to localize many objects and perform completely different types of interactions.

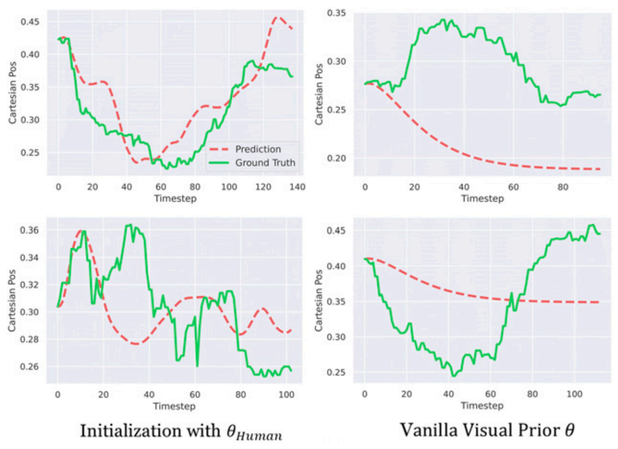

Figure 14. Networks initialized using action priors on human data without further training are closer to ground-truth robot trajectories than networks only initialized using visual priors.

5.2.4. Comparing effects of actions, visual, and physical priors. Firstly, we ran an ablation where we pertained a policy on human videos performing the place task and finetune it on the uncover task (using robot data). Similarly, we pretrained a policy on Uncover and finetuned on Place. The results are in the below table under VideoDex-Transfer. We see that for both tasks, the performance degrades slightly, especially in the place task. We also train by adding noise to the demonstration trajectories, by adding two different levels of Gaussian noise with standard deviation being 0.01 and 0.05, shown as VideoDex-Noise-0.01 and VideoDex-Noise-0.05. We find that adding more noise definitely hurts the performance of the method. We also train ResNet18 (He et al., 2015) features initialized from ImageNet (Deng et al., 2009) training instead of the R3M (Nair et al., 2022) features, and the results in VideoDex-ImageNet. We can see that performance drops off, which indicates that visual priors are important. Note that all of the reported numbers are on test objects. We present the results in Table 7.

5.2.5. Generalization with less data. We limit VideoDex to a maximum of 5 and 10 teleoperated demonstrations per variant (we have 12–15 variants in our setup). As shown in Table 6, even with 5 instances per variant, we still see a 30% success rate for unseen objects. Empirically, the policies generally go to the right area but are not able to grasp objects properly. With less robot experience, VideoDex outperforms which demonstrates that action priors also boost sample efficiency.

5.2.6. Hand versus 2-finger gripper. We compare whether the action priors from VideoDex also help in the more general 1-DOF gripper setting. In Table 8, we find that in the 1-DOF setting, VideoDex still improves performance on these tasks. This is because the priors from human internet videos still encode typical wrist trajectory behaviors as well as when the gripper should close for each task.

6. Hand retargeting analysis

In this section, we analyze the finger retargeting method introduced in Section 3 that goes from human hand to robot hand. We analyze its accuracy and the effect of the self-

Table 6. We present the results of train objects and test objects for Videodex and baselines for each task as described.

	Pick		Rotate		Open		Cover		Uncover		Place		Push	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
BC-NDP (Bahl et al., 2021)	0.64	0.38	**0.94**	0.56	**0.90**	0.60	**0.78**	0.58	0.88	0.82	0.70	0.35	1.00	0.71
BC-Open (Dasari et al., 2021)	0.50	0.44	0.72	0.38	0.80	0.40	0.44	0.58	**1.00**	**0.91**	0.40	0.25	1.00	0.93
BC-RNN (Dasari et al., 2021)	0.56	0.31	0.78	0.50	**0.90**	0.50	0.56	0.42	0.88	0.75	0.70	0.50	1.00	**1.00**
VideoDex	**0.83**	**0.77**	0.85	**0.71**	**0.80**	**0.80**	**0.75**	**0.63**	**0.96**	0.92	**0.89**	**0.80**	1.00	**1.00**

Table 7. Ablations that compare effects of different action, visual and physical priors, as well as seeing how pretraining on different data transfers to other tasks.

