cHand: Open Source Hand Posture Visualization in CHAI3D

Edoardo Battaglia and Ann Majewicz Fey

***Abstract*— Visualization of hand movement is an important part of many medical training, VR and haptic experiences, which researchers typically address by developing application specific hand visualization tools. While some existing simulators allow for hand kinematic visualization using a generic hand model, they are usual targeted for robotic grasp planning rather than designed specifically for rendering in applications that include haptic experiences. To fill this gap, in this paper we present cHand, an extension of the haptics software library CHAI3D, that enables hand kinematic visualization of an arbitrary hand model. A representation of the hand can be achieved with elementary geometric shapes that are provided by CHAI3D, or with custom geometries loaded from STL files. We release cHand as an open source contribution to keep with the open source nature of CHAI3D, and present a tutorial on its use in this manuscript.**

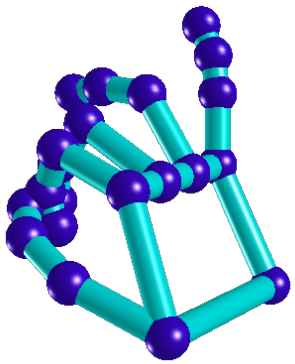


Fig. 1: Example of visualization obtained with cHand.

I. Introduction

Hands play an important role in our interaction with the environment, both because they are the primary means through which we interact with the physical world around us, and because of their strong perceptual capabilities [1]. Visualization of hand posture is thus a desirable feature for many applications, yet there are few widely available tools that can be used to quickly visualize a generic hand model, and none of them were specifically designed with haptic applications in mind. Indeed, most efforts from the research community to provide tools for simulating hands has focused on robotic hands and grasp planning.

For example, in 2004 Miller and Allen introduced Graspit! [2], which is still being maintained today and focuses on the simulation of robotic grasps, with support for data acquisition from hardware being offered for a few specific devices, most of them robotic hands. Leon et al. released Opengrasp [3] in 2010 for similar purposes and comparable features but, to the best of our knowledge, this tool has not been updated since 2011. A third example is Syngrasp [4], [5], which is a MATLAB toolbox aiming to simulate both human and robotic hands that was released in 2013 and is still being maintained. Syngrasp also focuses more on grasp simulation and planning than on live visualization based on data received from hardware, as evidenced by the choice of using a language such as MATLAB, which is most powerful when used for data analysis and post processing, instead of a faster platform such as C++.

This work was partially supported by the NSF: NRI: grant FND: "Customizable Haptic Co-Robots For Training Emergency Surgical Procedures"

E.B. is with the Department of Mechanical Engineering, The University of Utah, Salt Lake City, UT 84112, USA. edoardo.battaglia@utah.edu

A.M.F. is with the Department of Mechanical Engineering, The University of Texas at Austin, TX 78712, USA.

A.M.F. is also with the Department of Surgery, UT Southwestern Medical Center, Dallas, TX 75390, USA

One example of a library specifically designed for haptics is CHAI3D [6], which was released in 2013 in the effort to provide a ready for use platform for haptic researchers and developers. CHAI3D offers a rich set of features including a C++ API, graphic rendering through OpenGL, templates to help with rapid prototyping, built-in force feedback algorithms and hardware support for a variety of commercial haptic devices, with the ability from the user to easily add custom haptic devices. This library was also the first haptic software development kit to be released as a non commercial, completely open source project, which is a desirable feature to foster engagement from the research community.

Despite its many features, CHAI3D does not provide an interface to visualize hand movement. When CHAI3D was first developed, haptic research primarily focused on grounded devices, such as the Phantom [7], for which visualization of kinematics revolves around visualizing the device itself, rather than the hand using it. However, recent years have seen a surge in applications of wearable haptics [8], which often involve the use of wearable sensors to track hand kinematics in order to provide haptic feedback as a response to interactions happening in a virtual environment. The lack of hand visualization capabilities currently represents a gap between what CHAI3D can offer and what is needed by the community.

