SAPIENZA
UNIVERSITÀ DI ROMA

Motion Retargeting for Grasping Tasks

Mapping synergies from humans to robotic hands with dissimilar kinematics

A.A. 2024 - 2025

Based on the work of G. Gioioso, G. Salvietti, M. Malvezzi, and D. Prattichizzo (2013), IEEE Transactions on Robotics[1]

Matteo Zamponi
Luca Colamarino
Giuseppe D'Addario
Federico Tranzocchi
Sapienza University of Rome

Contents

Chapter 1

Introduction

Robotic hands are becoming increasingly common in research and industry. Modern robotic hands often have many degrees of freedom (DoF) to imitate the dexterity of the human hand, which has over 20 DoF (e.g., Figure 1.1 shows a typical example of robotic hand (right)). A high number of DoF provides greater dexterity and the ability to carry out complex manipulation tasks. In fact, as humans, we can perform a huge variety of grasps in a natural way. However, controlling all of these DoF individually is generally not feasible for a robotic hand, as it would require an impractical number of actuators: robotic hand models are generally *underactuated*, meaning they have fewer actuators than DoF.

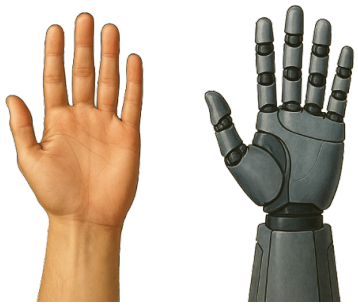


Figure 1.1: Comparison between a human hand (left) and a robotic hand (right).

This concept goes under the name of *postural synergies* [2]: the human brain controls the hand easily using coordinated patterns, the synergies, that reduce the complexity of hand control.

This project addresses the problem of *motion retargeting* from a human hand to a robotic hand, which can also be non-anthropomorphic. The goal is to leverage the patterns described by the postural synergies to control a target robotic hand that has a different kinematic structure from that of the human hand. To do this, we use an object-based mapping approach called the *virtual sphere* method [1], which focuses on the interaction between the hand and the object being manipulated, rather than on the specific structure of the hand itself.

1.1 The correspondence problem

A major challenge in teleoperation is the *correspondence problem*. Human hands and robotic hands are rarely identical: they usually have different bone lengths, different types of joints, and a different number of fingers. Because of these differences, standard mapping techniques like

mapping joint angles directly or mapping fingertip positions fail to accurately translate human motion. To solve this, we adopt an approach proposed by Gioioso et al. [1] that works in the *object domain*. Instead of mapping the anatomy of the hand directly, we map the effect that the hand has on a virtual object (a sphere) held in the hand, as illustrated in Figure 1.2.

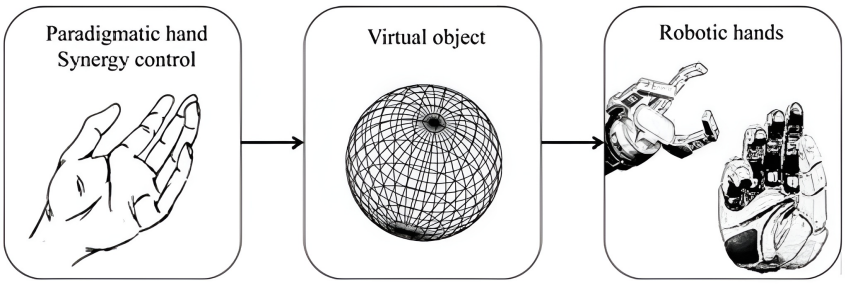


Figure 1.2: Schematic representation of the synergy-driven control of a robotic hand via the virtual sphere method. The human hand controls a virtual sphere, whose motion is then replicated by the robotic hand.

1.2 Synergy-driven input

To control the robotic hand, we first need a reliable reconstruction of the human hand pose. In this project, we use the Weart TouchDIVER G1 haptic glove in Figure 1.3, which provides the closure of three fingers (thumb, index, middle) and the abduction of the thumb, plus other measurements that are not of interest for this work.

Figure 1.3: The Weart TouchDIVER G1 haptic glove. (Image source: https://weart.it/task-operations/).

To reconstruct the full hand pose from this limited input, we rely on a reconstruction system developed in a previous work by Primiceri et al. [3]. Their system uses neural networks to estimate the full human hand pose from the sparse glove data, based on the theory of *postural synergies* [2]. We take this reconstructed hand pose as the input for our motion retargeting system, focusing on the mathematical translation of this motion to the robotic hand.

1.3 Objective

Our main goal is to implement and validate a robust pipeline that allows a human operator to control robotic hands with different kinematics in real-time. Specifically, we present an

algorithm that creates a virtual sphere inside the human hand and maps its deformation and movement to the robotic hand. Thanks to the mapping in the domain of the manipulated object, we create a generalized kinematic solver that can handle different robotic structures (e.g., from a 5-fingered hand to a 3-fingered one, or a 2-fingered gripper) without needing to redesign the mapping for each case.

Finally, we evaluate the performance of our system by analyzing the fidelity of the retargeting process. In particular, we assess how accurately the robotic hand replicates the deformation of the virtual sphere compared to the human reference, measuring the error in terms of sphere radius, position, and rotation. Additionally, we analyze the energy associated with the elastic deformation of the grasp to quantify the similarity between the human and robotic grasping strategies.

1.4 Structure

The rest of this report is structured as follows. In Chapter 2 we review the relevant literature on hand synergies and motion retargeting techniques. In Chapter 3 we provide a mathematical formulation of the virtual sphere method and our implementation details. In Chapter 4 we describe the whole system architecture, including the hardware and software components, and the error metrics used for the simulation. In Chapter 5 we present the results of our experiments and evaluate the performance of the retargeting system. Finally, in Chapter 6 we summarize our findings and discuss potential future work.