Method/Task	Place	Uncover
VideoDex-Noise-0.01	0.55	0.87
VideoDex-Noise-0.05	0.50	0.60
VideoDex-ImageNet	0.40	0.62
VideoDex-Transfer	0.60	0.87
VideoDex-Original	0.70	0.90

Table 8. We compare how the 1-DOF xArm gripper performs using Videodex. UFactory separate demonstrations were collected using this gripper. While Videodex improves the performance of the final 1-DOF system, the improvment here is less than that from the 16DOF allegro hand.

	Place	Open	Pick
1-DOF Gripper Results:			
1-DOF BC-Open	0.62	0.69	0.71
1-DOF VideoDex	**0.69**	**0.82**	**0.77**
Improvement of Videodex from BC-Open:			
1-DOF improvement	**0.07**	**0.13**	**0.06**
Hand improvement	**0.55**	**0.4**	**0.33**

collision classifier for hand retargeting. We also analyze the initial pose orientation computation for the 3rd person video retargeting of the wrist.

6.1. Accuracy of retargeter network

We compare the accuracy of DexPilot-Monocular*'s *online optimization* with our neural network retargeter that relies on *offline optimization* during training. We gather a test set of 500 sequences from the DexYCB video dataset Chao et al. (2021), which contains videos with annotated ground-truth human hand poses. For each video, at each time step, the poses are fed to both our neural network and DexPilot-Monocular* with a (generous) time budget of 40 ms to solve. We emphasize that both retargeters optimize the same energy function but in different ways.

We do not, however, have access to "ground-truth" Allegro joint angles against which to compare the output of the two retargeters. To circumvent this, we design a version of DexPilot-Monocular* that is allowed an *infinite time budget* to run until convergence. We call this the pseudo-ground-truth oracle, and our assumption is that its final output is as close to optimal as possible.

We compare the root mean squared error (RMSE) between the oracle's outputs and the outputs of each of our two retargeters on the dataset. Our neural network retargeter outperforms DexPilot-Monocular* in matching the oracle. The neural network retargeter achieves an RMSE of **0.17** radians (about 10°), while DexPilot-Monocular* achieves an RMSE of **0.25** radians per joint (about 14°).

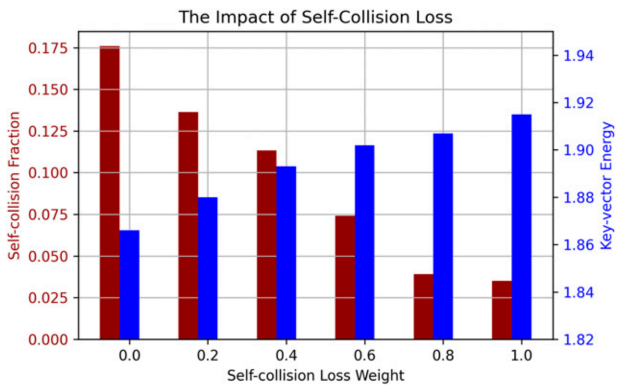

Figure 15. As the weight of the adversarial self-collision loss is increased, the hand retargeter network produces fewer self-colliding joint configurations (maroon), but incurs a higher energy with less similar poses (blue). A higher energy means that the predicted robot hand pose is dissimilar to the operator's hand pose.

6.2. Self-collision avoidance

We perform an ablation on the weight of the self-collision classifier (Section 3.1.3) in the energy function, to see how it affects the behavior of the hand retargeter. We use a test set of 3000 held-out hand poses from the FreiHand dataset (Zimmermann et al., 2019) and consider 6 different hand retargeter networks, trained with collision-loss weights of 0, 0.2, 0.4, 0.6, 0.8, and 1. (A weight of 0.8, e.g., means that the self-collision loss is weighed 0.8 as heavily as the sum of all the other key-vector matching loss terms in the energy function). Each network makes predictions on the data and we compute (1) the fraction of resulting Allegro joint angle vectors that result in self-collision, and (2) the average value of the key-vector energy terms over the dataset.