This has led researchers to develop application specific visualization methods [9], [10], [11], [12], [13], often relying on commercial rendering software or game engines. While this approach can be convenient in the fact that it allows designers to get up and running with a virtual environment relatively quickly, it comes with its drawbacks, such as the large overhead from using complex software that has many features that are not useful for haptic applications, and the lack of some specific features desirable to better simulate haptic interaction. In fact, Balzarotti and Baud Bovy

presented in [14] an attempt to create a bridge between CHAI3D and the game engine Unity3D, and found that there are intrinsic limits to how accurately Unity3D can simulate haptics in physical interaction scenarios, so much so that it limited what could be done in their integration with CHAI3D.

In order to fill this gap, in this paper we present cHand, an extension of CHAI3D that allows users to define and visualize a generic hand model. We will first present an overview of the main features provided as well as a description of relevant public functions, and then show a few examples of applications, which will serve as a short tutorial on how to get started with the tool, similarly to what was done for CHAI3D in [6]. In the final section, we describe a setup where cHand was used with a sensor and a motor for live data visualization. The CHAI3D extension cHand, together with all the code that we refer to in this paper, is open source and available on GitHub[1].

II. Overview and Implementation

With cHand we aim to enable visualization of a generic hand model, building upon the existing framework in CHAI3D. In order to accomplish this goal there are some requirements that need to be met:

- The user needs to be able to define a generic hand model for the software to process;
- Once the model is defined, a visualization needs to be rendered and its kinematics updated as necessary based on changes of joint angle valueis;
- There needs to be a way to detect contacts between the rendered hand visualization and other objects in the virtual world;
- All of the above need to be accomplished by leveraging as much as possible upon the existing structures existing in CHAI3D and keeping compatibility with the rest of the library.

In order to explain how we fulfilled these requirements, we need to briefly describe some elements of the architecture of CHAI3D itself. CHAI3D has a hierarchical structure, built starting from a base class cGenericObject which is then used to define all renderable elements through inheritance. Since this base class defines methods to store and obtain information on position and orientation, all objects derived from it have a rigid transformation associated to them, which allows to define and move them in the virtual space.

To insert cHand in this structure, we decided to implement it as a class derived from cGenericObject. Figure 2 shows a partial[2] representation of the inheritance diagram for CHAI3D, showing how cHand inserts itself into it. In addition to the features it inherits from cGenericObject, cHand contains attributes allowing it to store information on the hand model through a matrix, implemented using `std::vector`, where the first level represents a finger and the second level individual joints in each finger. More in detail, the following main features are implemented:

[1] https://github.com/ebattaglia/cHand/

[2] We refer to the CHAI3Ddocumentation for a complete description https://www.chai3d.org/download/doc/html/

	Function	Description
Initialization	`initialize_transforms`	Defines hand kinematics (joints position and orientation) through a collection of cTransforms
	`initialize_graphics`	Overloaded function used to build graphic visualization based on either primitive shapes, or STL files stored locally, depending on how it is called
	`initialize`	Calls `initialize_transforms` and `initialize_graphics` in sequence
Kinematics	`updateAngles`	Updates values for each joint angle
	`updateKinematics`	Updates the kinematic of the hand based on the current joint angles
Utility	`makeTFromFile`	Loads a text file containing information on relative transforms between hand joints and returns a collection of cTransforms in the format expected by `initialize_transforms`
	`getHandCenters`	Returns the position of centers of rotations for all joints, plus the center of fingertips (used for contact detection)
	`toggleArrows`	Toggles visualization of rotation axes
	`getdof`	Returns the total number of degrees of freedom
	`getnfingers`	Returns the number of fingers
	`getdof_finger`	Returns the number of degrees of freedom for a given finger

TABLE I: Code structure: main public member functions.

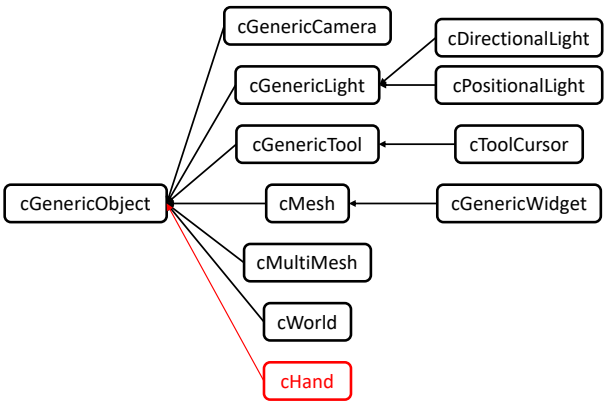


Fig. 2: Simplified inheritance diagram for CHAI3D.