Chapter 2

Background

In this chapter, we review the theoretical foundations and existing methodologies that form the basis of our work. We first discuss the biomechanical principles of human hand control, specifically the concept of postural synergies. We then analyze standard approaches for mapping human motion to robotic hands, highlighting their limitations when dealing with dissimilar kinematics. Finally, we introduce the object-based retargeting approach, which provides the theoretical framework for the virtual sphere method used in this project.

2.1 Postural synergies

The human hand is a kinematic structure with more than 20 degrees of freedom (DoF) controlled by a complex network of muscles and tendons. Despite this mechanical complexity, humans are able to grasp objects of different shapes and sizes with ease and dexterity. Neuroscientific studies suggest that the central nervous system simplifies the control of the hand by coordinating the movement of multiple joints through a reduced set of control variables known as *postural synergies* [2]. These synergies represent patterns of joint coordination that capture the most significant variations in hand posture during grasping tasks.

In their study, Santello et al. [2] analyzed the hand postures of different subjects while grasping imaginary objects. By applying *Principal Component Analysis* (PCA) to the collected joint angle data, they identified that a large portion of the variance in hand postures could be explained by a very small number of principal components. Specifically, the first principal components (*synergies*) account for more than 80% of the variance in hand posture: the first synergy corresponds to the coordinated flexion and extension of all fingers (opening and closing the hand), resembling a power grasp (Figure 2.1, horizontal axis); the second synergy accounts for the abduction of the fingers and the opposition of the thumb, effectively controlling the arching of the palm (Figure 2.1, vertical axis). The remaining synergies can be used to fine-tune the hand posture for specific grasp types, but contribute increasingly less to the overall variance. We graphically illustrate the underlying flow of information in human hand control with the block diagram in Figure 2.2.

This finding is fundamental to our work: it implies that we do not need to measure every single joint of the human hand to understand its pose. Instead, we can capture the underlying correlations to reconstruct the full hand posture from the sparse data provided by the Weart glove, as demonstrated by the neural network approach [3] that we use.

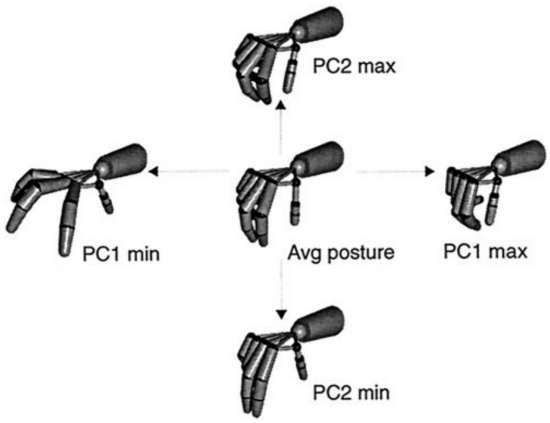


Figure 2.1: Visual representation of the first two postural synergies. Synergy 1 (on the horizontal axis) controls the general opening and closing of the hand. Synergy 2 (on the vertical axis) modulates the arching of the palm and finger adduction/abduction.

2.2 Standard motion retargeting strategies

Retargeting human hand motion to a robotic hand is a classic problem in robotics, often referred to as the *correspondence problem*. The challenge arises from the fact that robotic hands typically have different kinematic structures, sizes, and joint limits compared to the human hand. In literature, we can find two main strategies to address this problem.

Joint-to-joint mapping This is the most direct approach, where the joint angles of the human hand are mapped one-to-one to the corresponding joints of the robotic hand. If the robot is anthropomorphic, we can map human joints $\boldsymbol{q}_h$ directly to robot joints $\boldsymbol{q}_r$ using a linear transformation; if the robot has fewer joints, we need a mapping function to approximate the motion.

This method is straightforward and relatively easy to implement, but it fails with non-anthropomorphic hands (e.g., a 3-fingered gripper). Since the kinematic chains differ, applying human angles directly can lead to unnatural or infeasible robot postures, self-collisions, or the inability to grasp objects properly.

Fingertip (Cartesian) mapping To overcome the limitations of joint-to-joint mapping, another common approach is to focus on the positions of the fingertips rather than the joint angles. The Cartesian coordinates of the human fingertips $\boldsymbol{p}_h$ are computed through forward kinematics and then used as target positions for the robot's fingertips $\boldsymbol{p}_r$. An inverse kinematics solver is then employed to find the joint angles $\boldsymbol{q}_r$ that achieve these positions.

While this ensures that the fingertips reach the target, it completely ignores the internal configuration of the hand, potentially leading to unnatural grasps or excessive joint movements. Additionally, if the robot has fewer fingers than the human hand, it becomes unclear how to map multiple human fingertips to fewer robotic ones: a fingertip position reachable by a human might be a singular or unreachable configuration for the robot.