We summarize the results in Figure 15. The plot shows there is a trade-off between minimizing self-collisions and minimizing key-vector dissimilarity. As we increase the weighting term of the self-collision avoidance loss term in the energy function, we produce fewer offending joint configurations but minimization performance degrades for the other terms in the energy function. We depict this trade-off visually in Figure 16. It is difficult to confidently assert that one is more valuable than the other, and in practice, we find that a middle ground works very effectively for the user.

6.3. Initial pose computation comparison

We compare three different ways to estimate α_p or $M^{C_1}_{World}$, the vector that points parallel to gravity. These methods contrast with VideoDex which uses the surface normal of objects that are typically parallel with the floor to calculate the direction of gravity. VideoDex-Fixed, assumes that α_p is [0,0]. This is reasonable as we are not relying on robots to exactly mimic the human but get a general action prior. VideoDex-Random randomizesα_p in the range of 15–45°, which is the typical egocentric camera angle. VideoDex-IMU uses the internal image stabilization sensor data to estimate the upright vector.

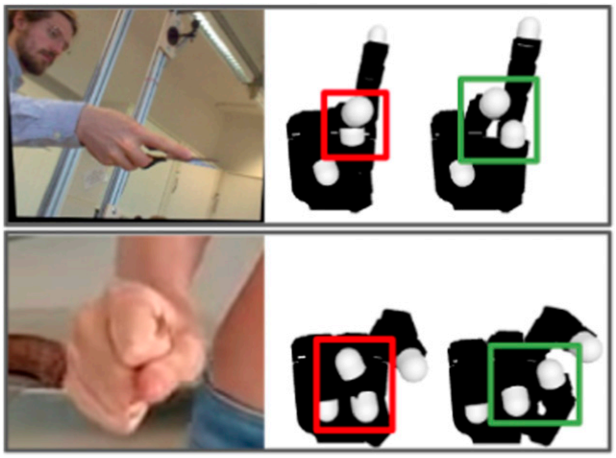

Figure 16. The contribution of an adversarial self-collision loss. The red boxes highlight instances where the vanilla retargeting network outputs Allegro hand poses that result in self-collision. The green boxes depict the predictions of the network trained with self-collision loss. These robot hand poses maintain functional similarity to the human's hand pose, but avoid self-collision.

Table 9. Ablations that compare the different ways of calculating the initial pitch of the camera with respect to gravity, on test objects. This enables us to transform human trajectories to be upright like the robot is.

	Place	Cover	Uncover
Videodex-Fixed	0.55	0.50	0.77
Videodex-Random	0.45	0.63	0.85
Videodex-IMU	0.70	**0.67**	0.90
VideoDex	**0.80**	0.63	**0.92**

None of these approaches use gyroscope data in SLAM, as we assume that the scaling factor is 1.0. In Table 9, we present the results of these experiments. The performance degrades when randomizing or setting $M_{World}^{C_1}$ to a fixed value, in all three of the tasks, but it is still comparable to or better than our baselines that do not use any human action data. A possible explanation for the fact that VideoDex-Surface performed better than our VideoDex-IMU is that the sensor data may be noisy and estimating surface normals from visual features is more robust.

7. Conclusion

We present two systems using our retargeting system from human video to robot actions. First, we introduced *Robotic Telekinesis*, a system for in-the-wild, real-time, remote visual teleoperation of a dexterous robotic hand and arm, in which a human operator demonstrates tasks to the robot just by moving their own hands. We leverage the latest advancements in 3D human pose estimation and thousands of hours of raw day-to-day human footage on the internet to train a system that can understand human motion, and retarget it to corresponding robot actions. Our method requires only a single color camera and can be used out-of-the-box by any operator on any task, without any actively collected robot training data. We show that our system enables experts and novices alike to successfully perform a number of different dexterous manipulation tasks. Second, we introduce Videodex which leverages *visual*, *action*, and *physical* priors from human video datasets to guide robot behavior. These actions and physical priors in the neural network dictate the typical human behavior for a particular robot task. We test our approach on a robot arm and dexterous hand-based system and show strong results on various manipulation tasks, outperforming various state-of-the-art methods. We hope that this retargeting method and learning from human videos will have many possible applications in the future.