- A matrix of pointers to cGenericObject which is used to store information on the relative position and orientation of all joints in the hand (`joint_transforms_container`);
- A matrix of pointers to cMultiMesh which is used to store graphic objects used for representation of the hand, plus a separate pointer to cMultiMesh that is used to represent the palm;
- A vector of doubles containing the current values of each joint angle in the model;
- A set of additional parameters containing information on the hand model, such as the number of fingers, degrees of freedom per finger and overall number of degrees of freedom.

These are the main elements defining the representation

of a hand, which are built and interacted with by using a set of member functions, the most relevant of which are summarized in Table I. Hand kinematics are initialized by calling `initialize_transforms`, requiring as input a matrix of cTransform. This matrix can be either defined explicitly in the code, or loaded from a text file with the function `makeTFromFile`.

Once the hand model kinematics are defined, the graphic visualization can be built by calling `initialize_graphics`. Two overloaded versions of this function are provided, one that builds a visualization with simple graphic primitives (cylinders and spheres) defined internally by CHAI3D, and one that loads STL files and requires a path to where they are stored locally, a vector of `cTransform` defining the local transformation in the reference frame where the STL file was saved, and a vector of integers containing IDs of parent joints for each graphic object, with 0 being used to define the base frame.

Once the cHand model and graphic representation are initialized, changing kinematics is simply a matter of updating the stored joint angle values (`updateAngles`) and updating the transformations accordingly by calling `updateKinematics`. Finally, contact detection can be accomplished by calling the function `getHandCenters`, which returns the current position of centers of rotations as well as positions of the fingertips, and using the built in contact models offered by CHAI3D[3].

In the next sections we will show a few examples of ways that cHand can be used, including a demonstration where hardware is used to read live data that can serve as a template for other applications.

III. Using cHand

This section showcases a few of the features of cHand, and is meant to be a quick tutorial on how to get started with it. Some GUI element are present in these examples, which were created using the free and open source Dear ImGui[4] user interface library. While we will include some code snippets, for readability's sake we will keep the lines of code to a minimum and focus on concepts, and we encourage the reader to browse the example files provided on the cHand repository for more details.

A. Hand model definition

In order to make its use as easy as possible, cHand provides some default hand models that the hand visualization can be initialized to. We chose as one of these models the one introduced by Tkach et al. in [15] and described more in detail in [16], for two reasons: (i) it is a fairly complex hand model, with a high number of degrees of freedom, which means that many hand models can be represented with this one by simply keeping some joint angles to zero, and (ii) authors in [16] released a large data set of hand postures, providing open source measurements readily available for use. Initializing a virtual hand with such a default model is straightforward and can be accomplished as follows:

```
HandModel = new cHand();
world->addChild(HandModel);

std::vector<std::vector<cTransform>> T =
    HandModel->t_default_Tkach25Dof;

HandModel->initialize(T);
```

Figure shows the resulting visualization from the example project *HandVisualizerGUI_TransformsDefinition*, which provides a simple GUI to visualize changes in joint angle values. This example also shows how to modify the matrix of transformations used to define the hand model to obtain desired finger lengths, done by simply assigning new values to the translation terms before calling `initialize`.

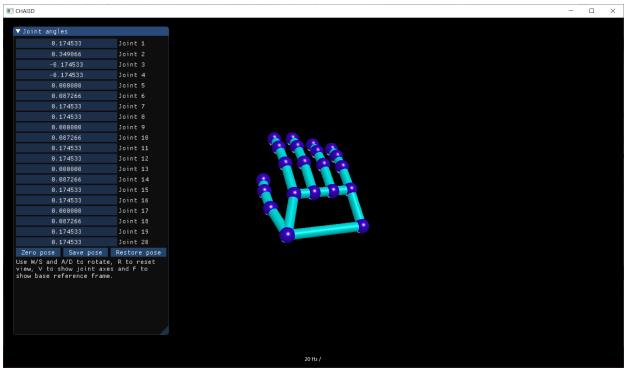


Fig. 3: Simple hand model.