2.3 Object-based retargeting

To address the limitations of the standard retargeting methods, Gioioso et al. [1] proposed an approach defined in the *object domain*: instead of mapping the hand itself, either in joint or

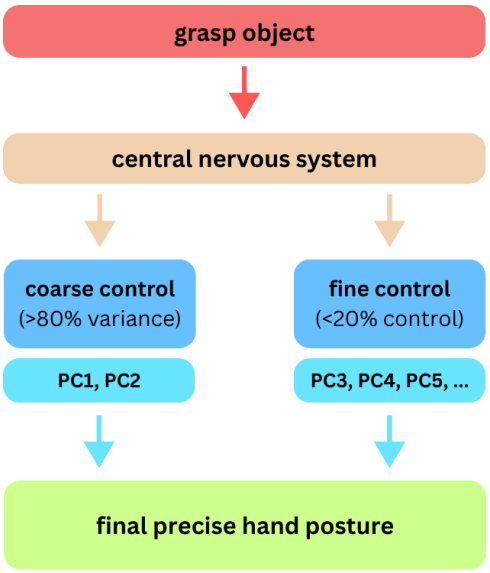


Figure 2.2: The hierarchical control scheme of the hand. Coarse control is achieved via the first two synergies ($> 80\%$ variance), while fine control is handled by higher-order synergies ($< 20\%$ variance).

Cartesian space, we map the effect the hand has on the object being manipulated. The key idea is that the primary goal of the hand is to interact with objects, so by focusing on the object, we can achieve more natural and effective grasps.

Since the object might not physically exist during teleoperation (e.g., in virtual reality scenarios), a *virtual object* (a sphere) is introduced in the grasping process. The method proceeds in the following steps:

1. A virtual sphere is mathematically fitted inside the human hand, defined by a set of reference points (e.g., fingertips and palm center). As the human hand moves via synergies, this sphere *translates*, *rotates*, and *deforms* accordingly.

2. The motion of the human sphere is scaled to match the size of the robotic hand, allowing a large human hand to control a small robotic gripper, or vice versa.

3. The robot is commanded to move its joints such that its own virtual sphere mimics the transformation of the human's sphere.

This object-based retargeting method is *independent* of the kinematic structure of both hands. It captures the intention of the grasp, namely whether the user is squeezing (shrinking the sphere) or moving the hand (translating the sphere), and it applies to the robot regardless of its number of fingers or joint structure.

Chapter 3

Framework

In this chapter, we present the mathematical formulation of the proposed motion retargeting framework. We detail the definition of the virtual sphere, the construction of the interaction matrix that relates hand motion to object deformation, and the derivation of the control law used to drive the robotic hand. Finally, we describe the redundancy resolution strategy employed to optimize the configuration of the robotic hand during manipulation task.

3.1 Reference points and sphere definition

The core concept of the object-based mapping is to abstract the hand's motion into the motion of a virtual object. We model this object as a *virtual sphere* defined by a set of reference points on the hand (see Figure 3.1 for an example). These reference points can include fingertips, the palm center, or any other significant point that captures the motion of the hand while grasping.

Let $\boldsymbol{p}_h \in \mathbb{R}^{3N_h}$ be the vector of the Cartesian positions of N_h reference points on the human hand. Similarly, let $\boldsymbol{p}_r \in \mathbb{R}^{3N_r}$ be the vector of N_r reference points on the robotic hand. At any time step t, the virtual sphere is defined as the *minimum enclosing ball*, i.e., the sphere of smallest radius that contains all reference points. Such a sphere is characterized by its center $\boldsymbol{o} \in \mathbb{R}^3$ and radius $r \in \mathbb{R}$.

The objective of the retargeting algorithm is to impose that the virtual sphere of the robotic hand mimics the rigid-body motion (translation and rotation) and the non-rigid deformation (scaling) of the human virtual sphere. To account for the size difference between the two hands, we introduce a scaling factor k_{sc}:

$$k_{sc} = \frac{r_r}{r_h} \tag{3.1}$$

where r_r and r_h are the radii of the robotic and human virtual spheres in their initial reference configurations, respectively.

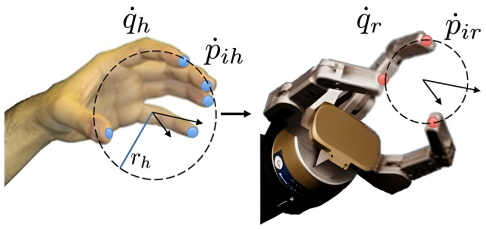


Figure 3.1: Example of virtual spheres defined by reference points on the human (left, blue dots) and robotic (right, red dots) hands.

3.2 Interaction matrix

To map the motion of the reference points to the motion of the sphere, we use the definition provided by Gioioso et al. [1]. The velocity of a generic reference points $\boldsymbol{p}_i$ can be expressed as a function of the sphere's linear velocity $\dot{\boldsymbol{o}}$, angular velocity $\boldsymbol{\omega}$, and rate of change of radius $\dot{r}$:

$$\dot{\boldsymbol{p}}_i = \dot{\boldsymbol{o}} + \boldsymbol{\omega} \times (\boldsymbol{p}_i - \boldsymbol{o}) + \dot{r}(\boldsymbol{p}_i - \boldsymbol{o}) \tag{3.2}$$

By stacking the velocities of all reference points, we can express the relationship between the sphere motion and the reference points' motion in matrix form:

$$\dot{\boldsymbol{p}} = \boldsymbol{A}\boldsymbol{v}_{\text{obj}} \tag{3.3}$$

where $\boldsymbol{v}_{\text{obj}} = [\dot{\boldsymbol{o}}^T, \boldsymbol{\omega}^T, \dot{r}]^T \in \mathbb{R}^7$ represents the generalized velocity of the virtual object. The matrix $\boldsymbol{A} \in \mathbb{R}^{3N\times 7}$ is the *interaction matrix*, constructed as follows:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_3 & -[\boldsymbol{p}_1 - \boldsymbol{o}]_\times & (\boldsymbol{p}_1 - \boldsymbol{o}) \\ \vdots & \vdots & \vdots \\ \boldsymbol{I}_3 & -[\boldsymbol{p}_N - \boldsymbol{o}]_\times & (\boldsymbol{p}_N - \boldsymbol{o}) \end{bmatrix} \tag{3.4}$$

Here, $\boldsymbol{I}_3$ is the 3×3 identity matrix, and $[\cdot]_\times$ denotes the skew-symmetric matrix operator for the cross product. This matrix relates the task-space velocities of the reference points to the deformation of the object.