Acknowledgements

We are grateful to Ankur Handa for initial feedback and to Murtaza Dalal, Russell Mendonca, Jianren Wang, and Sudeep Dasari for feedback on the project. We would also like to thank Murtaza Dalal, Alex Li, Kathryn Chen, Ankit Ramchandani, Ananye Agarwal, Jianren Wang, Zipeng Fu, Shivam Duggal, and Sam Triest for helping with experiments.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: KS is supported by NSF Graduate Research Fellowship under Grant No. DGE2140739. The work was supported in part by NSF IIS-2024594, ONR N00014-22-1-2096, GoodAI Research Award, and by the Samsung GRO Research Award.

ORCID iDs

Kenneth Shaw https://orcid.org/0009-0002-8571-2922
Aditya Kannan https://orcid.org/0000-0002-3104-7560

References

Agarwal A, Uppal S, Shaw K, et al. (2023) Dexterous functional grasping. In: Conference on robot learning. PMLR, 3453–3467.

Antotsiou D, Garcia-Hernando G and Kim TK (2018) Task-oriented hand motion retargeting for dexterous manipulation imitation. In: Proceedings of the European conference on computer vision (ECCV) workshops. Munich, Germany, 8 September–14 September 2018.

Arunachalam SP, Silwal S, Evans B, et al. (2022) *Dexterous Imitation Made Easy: A Learning-Based Framework for Efficient Dexterous Manipulation*. DOI: 10.48550/ARXIV.2203.13251. https://arxiv.org/abs/2203.13251

Bahl S, Mukadam M, Gupta A, et al. (2020) Neural dynamic policies for end-to-end sensorimotor learning. In: *NeurIPS*.

Bahl S, Gupta A and Pathak D (2021) *Hierarchical Neural Dynamic Policies*. RSS.

Bahl S, Gupta A and Pathak D (2022) *Human-to-robot Imitation in the Wild*. RSS.

Bahl S, Mendonca R, Chen L, et al. (2023) Affordances from human videos as a versatile representation for robotics. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, Vancouver, BC, Canada, 17 June–24 June 2023, 13778–13790.

Bhat SF, Alhashim I and Wonka P (2021) Adabins: depth estimation using adaptive bins. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, Nashville, TN, USA, 20 June–25 June 2021, 4009–4018.

Bojarski M, Del Testa D, Dworakowski D, et al. (2016) *End to End Learning for Self-Driving Cars*. DOI: 10.48550/ARXIV.1604.07316. https://arxiv.org/abs/1604.07316.

Bradbury J, Frostig R, Hawkins P, et al. (2018) *JAX: Composable Transformations of Python+NumPy Programs*. https://github.com/google/jax

Brown TB, Mann B, Ryder N, et al. (2020) *Language Models Are Few-Shot Learners*. arXiv.

Buss SR and Kim JS (2005a) Selectively damped least squares for inverse kinematics. *The Journal of Graphics Tools* 10(3): 37–49.

Buss SR and Kim JS (2005b) Selectively damped least squares for inverse kinematics. *The Journal of Graphics Tools* 10(3): 37–49. DOI: 10.1080/2151237X.2005.10129202

Campos C, Elvira R, Rodríguez JJG, et al. (2021) Orb-slam3: an accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics* 37(6): 1874–1890.

Cao Z, Hidalgo G, Simon T, et al. (2019) Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43(1): 172–186.

Carpentier J, Saurel G, Buondonno G, et al. (2019) The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In: IEEE International Symposium on System Integrations (SII). IEEE.

Chao YW, Yang W, Xiang Y, et al. (2021) DexYCB: a benchmark for capturing hand grasping of objects. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20 June–25 June 2021.