If a custom hand model is desired, translation, rotation and the overall transformation of each joint can be easily defined using the structures provided by CHAI3D, e.g.:

```
cVector3d f1joint1pos(-0.05f, 0.045f,
    -0.0f);

cMatrix3d f1joint1rot(cVector3d(1, 0, 0)
    , 0);

cTransform Tf1j1(f1joint1pos,
    f1joint1rot);
```

and the chain of transformations can be defined for each finger as:

```
vector<cTransform> Tf1;
\\ add more for more fingers

vector<vector<cTransform>> Tmodel;

Tf1.push_back(Tf1j1);
\\ repeat for all joints in each finger

Tmodel.push_back(Tf1);
\\ repeat to add all fingers to the
    kinematic model
```

The visualization can then be obtained by simply calling `initialize(TModel)`.

It is worth pointing out that the hand model does not necessarily have to be a five-fingered model. Figure 4 shows

[3] A simplified, kinematic based contact model can also be implemented when no classical kinesthetic force feedback is present, as we will show in the last section

[4] https://github.com/ocornut/imgui

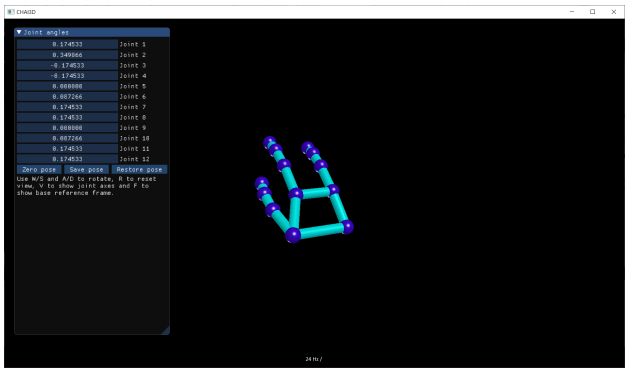


Fig. 4: Custom hand model with three fingers.

Fig. 5: Hand model from [19] with STL visualization of bones.

the visualization for a three-fingered hand model obtained using the procedure described above, but a higher number of fingers can also be used if desired. This could be useful for example for some applications in teleoperation or augmented reality where a robotic manipulator is controlled by a user receiving haptic feedback [17], or for scenarios where it is desirable to change the representation of the hand [18].

B. Loading complex and hand model data from local files

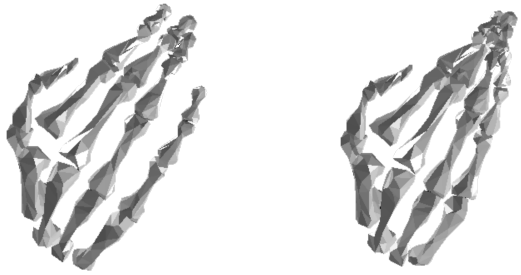
In the previous subsection we showed some examples of visualization obtained through simple geometries defined by CHAI3D. In some cases a more customized visualization can be desirable, and for this reason cHand supports loading custom geometries from STL files. The idea is to attach a geometry to each joint, so that it moves as the local reference frame moves according to the global kinematic. This is implemented with two overloaded versions of `initialize_graphics`, one that can be called with no arguments and builds the standard visualization with simple shapes, and one that requires additional arguments as well as local files to load the necessary geometries.

The arguments needed for the latter are the following:

1) A vector of strings, each containing the path to the STL mesh file to be loaded;
2) A vector of cTransform, each describing the local transformation of each STL file as observed in the reference frame that the file was saved in;
3) A vector of integers containing the mesh map, i.e. a parametrization of which joints have a geometry associated to them, with 0 being used to design the base frame;

Figure 5 shows the visualization for a hand model obtained from an MRI scan in [19], for which bone geometries, kinematic description and some examples of grasp are provided as part of the open access repository HandCorpus [20][5]. It is worth noticing that in this case, in addition to loading the STL geometries from file, the hand model itself is also loaded from a .dat text file, using the member function `makeTFromFile` provided by cHand. Additionally, this model includes additional degrees of freedom for arcpalmar joints, i.e. degrees of freedom on the palm between the bases

[5] https://www.handcorpus.org/?p=97



(a) Zero pose. (b) Arcpalmar movement.

Fig. 6: Arcpalmar degrees of freedom from the model defined in [19], as visualized through cHand.