3.3 Retargeting control law

Our retargeting strategy relies on mapping the velocity of the virtual object from the human domain to the robot domain. In the original formulation by Gioioso et al. [1], the human hand velocity is derived analytically from the synergy inputs.

Let $\boldsymbol{z} \in \mathbb{R}^{n_z}$ be the vector of synergy activation coefficients. The human joint velocities $\dot{\boldsymbol{q}}_h$ can be expressed as:

$$\dot{\boldsymbol{q}}_h = \boldsymbol{S}\dot{\boldsymbol{z}} \tag{3.5}$$

where $\boldsymbol{S} \in \mathbb{R}^{n_{q_h}\times n_z}$ is the *synergy matrix* mapping low-dimensional inputs to the full joint space. Consequently, the velocity of the human reference points $\dot{\boldsymbol{p}}_h$ would be computed as:

$$\dot{\boldsymbol{p}}_h = \boldsymbol{J}_h\dot{\boldsymbol{q}}_h = \boldsymbol{J}_h\boldsymbol{S}\dot{\boldsymbol{z}} \tag{3.6}$$

where $\boldsymbol{J}_h \in \mathbb{R}^{3N_h\times n_{q_h}}$ is the human hand Jacobian.

However, in our specific architecture, the reconstruction of the human hand pose is performed by a neural network [3], which directly outputs the full joint configuration $\boldsymbol{q}_h$ of the virtual human hand in Unity. Since the full pose is known at every frame, computing the analytical Jacobian $\boldsymbol{J}_h$ and the synergy matrix $\boldsymbol{S}$ explicitly is unnecessary. Instead, we obtain the reference point velocities $\dot{\boldsymbol{p}}_h$ via *numerical differentiation* of the tracked points in the virtual environment:

$$\dot{\boldsymbol{p}}_h(t) \approx \frac{\boldsymbol{p}_h(t) - \boldsymbol{p}_h(t - \Delta t)}{\Delta t} \tag{3.7}$$

This approach allows us to bypass the complexity of modeling the human kinematic chain analytically while ensuring that the motion fed into the retargeting algorithm accurately reflects the reconstructed hand pose.

Once $\dot{\boldsymbol{p}}_h$ is obtained, we compute the generalized velocity $\boldsymbol{v}_{\text{obj},h}$ of the human virtual sphere by inverting the interaction matrix. Since $\boldsymbol{A}_h$ is typically tall (more reference points than object DoFs), we us the *Moore-Penrose pseudoinverse* $\boldsymbol{A}_h^{\#}$:

$$\boldsymbol{v}_{\text{obj},h} = \boldsymbol{A}_h^{\#} \dot{\boldsymbol{p}}_h \tag{3.8}$$

Next, we map this motion to the robotic domain using a scaling matrix $\boldsymbol{K}_c \in \mathbb{R}^{7\times7}$:

$$\boldsymbol{v}_{\text{obj},r} = \boldsymbol{K}_c \boldsymbol{v}_{\text{obj},h} \tag{3.9}$$

The matrix $\boldsymbol{K}_c$ scales the translation and radial growth components of the object velocity by the factor k_{sc}, while leaving the angular velocity unchanged (which is independent of size):

$$\boldsymbol{K}_c = \begin{bmatrix} k_{sc}\boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 & \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} & k_{sc} \end{bmatrix} \tag{3.10}$$

We then compute the target velocity for the robot reference points $\dot{\boldsymbol{p}}_{r,\text{des}} = \boldsymbol{A}_r \boldsymbol{v}_{\text{obj},r}$. To track these points, we use the robot's differential kinematics equation $\dot{\boldsymbol{p}}_r = \boldsymbol{J}_r \dot{\boldsymbol{q}}_r$, where $\boldsymbol{J}_r$ is the Jacobian matrix mapping robot joint velocities $\dot{\boldsymbol{q}}_r$ to end-effector velocities $\dot{\boldsymbol{p}}_r$.

The final control law for the robot joint velocities is obtained by inverting this equaton:

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}_{r,\text{DLS}}^{\#} \dot{\boldsymbol{p}}_{r,\text{des}} \tag{3.11}$$

where $\boldsymbol{J}_r^{\#}$ is the *Damped Least Squares* (DLS) inverse of the robot Jacobian, which ensures numerical stability even when the robot is near singular configurations.

3.4 Redundancy resolution

In those scenarios where particularly dexterous robotic hands are employed, one can exploit their kinematic redundancy to optimize their internal configuration while performing grasping tasks.