Chen T, Kornblith S, Norouzi M, et al. (2020) A simple framework for contrastive learning of visual representations. In: HD III and A Singh (eds) Proceedings of the 37th International conference on machine learning, proceedings of machine learning research. PMLR, Vol. 119, 1597–1607. URL https://proceedings.mlr.press/v119/chen20j.html

Chen AS, Nair S and Finn C (2021) *Learning Generalizable Robotic Reward Functions from" In-The-Wild" Human Videos*. arXiv preprint arXiv:2103.16817.

Damen D, Doughty H, Farinella GM, et al. (2018) Scaling egocentric vision: the epic-kitchens dataset. In: European conference on computer vision (ECCV). Springer.

Das P, Xu C, Doell RF, et al. (2013) A thousand frames in just a few words: lingual description of videos through latent topics and sparse object stitching. In: Proceedings of the IEEE Conference on computer vision and pattern recognition. IEEE, 2634–2641.

Das N, Bechtle S, Davchev T, et al. (2020) *Model-based Inverse Reinforcement Learning from Visual Demonstrations*. arXiv preprint arXiv:2010.09034.

Dasari S, Wang J, Hong J, et al. (2021) Rb2: robotic manipulation benchmarking with a twist. In: NeurIPS datasets and benchmarks track (Round 2). NIPS.

Deng J, Dong W, Socher R, et al. (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on computer vision and pattern recognition. IEEE, pp. 248–255.

Devlin J, Chang MW, Lee K, et al. (2018) *Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805.

Fang B, Wei X, Sun F, et al. (2019) Skill learning for human-robot interaction using wearable device. *Tsinghua Science and Technology* 24(6): 654–662.

Fischler MA and Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24(6): 381–395. DOI: 10.1145/358669.358692

Goyal R, Ebrahimi Kahou S, Michalski V, et al. (2017) The "something something" video database for learning and evaluating visual common sense. In: Proceedings of the IEEE International conference on computer vision (ICCV). IEEE.

Grauman K, Westbury A, Byrne E, et al. (2022) Ego4d: around the world in 3,000 hours of egocentric video. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition. IEEE, 18995–19012.

Han S, Liu B, Wang R, et al. (2018) Online optical marker-based hand tracking with deep labels. *ACM Transactions on Graphics* 37(4): 1–10.

Handa A, Van Wyk K, Yang W, et al. (2020) Dexpilot: vision-based teleoperation of dexterous robotic hand-arm system. In: 2020 IEEE International conference on robotics and automation (ICRA). IEEE, 9164–9170. DOI: 10.1109/ICRA40945.2020.9197124

He K, Zhang X, Ren S, et al. (2015) *Deep Residual Learning for Image Recognition*. CoRR abs/1512.03385. URL https://arxiv.org/abs/1512.03385

He K, Gkioxari G, Dollar P, et al. (2017) Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). IEEE.

He K, Chen X, Xie S, et al. (2022) Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, 16000–16009.

Heilbron FC, Escorcia V, Ghanem B, et al. (2015) Activitynet: a large-scale video benchmark for human activity understanding. In: CVPR, 961–970.

Hodgins J (n.d) *Cmu Graphics Lab Motion Capture Database*. https://mocap.cs.cmu.edu/

Ionescu C, Papava D, Olaru V, et al. (2013) Human3. 6 m: large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Transactions* *on Pattern Analysis and Machine Intelligence* 36(7): 1325–1339.

Kalashnikov D, Irpan A, Pastor P, et al. (2018) *Qt-opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation*. *arXiv preprint arXiv:1806.10293*.

Kanazawa A, Black MJ, Jacobs DW, et al. (2017) *End-to-End Recovery of Human Shape and Pose*. CoRR abs/171206584. URL https://arxiv.org/abs/1712.06584

Kannan A, Shaw K, Bahl S, et al. (2023) *Deft: Dexterous Fine-Tuning for Real-World Hand Policies*. CoRL.