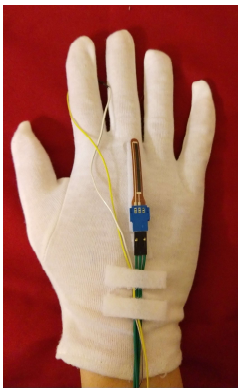
of the kinematic chains of each finger. This is a feature that is not very common in kinematic models of hands, but it can be nonetheless desirable for some applications [21], and is supported by cHand through an optional argument in `initialize_transforms`.

IV. Visualizing Live Data with cHand

The true strength of our cHand is to enable visualization of hand kinematics in a live experiment, based on sensor data. In this section we show a simple experimental scenario using a glove which is meant to provide a template that can be used to get started with other projects.

Figure 7 shows the sensorized glove that was used for this demonstration. To keep the setup simple and easy to reproduce, we endowed the glove with a single bend sensor (3" bidirectional bend sensor from Flexpoint) and one small vibration motor (10 mm shaftless vibration motor 310-101.945, from Precision Microdrives). The sensor is placed over the metacarpophalangeal middle finger joint, while the vibrotactile feedback is located on the fingertip of the index finger. An Arduino Micro board was used to handle USB communication from the glove to a laptop, where CHAI3D and cHand were ran in a Visual Studio 2019 environment. A simple voltage divider was used to read the variable resistance from the bend sensor.

In order to be able to measure the full hand kinematics with bend sensors, one would need to place one on each



(a) Top view.

(b) Bottom view.

Fig. 7: Simple haptic glove used for live demonstration.

finger, building a sensing glove similar to the CyberGlove[6]. However, this also introduces a lot of complexity in terms of added hardware and calibration. Instead we chose to use a single sensor to provide a coarse estimation of the full hand posture using the concept of *postural hand synergies*. Postural synergies are coordinated joint patterns that have been observed in human hands movement [22], and have led to applications in robotics for the design and control of artificial hands [23], [24], [25], as well as sensing of human hands [26], [9].

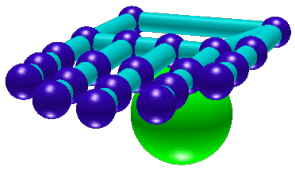
Such coordination can be quantified by performing a Principal Component Analysis (PCA) of collections of complete hand joint measurements, and evaluating the amount of variance in the dataset that is explained by the first few Principal Components (PCs). A linear combination of these PCs can then used to represent an approximation of the entire joint angle space. More formally if, n is the number of degrees of freedom of the hand model, we can use the first p PCs to approximate a joint angle vector $\boldsymbol{\theta}$ as:

$$\boldsymbol{\theta} \approx c_1 P_{c_1} + \ldots + c_p P_{c_p}, \qquad \text{with } \boldsymbol{\theta}, P_{c_i} \in \mathbb{R}^n \tag{1}$$

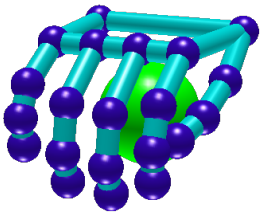
Santello et al. in particular observed in [22] that on average around 60% of the hand movement associated with their dataset was explained by just the first postural synergy. It is then reasonable to think about obtaining an approximation of hand posture from our single sensor through the first postural synergy, where the bend sensor controls the level of opening and closure of the hand (Figure 8b). Since the kinematic model that was used in [16] is pre-built in cHand, and since authors of [16] released their data set publicly, we were able to run a PCA on that dataset and use the outcome to control the visualization of hand closure as mentioned above.

To make the demonstration more interesting, we added an spherical object for the user to interact with and through which to obtain vibrotactile feedback, as shown in Figure 8. Contact detection is implemented by using the cHand member function `getHandCenters` to obtain the position of centers of rotation for each joint, as well as the fingertips (i.e., all elements represented by blue spheres in the figure)

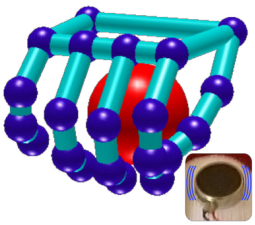
[6]http://www.cyberglovesystems.com/



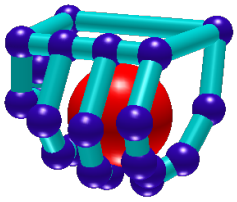
(a) Initial hand in the zero pose, before the data acquisition session is started.