To achieve this, we can augment the control law derived in Eq. 3.11 with a secondary objective using the *null-space projection* method: we exploit the null space of the Jacobian to perform secondary tasks without affecting the primary goal of tracking the virtual sphere motion. The modified control law becomes:

$$\dot{\boldsymbol{q}}_r = \underbrace{\boldsymbol{J}_{r,\text{DLS}}^{\#} \dot{\boldsymbol{p}}_{r,\text{des}}}_{\dot{\boldsymbol{q}}_{r,\text{primary}}} + (\boldsymbol{I} - \boldsymbol{J}_{r,\text{DLS}}^{\#} \boldsymbol{J}_r) \dot{\boldsymbol{q}}_0 \tag{3.12}$$

The term $(\boldsymbol{I} - \boldsymbol{J}_{r,\text{DLS}}^{\#} \boldsymbol{J}_r)$ projects an *arbitrary* velocity vector $\dot{\boldsymbol{q}}_0$ into the null space of the primary task. This allows us to define $\dot{\boldsymbol{q}}_0$ as the gradient of a performance criterion $H(\boldsymbol{q}_r)$ designed to keep the joints of the robotic hand away from the mechanical limits. Following standard kinematic control theory, we utilize the *joint range* availability function:

$$H(\boldsymbol{q}_r) = \frac{1}{2N} \sum_{i=1}^{N} \left(\frac{q_i - \bar{q}_i}{q_{i,\text{max}} - q_{i,\text{min}}} \right)^2 \tag{3.13}$$

where N is the number of joints, $q_{i,\text{max}}$ and $q_{i,\text{min}}$ are the upper and lower limits, and $\bar{q}_i$ is the midpoint of the range of joint i.

The secondary velocity task is defined as the steepest descent direction of this function:

$$\dot{\boldsymbol{q}}_0 = -\eta \nabla_{\boldsymbol{q}} H \tag{3.14}$$

where η is a positive scalar gain that regulates the influence of the secondary task, and the i-th component of the gradient is:

$$\frac{\partial H}{\partial q_i} = \frac{1}{N} \frac{q_i - \bar{q}_i}{(q_{i,\text{max}} - q_{i,\text{min}})^2} \tag{3.15}$$

This formulation normalizes the error, ensuring that joints with smaller ranges are prioritized (pushed harder towards the center of their range) comapred to joints with larger ranges.

For underactuated grippers with fewer degrees of freedom than the task requirements, the null space term naturally vanishes or has no effect. In these cases, the pseudoinverse in Eq. 3.11 provides the least-squares solution that minimizes the error between the desired and actual sphere motion.

Chapter 4

Experimental setup

In this chapter, we describe the complete hardware and software architecture developed to validate the proposed retargeting framework. We will first go through the data flow from the input device to the simulation environment, delve into the software implementation of the control algorithm, present the specific robotic hand models used in our experiments, and finally define the metrics employed to evaluate the performance of the whole architecture.

4.1 System overview

The motion retargeting pipeline shown in Figure 4.1 is designed as a modular system composed of three main blocks: the *input layer* (haptic glove and reconstruction server), the *retargeting layer* (unity simulation and retargeting logic), and the *output layer* (robotic hand models).

The data flow consists of the following steps:

1. *Acquisition*: the user wears the Weart TouchDIVER G1 haptic glove, which captures sensor data (finger closure and abduction) that are sent to a Python server.
2. *Reconstruction*: the Python server processes the glove inputs using a neural network [3] to reconstruct the full pose of the virtual human hand in Unity.
3. *Transmission*: the reconstructed joint angles are sent to the Unity simulation environment in real-time.
4. *Retargeting*: inside Unity, the retargeting algorithm detailed in Chapter 3 computes the deformation of the virtual sphere and derives the corresponding joint velocities for the target robotic hand model.
5. *Actuation*: the resulting joint velocities are integrated to update the visual pose of the robotic hand in the simulation.

4.2 Human input layer

The input interface is the Weart TouchDIVER G1, a wearable haptic device capable of tracking hand movements and rendering force, texture, and thermal cues. For the scope of this work, we focus on its motion tracking capabilities: it provides raw data about the closure of the thumb, index, and middle fingers, as well as the abduction of the thumb relative to the palm.

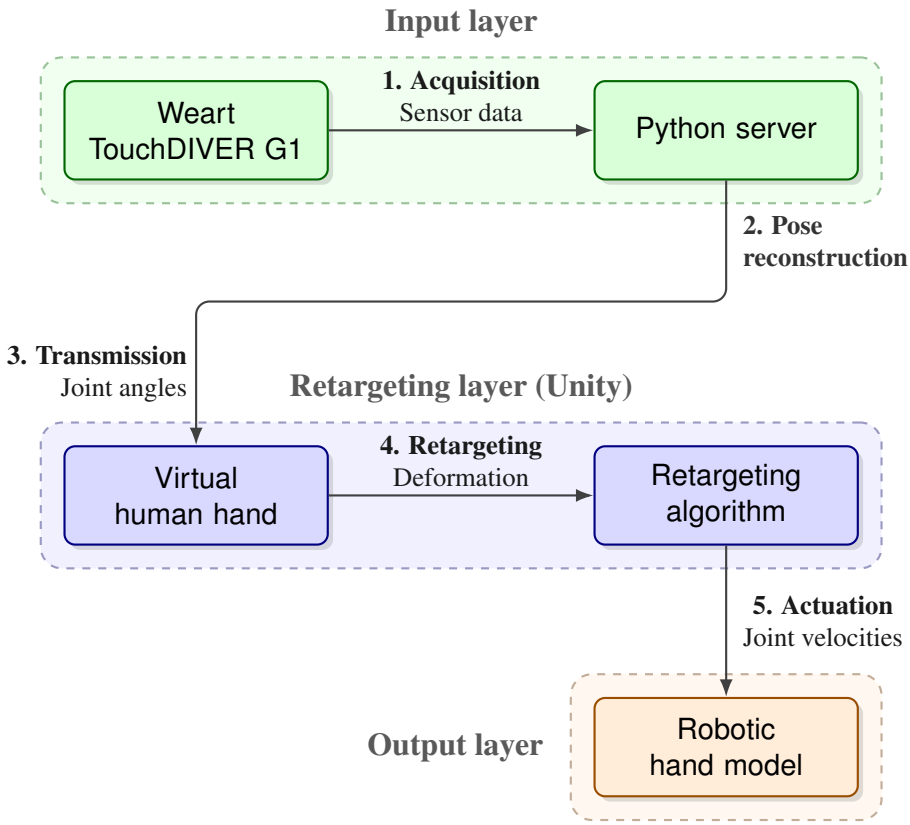


Figure 4.1: System architecture overview showing the complete motion retargeting pipeline. The data flows from the haptic glove through the reconstruction server, into the Unity-based retargeting layer, and finally to the robotic hand model for actuation.