Kumar V and Todorov E (2015) Mujoco haptix: a virtual reality system for hand manipulation. In: 2015 IEEE-RAS 15th International conference on humanoid robots (Humanoids), Seoul, Korea, 3 November–5 November 2015, 657–663. DOI: 10.1109/HUMANOIDS.2015.7363441

Kumar A, Zhou A, Tucker G, et al. (2020) Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33: 1179–1191.

Lee J and Ryoo MS (2017) Learning robot activities from first-person human videos using convolutional future regression. In: CVPR Workshops, 1–2.

Lee DH, Park JH, Park SW, et al. (2017) Kitech-hand: a highly dexterous and modularized robotic hand. *IEEE/ASME Transactions on Mechatronics* 22(2): 876–887. DOI: 10.1109/TMECH.2016.2634602

Levine S, Finn C, Darrell T, et al. (2016) *End-to-end Training of Deep Visuomotor Policies*. JMLR.

Li S, Ma X, Liang H, et al. (2019) Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network. In: 2019 International Conference on Robotics and Automation (ICRA). IEEE, 416–422.

Loper M, Mahmood N, Romero J, et al. (2015) Smpl: a skinned multi-person linear model. *ACM Transactions on Graphics* 34(6): 1–16.

Ma RR and Dollar AM (2011) On dexterity and dexterous manipulation. In: 2011 15th International Conference on Advanced Robotics. ICAR, Tallinn, Estonia, 20 June–23 June 2011, 1–7. DOI: 10.1109/ICAR.2011.6088576

Makoviychuk V, Wawrzyniak L, Guo Y, et al. (2021) *Isaac Gym: High Performance Gpu-Based Physics Simulation for Robot Learning*. *arXiv preprint arXiv:2108.10470*.

Mandikal P and Grauman K (2022) Dexvip: learning dexterous grasping with human hand pose priors from video. In: Conference on Robot Learning, Atlanta, GA, 6 November–9 November, 2023. PMLR, 651–661.

Mannam P, Shaw K, Bauer D, et al. (2023) Designing anthropomorphic soft hands through interaction. In: 2023 IEEE--RAS 22nd International Conference on Humanoid Robots (Humanoids). IEEE, 1–8. DOI: 10.1109/Humanoids57100.2023.10375195

Mendonca R, Bahl S and Pathak D (2023) *Structured World Models from Human Videos*. *arXiv preprint arXiv:2308.10901*.

Nair AV, Pong V, Dalal M, et al. (2018) Visual reinforcement learning with imagined goals. In: *NeurIPS*. pp: 9191–9200.

Nair S, Rajeswaran A, Kumar V, et al. (2022) *R3m: A Universal Visual Representation for Robot Manipulation*. *arXiv preprint arXiv:2203.12601*.

Pari J, Muhammad N, Arunachalam SP, et al. (2021) *The Surprising Effectiveness of Representation Learning for Visual Imitation*. *arXiv preprint arXiv:2112.01511*.

Pavlakos G, Choutas V, Ghorbani N, et al. (2019) Expressive body capture: 3D hands, face, and body from a single image. In: Proceedings IEEE Conf. On Computer Vision and Pattern Recognition (CVPR). IEEE, 10975–10985.

Peng XB, Coumans E, Zhang T, et al. (2020) *Learning Agile Robotic Locomotion Skills by Imitating Animals*. *arXiv preprint arXiv:2004.00784*.

Pinto L, Gandhi D, Han Y, et al. (2016) *The Curious Robot: Learning Visual Representations via Physical Interactions*. ECCV.

Pomerleau DA (1988) Alvinn: an autonomous land vehicle in a neural network. In: D Touretzky (ed) *Advances in neural information processing systems*. Morgan-Kaufmann, Vol. 1. https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf

Qin Y, Wu YH, Liu S, et al. (2021) *Dexmv: Imitation Learning for Dexterous Manipulation from Human Videos*. *arXiv preprint arXiv:2108.05877*.