(b) As the user closes the hand the sensor reading increases and the hand starts closing.



(c) Contact detection triggers a color change in the grasped object and a haptic cue to the user.



(d) The fingers not in contact keep moving, while those interacting with the object do not.

Fig. 8: Simulation of contact with an object based on hand postural soft synergies.

and then comparing their distance from the center of the object with the sphere radius. Once contact is detected on any of the above mentioned points, the sphere changes color and a vibrating pulse is triggered (Figure 8c). After this initial contact, additional closure of the user's hand as measure by the bend sensor is handled by only moving the portions of the kinematic chain that are distal with respect to the detected contacts, following the principle of the *soft synergy* model for grasp and manipulation which augments postural hand synergies with compliance [27].

It is worth pointing out that this is not the standard way that contact detection happens in CHAI3D. As also observed in [14], the primary way to do contact detection in CHAI3D is based on a force model, which means that it can not be used for applications such as vibrotactile feedback where there is no direct force exchange. The ability to extract the position of the centers of rotation and fingertip of the hands allows to get around that, where the problem turns then into defining a collision detection box around the interaction object and checking if points on the hand are inside it (which in the case of interaction with a sphere can be done by simply checking the distance of a point from its center). This enables the definition of contact models for wearable haptic applications that might not necessarily include a kinesthetic force component.

This simple demonstration of interfacing cHand with sensors and actuators serves as starting point for other projects. The files for this project, including the Arduino sketch, are available for download from the cHand repository, together with files for the previously mentioned examples.

V. Conclusions

In this paper we presented cHand, an extension of the CHAI3D haptic library that augments it with support for visualization of hand kinematics. Features include definition of a generic hand model, which can be defined in the code or loaded from file, visualization through both simple geometrical shapes and more complex custom geometries from STL models, and the possibility to define contacts kinematically from the position of the centers of rotation of the hand and the fingertips. In addition to introducing the add on and its features, we provided a series of examples meant to serve as tutorials on how to get started with cHand. In the last section, we described an easy to reproduce experimental setup showing how to use cHand for live data acquisition. This can serve as a template for future projects involving other wearable haptic systems.

We believe that the addition of built in hand visualization capabilities to CHAI3D can be beneficial for several applications in haptics. Where in the past researchers in wearable haptics had to choose between using CHAI3D with no immediate way to visualize hand kinematics, or other software such as game engines which can provide easier visualization, but no real support for haptic applications, they now have the option of getting up and running immediately with hand models in CHAI3D. Given the recent growth in the interest for this relatively new field of haptics [8], we think that this tool can be beneficial for the community, and we release it as an open source project.

Future work will revolve around addressing some of the limitation of the current version, such as providing kinematic contact detection not just on fingertips and centers of rotation, but also on the rest of the hand. Additionally the demonstration described in the final section shows an example of non force based contact detection for contact with a simple geometry. Detection of contact with more complex geometries can be performed using built in CHAI3D tools to calculate a bounding box, but such calculations are not exposed to the user. Future development of cHand will address this by providing exposure through the cHand class.

References

[1] S. J. Lederman and R. L. Klatzky, "Haptic perception: A tutorial," *Attention, Perception, & Psychophysics*, vol. 71, no. 7, pp. 1439–1459, 2009.

[2] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[3] B. León, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, *et al.*, "Opengrasp: a toolkit for robot grasping simulation," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2010, pp. 109–120.

[4] M. Malvezzi, G. Gioioso, G. Salvietti, D. Prattichizzo, and A. Bicchi, "Syngrasp: A matlab toolbox for grasp analysis of human and robotic hands," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1088–1093.

[5] M. Malvezzi, G. Gioioso, G. Salvietti, and D. Prattichizzo, "Syngrasp: A matlab toolbox for underactuated and compliant hands," *IEEE Robotics & Automation Magazine*, vol. 22, no. 4, pp. 52–68, 2015.

[6] F. Conti, F. Barbagli, R. Balaniuk, M. Halg, C. Lu, D. Morris, L. Sentis, J. Warren, O. Khatib, and K. Salisbury, "The chai libraries," in *Proceedings of Eurohaptics 2003*, Dublin, Ireland, 2003, pp. 496–500.