Since the glove does not track the ring and pinky fingers, nor the individual phalanx rotations (MCP, PIP, DIP) explicitly, the raw sensor data is insufficient for teleoperation. To address this limitation, we employ the reconstruction module developed by Primiceri et al. [3]: a Python script acts as a server listening for incoming data from the glove, and feeds this data into a pre-trained neural network which outputs a 62-dimensional vector containing the sine and cosine encodings of all 15 human hand joints; this vector is then converted back into Euler angles and sent to the Unity client.

4.3 Software implementation

The core logic of the project is implemented in C# in the Unity engine. The implementation relies on the `MathNet.Numerics` library for efficient linear algebra computations, such as matrix multiplications and pseudoinverses.

Communication A dedicated thread handles the TCP communication with the Python server to separate the network operations from the main Unity thread, so as to not interfere with the simulation's frame rate. The received human pose $\boldsymbol{q}_h$ is applied to a kinematic model of the human hand, which serves as the *master* for the retargeting algorithm.

Kinematic solver To guarantee the system is adaptable to different robots, we developed a generalized `Kinematics` library: unlike standard solutions that hardcode the Jacobian matrix

for a specific robot, our implementation computes the robot Jacobian $\boldsymbol{J}_r$ dynamically. We define the robot structure in the Unity inspector by assigning a list of joint Transforms and their corresponding types (e.g., `HingeX`, `HingeY`, `HingeZ`, or `Ball`), allowing the software to handle any serial kinematic chain without code modification.

Virtual sphere The `VirtualSphere` script implements the mathematical framework described in Chapter 3 for the deformation and retargeting logic. It takes as input an array of `Transform` objects representing the reference points, and at every update:

1. it extracts the local positions of the reference points with respect to the palm;
2. it computes the *minimum enclosing ball* using Welzl's algorithm;
3. it constructs the interaction matrix $\boldsymbol{A}$ from Eq. 3.4 based on the sphere's current center and radius.

This script is attached to both the human hand and the robotic hand, providing the necessary matrices $\boldsymbol{A}_h$ and $\boldsymbol{A}_r$ to the main controller.

Code The complete source code developed for this project is open source and available on GitHub. [1].

4.4 Robot models

To evaluate the flexibility of the object-based retargeting approach [1], we selected robotic hands with significantly different kinematic structures to that of the human hand. In our experiments, we used three specific models: the *Barrett Hand*, the *Mia Hand*, and the *Shadow Dexterous Hand*. The kinematic descriptions (URDF models) for the Barrett and Mia hands were obtained from the embodiment framework by Fabisch et al. [4], while the Shadow Hand model was sourced from the Bunny-VisionPro framework [5]. All of them provide a full implementation of the physical properties (dynamics, collisions, limits, etc.) for realistic experiments, and were imported into Unity using the *URDF Importer* package.

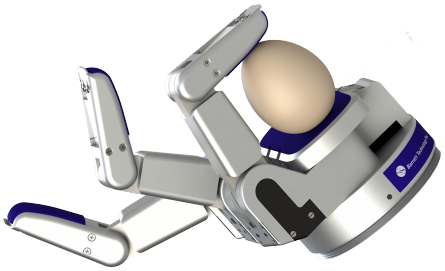
4.4.1 Barrett Hand

The *Barrett Hand* (Figure 4.2) is a multi-fingered programmable grasper whose kinematic structure is deeply different from the human hand, making it an interesting candidate for testing the robustness of the virtual sphere retargeting method.

Kinematics It consists of three fingers: one finger is fixed, while the other two can rotate synchronously around the palm (*spread motion*) up to $180°$. This allows the hand to change its configuration dynamically, switching from a parallel grasp with the three fingers aligned, to an opposition grasp with the two rotating fingers opposing the fixed one.

[1] https://github.com/yvhem/object_domain_synergy_mapping

(a) Kinematic structure



(b) Grasping an egg

Figure 4.2: The Barrett Hand. (a) shows the hand in an open configuration, highlighting the three-finger design (Source: ROS.org). (b) demonstrates the adaptability of the hand grasping an egg (Source: Barrett Technology).

Actuation The hand has 8 axes of motion but is *underactuated*, driven by only 4 motors: one controls the spread of the two movable fingers, and the other three control the flexion of each finger independently. [2] A proprietary "*TorqueSwitch*" mechanism couples the proximal and distal links: the distal link remains stationary until the proximal link encounters resistance, at which point the distal link curls to enclose the object. However, we are interested in evaluating the retargeting performance based on kinematics alone; therefore, in our kinematic model of the Barrett Hand, we treat all 8 axes as independent degrees of freedom.

4.4.2 Mia Hand

The *Mia Hand* (Figure 4.3) by Prensilia is an anthropomorphic end-effector designed primarily for prosthetics and research applications. Unlike the Barrett Hand, the Mia Hand has a kinematic structure that resembles that of the human hand, although with less dexterity.

(a) Anthropomorphic structure

(b) Cylindrical grasp

Figure 4.3: The Mia Hand. (a) shows the hand's resting posture, highlighting the mechanical design and soft pads. (b) shows the hand performing a cylindrical grasp, demonstrating finger coordination. Source: Prensilia.