Qin Y, Su H and Wang X (2022) *From One Hand to Multiple Hands: Imitation Learning for Dexterous Manipulation from Single-Camera Teleoperation*. DOI: 10.48550/ARXIV.2204.12490. https://arxiv.org/abs/2204.12490

Romero J, Tzionas D and Black MJ (2017) Embodied hands: modeling and capturing hands and bodies together. *ACM Transactions on Graphics* 36(6): 1–17.

Rong Y, Shiratori T and Joo H (2021) Frankmocap: a monocular 3d whole-body pose estimation system via regression and integration. In: Proceedings of the IEEE/CVF International conference on computer vision (ICCV) workshops. IEEE, 1749–1759.

Schmeckpeper K, Rybkin O, Daniilidis K, et al. (2020) *Reinforcement Learning with Videos: Combining Offline Observations with Interaction*. *arXiv preprint arXiv:2011.06507*.

Schönberger JL, Zheng E, Pollefeys M, et al. (2016) Pixelwise view selection for unstructured multi-view stereo. In: European conference on computer vision (ECCV). Springer Science.

Sermanet P, Lynch C, Chebotar Y, et al. (2018) Time-contrastive networks: self-supervised learning from video. In: ICRA, Brisbane, QLD, Australia. IEEE.

Shan D, Geng J, Shu M, et al. (2020) Understanding human hands in contact at internet scale. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, 9869–9878.

Shao L, Migimatsu T, Zhang Q, et al. (2021) Concept2robot: learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research* 40(12-14): 1419–1434.

Sharma P, Pathak D and Gupta A (2019) *Third-Person Visual Imitation Learning via Decoupled Hierarchical Controller*. *arXiv preprint arXiv:1911.09676* 32.

Shaw K, Agarwal A and Pathak D (2023a) Leap Hand: Low-Cost, Efficient, and Anthropomorphic Hand for Robot Learning. RSS.

Shaw K, Bahl S and Pathak D (2023b) Videodex: learning dexterity from internet videos. In: Conference on robot learning. PMLR, 654–665.

Simonyan K and Zisserman A (2014) *Very Deep Convolutional Networks for Large-Scale Image Recognition*. *arXiv preprint arXiv:1409.1556*.

Sivakumar A, Shaw K and Pathak D (2022) *Robotic Telekinesis: Learning a Robotic Hand Imitator by Watching Humans on Youtube*. arXiv.

Smith L, Dhawan N, Zhang M, et al. (2020) *Avid: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos*. RSS.

Todorov E, Erez T and Tassa Y (2012) *MuJoCo: A Physics Engine for Model-Based Control*. IROS.

UFactory (n.d) xarm6 by ufactory. https://www.ufactory.cc/xarm-collaborative-robot

Umeyama S (1991) Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(04): 376–380.

Villegas R, Yang J, Ceylan D, et al. (2018) *Neural Kinematic Networks for Unsupervised Motion Retargetting*. CoRR abs/1804.05653. URL https://arxiv.org/abs/1804.05653

Wang J, Mueller F, Bernard F, et al. (2020) Rgb2hands: real-time tracking of 3d hand interactions from monocular RGB video. *ACM Transactions on Graphics* 39(6): 1–16.

Xiao T, Radosavovic I, Darrell T, et al. (2022) *Masked Visual Pre-training for Motor Control*. *arXiv preprint arXiv:2203.06173*.

Young S, Gandhi D, Tulsiani S, et al. (2020) *Visual Imitation Made Easy*. *arXiv preprint arXiv:2008.04899*.

Zakka K, Zeng A, Florence P, et al. (2021) *Xirl: Cross-Embodiment Inverse Reinforcement Learning*. *arXiv preprint arXiv:2106.03911*.

Zhou X, Girdhar R, Joulin A, et al. (2022) *Detecting Twenty-Thousand Classes Using Image-Level Supervision*. ECCV.

Zimmermann C, Ceylan D, Yang J, et al. (2019) Freihand: a dataset for markerless capture of hand pose and shape from single RGB images. In: Proceedings of the IEEE/CVF International conference on computer vision, Seoul, Korea, 27 October–28 October 2019, 813–822.