[7] T. H. Massie, J. K. Salisbury, *et al.*, "The phantom haptic interface: A device for probing virtual objects," in *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, vol. 55, no. 1. Chicago, IL, 1994, pp. 295–300.

[8] C. Pacchierotti, S. Sinclair, M. Solazzi, A. Frisoli, V. Hayward, and D. Prattichizzo, "Wearable haptic systems for the fingertip and the hand: taxonomy, review, and perspectives," *IEEE transactions on haptics*, vol. 10, no. 4, pp. 580–600, 2017.

[9] S. Ciotti, E. Battaglia, N. Carbonaro, A. Bicchi, A. Tognetti, and M. Bianchi, "A synergy-based optimally designed sensing glove for functional grasp recognition," *Sensors*, vol. 16, no. 6, p. 811, 2016.

[10] M. Azmandian, M. Hancock, H. Benko, E. Ofek, and A. D. Wilson, "Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences," in *Proceedings of the 2016 chi conference on human factors in computing systems*, 2016, pp. 1968–1979.

[11] E. Battaglia, M. G. Catalano, G. Grioli, M. Bianchi, and A. Bicchi, "Exosense: measuring manipulation in a wearable manner," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2774–2781.

[12] E. Pezent, A. Israr, M. Samad, S. Robinson, P. Agarwal, H. Benko, and N. Colonnese, "Tasbi: Multisensory squeeze and vibrotactile wrist haptics for augmented and virtual reality," in *2019 IEEE World Haptics Conference (WHC)*. IEEE, 2019, pp. 1–6.

[13] S. V. Salazar, C. Pacchierotti, X. de Tinguy, A. Maciel, and M. Marchal, "Altering the stiffness, friction, and shape perception of tangible objects in virtual reality using wearable haptics," *IEEE transactions on haptics*, vol. 13, no. 1, pp. 167–174, 2020.

[14] N. Balzarotti and G. Baud-Bovy, "Hpge: an haptic plugin for game engines," in *International Conference on Games and Learning Alliance*. Springer, 2018, pp. 330–339.

[15] A. Tkach, A. Tagliasacchi, E. Remelli, M. Pauly, and A. Fitzgibbon, "Online generative model personalization for hand tracking," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 6, pp. 1–11, 2017.

[16] O. Glauser, S. Wu, D. Panozzo, O. Hilliges, and O. Sorkine-Hornung, "Interactive hand pose estimation using a stretch-sensing soft glove," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–15, 2019.

[17] D. Lee and Y. S. Park, "Implementation of augmented teleoperation system based on robot operating system (ros)," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5497–5502.

[18] V. Schwind, L. Lin, M. Di Luca, S. Jörg, and J. Hillis, "Touch with foreign hands: The effect of virtual hand appearance on visual-haptic integration," in *Proceedings of the 15th ACM Symposium on Applied Perception*, 2018, pp. 1–8.

[19] G. Stillfried, U. Hillenbrand, M. Settles, and P. van der Smagt, "Mri-based skeletal hand movement model," in *The human hand as an inspiration for robot hand development*. Springer, 2014, pp. 49–75.

[20] M. Bianchi and M. V. Liarokapis, "HandCorpus, a new open-access repository for sharing experimental data and results on human and artificial hands," in *IEEE World Haptics Conference (WHC)*, April 2013.

[21] M. Gabiccini, G. Stillfried, H. Marino, and M. Bianchi, "A data-driven kinematic model of the human hand with soft-tissue artifact compensation mechanism for grasp synergy analysis," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3738–3745.

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

[23] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.

[24] M. G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi, "Adaptive synergies for the design and control of the pisa/iit softhand," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 768–782, 2014.

[25] F. Ficuciello, G. Palli, C. Melchiorri, and B. Siciliano, "Postural synergies of the ub hand iv for human-like grasping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 515–527, 2014.

[26] M. Bianchi, P. Salaris, and A. Bicchi, "Synergy-based hand pose sensing: Optimal glove design," *The International Journal of Robotics Research*, vol. 32, no. 4, pp. 407–424, 2013.

[27] M. Gabiccini, A. Bicchi, D. Prattichizzo, and M. Malvezzi, "On the role of hand synergies in the optimal choice of grasping forces," *Autonomous Robots*, vol. 31, no. 2-3, p. 235, 2011.