[2] BarrettHand BH8-282 specs: https://barrett.com/barretthand

Kinematics The Mia Hand's dimensions are similar to an average human hand (palm width 83 mm). It consists of five fingers and an opposable thumb, designed to perform various grasp types such as cylindrical, spherical, and lateral grasps. [3]

Actuation To keep a lightweight profile (approx. 540 g), the Mia Hand employs an underactuated mechanism driven by 3 motors actuating the flexion/extension of the fingers and the opposition of the thumb. Again, like for the Barrett Hand, in our simulation we treat the hand's kinematic chain as fully articulated, meaning we control the individual joints of the fingers independently. Consequently, the kinematic model has more degrees of freedom ($N \approx 15$) than the virtual sphere task ($N = 7$), making the system *kinematically redundant* with respect to it. We therefore employ the redundancy resolution strategy described in Chapter 3 to distribute the motion among the joints, keeping them away from their mechanical limits while tracking the object deformation.

4.4.3 Shadow Dexterous Hand

The *Shadow Dexterous Hand* (Figure 4.4) is one of the most advanced anthropomorphic end-effectors available on the market, designed to reproduce the kinematics and dexterity of the human hand as closely as possible.

(a) Open configuration (b) Tool use (c) Grasping an object

Figure 4.4: The Shadow Dexterous Hand. Its 24 degrees of freedom allow it to perform complex manipulation tasks, from power grasps to precision tool use. Source: Shadow Robot.

Kinematics The Shadow Hand has 24 joints (actuated and passive) providing 20 independent degrees of freedom: the thumb has 5 joints and 5 DoF, allowing for complex opposition movements, while each of the four fingers has 4 joints but only 3 DoF due to the mechanical coupling of the distal joint (DIP) and the middle joint (PIP). This design ensures that the angle of the middle joint is always greater than or equal to that of the distal joint, mimicking the natural behavior of human fingers. The little finger has an extra joint in the palm to allow for opposition (forming a palm arch). The wrist provides 2 DoF.

Actuation Unlike the significantly underactuated Barrett and Mia hands, the Shadow Hand is highly actuated: indeed, it has 24 joints driven by 20 electric "*Smart Motors*" located in the

[3] Technical specification of the Mia Hand can be found at: https://www.mia-hand.com/

forearm base, which transmit force to the joints via tendons. [4] This high number of degrees of freedom makes the Shadow Hand highly *kinematically redundant* with respect to the 7-DoF virtual sphere task. Therefore, the redundancy resolution strategy defined in Eq. 3.12 becomes crucial to stabilize the internal configuration of the hand keeping the 20 actuated joints away from their mechanical limits, while the virtual sphere accurately defines the overall grasp shape.

4.5 Evaluation metrics

To quantitatively assess the performance of the retargeting algorithm, we define a set of metrics that measure how well the robotic hand tracks the desired object deformation and how effectively it uses its kinematic capabilities. Since the virtual sphere represents the core of our control strategy, these metrics focus on comparing the state of the virtual sphere of the robot against that of the human over time.

We define t_0 as the start time of the experiment. The subscripts H and R denote the Human (reference) and Robot (measured) quantities, respectively.

4.5.1 Geometry metrics

These metrics evaluate how accurately the robotic hand tracks the translation, rotation, and deformation of the virtual object.

Radius error This metric measures the ability of the system to replicate the scaling of the sphere, namely opening and closing of the hand. Since the robot and human hands have different sizes, the radius of the human sphere r_H must be scaled by a factor k_{sc} (defined in Eq. 3.1) before comparison. The absolute radius error $e_{\text{rad}}(t)$ is defined as:

$$e_{\text{rad}}(t) = |r_R(t) - k_{sc} r_H(t)| \tag{4.1}$$

Position variation error This metric assesses the "drift" of the object within the hand. Ideally, if the human hand translates the sphere by a certain vector, the robot should replicate that same displacement vector. We therefore compare the displacement relative to the initial position at t_0 to account for different starting offsets in the workspace.

Let $\boldsymbol{p}(t) = [x, y, z]^T$ be the position of the sphere center. The relative displacement is defined as:

$$\Delta \boldsymbol{p}(t) = \boldsymbol{p}(t) - \boldsymbol{p}(t_0)$$

The position variation error $e_{\text{pos}}(t)$ is then obtained as the magnitude of the difference between the human and robot relative displacements:

$$e_{\text{pos}}(t) = ||\Delta \boldsymbol{p}_H(t) - \Delta \boldsymbol{p}_R(t)|| \tag{4.2}$$

Rotation variation error This metric evaluates the tracking of the sphere's orientation. Let $\boldsymbol{q}(t)$ be the quaternion representing the sphere's rotation. The relative rotation from t_0 is computed as:

$$\boldsymbol{q}_{\text{rel}}(t) = \boldsymbol{q}(t) \otimes \boldsymbol{q}(t_0)^{-1}$$

where $\otimes$ denotes quaternion multiplication and $^{-1}$ denotes the quaternion conjugate.

[4] Shadow Dexterous Hand technical specification: https://shadowrobot.com/

We define an extraction operator $\Phi(\boldsymbol{q})$ that returns the rotation angle (magnitude) from a quaternion:

$$\Phi(\boldsymbol{q}) = 2\arccos(q_w)$$

The rotation error $e_{\text{rot}}(t)$ is then defined as the angle of the difference quaternion between the human and robot relative rotations:

$$\boldsymbol{q}_{\text{diff}}(t) = \boldsymbol{q}_{R,\text{rel}}(t) \otimes \boldsymbol{q}_{H,\text{rel}}(t)^{-1} \quad \Rightarrow \quad e_{\text{rot}}(t) = \Phi(\boldsymbol{q}_{\text{diff}}(t)) \tag{4.3}$$

4.5.2 Energy metrics

Besides the geometry of the virtual sphere, we are also interested in evaluating the *intensity* of the grasp. In the virtual sphere framework, the squeezing of the object corresponds to an accumulation of elastic potential energy in the virtual stiffness model.

Elastic energy error $R_H^{(t)}$ and $R_R^{(t)}$ be the human and robotic hand virtual sphere radius, respectively, at a generic time t. The deformation is then computed as the normalized difference between the current radius and the initial one:

$$\Delta_{H/R} = \frac{\max(0, R_{H/R}^0 - R_{H/R}^{(t)})}{R_{H/R}^0} \tag{4.4}$$

Since the final aim is to evaluate a grasping task, the deformation is only relevant when the sphere is compressed, namely when $R_{H/R}^0 - R_{H/R}^{(t)} > 0$, thus its value is clamped to be non-negative.

Let N_H be the number of contact points on the human virtual sphere, and N_R be the one on the robotic virtual sphere. Let's define the ratio between them as the constant $k_{cp} = \frac{N_H}{N_R}$. The elastic energy is then computed as:

$$U_{H/R}(t) = \frac{1}{2} k_{cp} k_s (\Delta_{H/R}(t))^2 \tag{4.5}$$

where k_s is the spring stiffness, for simplicity set to 1. Having now both human and robotic energy, the absolute error is:

$$e_{\text{energy}}(t) = |U_R(t) - U_H(t)| \tag{4.6}$$

Relative percentage error To normalize the energy performance, we compute the relative percentage error. To avoid numerical instability when the energy is near zero (e.g., when the hand is open), we introduce a small constant $\epsilon = 1 \times 10^{-9}$ and apply a conditional mask to zero out the error during idle phases (when $U_H(t) \leq 5\%$ of its maximum).

$$e_{\%}(t) = \frac{|U_R(t) - U_H(t)|}{U_H(t) + \epsilon} \times 100 \tag{4.7}$$

Finally, for an easy interpretation of the results, we smooth the percentage error using a moving average filter $\mathcal{S}$ over a window of 10 frames:

$$e_{\%,\text{smooth}}(t) = \mathcal{S}(e_{\%}(t))$$

4.6 Qualitative validation

The quantitative metrics defined above provide a measure of kinematic accuracy, but they do not capture the functional success of a grasp in a practical scenario. To assess the applicability of the retargeting algorithm in real-world manipulation tasks, we also perform qualitative evaluations by simulating various grasping tasks within the Unity environment.

We selected three objects of standard geometric shapes to test different grasp types:

- *Sphere*: to test spherical grasps and the ability to conform to curved surfaces.
- *Cube*: to evaluate the ability of the robotic fingers to adapt to flat surfaces and edges.
- *Cylinder*: to assess the performance in cylindrical grasps, which are common in everyday tasks.

For each object, we simulate a grasping task using the Weart glove as the input device. We visually inspect the resulting robotic motion to determine two key aspects:

- Whether the robot fingers enclose around the object in a natural way without unfeasible interpenetration or unnatural internal joint configurations;
- whether the virtual object remains stably within the robotic hand during the closure phase.

4.6.1 Simulation physics setup

Simulating contact interactions between rigid bodies in Unity is complex, and a wrong configuration can lead to unrealistic behaviors such as the two models flying away from each other.

To mitigate this issue and simualate a realistic interaction, we introduced linear and angular *damping* to the manipulated objects. This mimics the effect of *friction* between the hand and the object, preventing this latter from reacting too abruptly to sudden contact forces.

Furthermore, we disabled gravity for the target objects. Since our experimental setup focuses strictly on the *grasping* phase, we skip the *approach* phase (where the hand moves towards the object) and fix the base of the robotic hand in space. Disabling gravity allows the object to remain suspended in the ”grasping workspace”, ensuring that the evaluation focuses solely on the closure of the fingers and the synergy mapping, rather than the robot's ability to chase a falling object.

Finally, it is important to note that the outcome of these tests is heavily influenced by the quality of the *colliders* of the robotic models: inaccurate or simplified collision meshes, such as box colliders, can lead to missed contacts or interpenetrations, affecting the realism of the grasp.

Chapter 5

Results

Chapter 6

Conclusion

6.1 Summary

6.2 Future work

Bibliography

[1] G. Gioioso, G. Salvietti, M. Malvezzi, and D. Prattichizzo, "Mapping synergies from human to robotic hands with dissimilar kinematics: An approach in the object domain," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 825–837, 2013.

[2] M. Santello, M. Flanders, and J. F. Soechting, "Postural hand synergies for tool use," *The Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, 1998.

[3] C. L. Primiceri, S. Trovalusci, D. I. Bubenek Turconi, and G. Pagano, *Motion retargeting on a human hand model*, M.Sc. Project Report, Sapienza University of Rome, Code available at: https://github.com/serenatrovalusci/Hand-Motion-Reconstruction-from-Minimal-Inputs-through-Latent-Space-Learning, Jun. 2025.

[4] A. Fabisch, M. Uliano, D. Marschner, M. Laux, J. Brust, and M. Controzzi, "A modular approach to the embodiment of hand motions from human demonstrations," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 801–808. DOI: 10.1109/Humanoids53995.2022.10000165.

[5] R. Ding, Y. Qin, J. Zhu, *et al.*, "Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning," 2024. [Online]. Available: https://arxiv.org/abs/2407.03162.