Position-Based Control of Under-Constrained Haptics: A System for the Dexmo Glove

Sebastian Friston[1], Elias Griffith[2], David Swapp[1], Alan Marshall[2] and Anthony Steed[1]

***Abstract*—The Dexmo glove is a haptic exoskeleton that provides kinesthetic feedback in Virtual Reality. Unlike many other gloves based on string-pulleys, the Dexmo uses a free-hinged link-bar to transfer forces from a crank to the fingertips. It also uses an admittance-based controller parameterised by position, as opposed to an impedance-based controller parameterised by force. When setting the controller's target position, developers must use its native angular coordinate system. The Dexmo has a number of uninstrumented degrees-of-freedom. Mature forward models can reliably predict the hand pose, even with these unknowns. When it comes to computing angular controller parameters from a target pose in Cartesian space however, things become more difficult. Complex models that provide attractive visuals from a small number of sensors can be non-trivial or even impossible to invert. In this paper we suggest side-stepping this issue. We sample the forward model in order to build a lookup table. This is embedded in 3D space as a curve, on which traditional queries against world geometry can be performed. Controller parameters are stored as attributes of the sample points. To compute the driver parameters for a target position, the application constrains the position to the geometry, and interpolates them. This technique is generalisable, stable, simple, and fast. We validate our approach by implementing it in Unity 2017.3 and integrating it with a Dexmo glove.**

I. Introduction

HAPTIC feedback extends the immersion of Virtual Environments (VEs) to new sensory modalities. Tactile feedback reproduces surface texture, while force feedback reproduces strain [1]. Haptic interface design is challenging due to the degrees-of-freedom and range-of-motion that must be supported. These lead to conflicting requirements: devices must be stiff but also lightweight; able to apply large forces on demand, but otherwise offer no impedance [2]. Tool-based devices, such as the Phantom Omni, provide high fidelity interaction, but only where the virtual form closely matches the real. Hand exoskeletons and similar devices have received continued interest as they can support more complex arrangements of forces.

One such device is the Dexmo, a commerical product [3] based on the passive glove presented by Gu et al. [4]. Unlike many other devices, the Dexmo uses admittance control, which renders positions, rather than impedance control, which renders forces. Users set the desired pose, and a local control loop attempts to drive it.

While well known in teleoperation, admittance control of haptic gloves is less common than impedance control. Some admittance gloves use angular coordinate systems, with target finger positions expressed as angular distances from the palm [5]. This approximates finger flex and can be passed to the driver as a target between the maximum and minimum extents of the user's pose. Expressing virtual interactions in the palm-fixed angular coordinate system could be challenging for VE designers however. Further, such systems could not take advantage of the latest hand pose estimation techniques.

In this paper we present a position-based approach for under-instrumented/under-actuated devices that can determine position parameters for an admittance controller without needing to directly compute the inverse function of the robot. We follow data-based approaches to inverse kinematics in computer graphics [6] [7], and implement the function between world-position and driver parameters as a lookup table embedded in 3D space. We sample the forward model for different controller parameters and store the results as vertices in 3D space. To determine parameters for a target pose, the barycentric coordinates on the primitives relating these vertices are computed, and the parameters simply read from the geometry. This approach supports complex hand models with multiple uninstrumented linkages. It supports devices with asymmetric motor-sensor degrees-of-freedom. It is simple to understand, and offers more deterministic performance than iterative approaches used in traditional inverse-kinematics.

Lately, a number of underinstrumented & underactuated haptic devices have been presented. These trade-off fidelity and dynamic range for portability and ease-of-use, in an effort to aid adoption and accessibility. This goal is furthered by control systems accessible to VE developers without deep robotics expertise. Our contribution is showing how embedded lookup tables can be used for driving admittance based haptics, and how their ease of implementation & integration with existing engines can simplify VE design. We describe a control system for the Dexmo glove based on this principle, and demonstrate its use with different haptic simulation techniques in an off-the-shelf game engine.

II. Previous Works

Bergamasco et al. [8] provide a good introduction to exoskeletons as haptic interfaces. Exoskeletons provide multiple points of attachment, allowing them to generate more complex

Manuscript received: Feburary, 22, 2019; Revised May, 24, 2019; Accepted June, 28, 2019.

This paper was recommended for publication by Editor Allison Okamura upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by EPSRC Grant EP/P004040/1.

[1]Department of Computer Science, University College London sebastian.friston@ucl.ac.uk, d.swapp@ucl.ac.uk, a.steed@ucl.ac.uk

[2]Department of Electrical Engineering and Electronics, University of Liverpool e.griffith@liverpool.ac.uk, alan.marshall@liverpool.ac.uk

Digital Object Identifier (DOI): see top of this page.

arrangements of forces than tool-based devices. A further benefit for Virtual Reality (VR) is that exoskeletons can be body-fixed, instead of ground-fixed, resulting in potentially larger working volumes.

Foumashi et al. [9] survey 28 robotic hand exoskeletons up to 2011. They included haptic interfaces and rehabilitation robots. Even so, there are common characteristics. For example, almost all exoskeletons were dorsal, used ligaments between the finger joints, and used actuators allowing them to actively push and pull. More recent devices have continued this trend. For example, Burton et al. [10] introduced an exoskeleton optimised for circular power-grasps. Ma and Ben-Tzvi [11] introduced an exoskeleton for remote-control of a mobile robot. Both designs used antagonistic cables and open pulleys.

The benefit of dorsal exoskeletons is that they don't inherently impede the normal workspace of the hand. Some designers however have explored palmar devices. Endo et al. [12] introduced the HIRO III - a five-fingered ground-fixed robot that behaves as a mirror of the hand. The robot has a limited range compared to hand-fixed robots, but significantly improved fidelity, with 3 Degrees of Freedom (DOF) force feedback at each fingertip. The Wolverine by Choi et al. [13] is a very lightweight admittance device designed to reproduce grasping sensations between three fingers and the thumb.

The Wolverine is one of a few newer devices that optimise for portability over fidelity or dynamic range. The workspace of hand-fixed exoskeletons can still be limited due to the need for powerful actuators, as most popularly seen in the CyberGrasp, for example [14]. In their survey of wearable devices, Pacchierotti et al. [15] consider 23 hand-mounted devices and 23 fingertip-mounted devices. They consider devices wearable only if they are also small, easy to carry, and do not impair the motion of the hand. Some of these achieve portability through the use of backpack mounting of the actuators, but more recently dorsal mounted DC motors and other small servo units have been explored.

The original Dexmo glove [4] was also an admittance device, but dorsal mounted, unlike the Wolverine. Each finger had three linkages, with an electronic brake on the middle one that could lock the relative orientations of the adjoining linkages, and therefore the entire digit. This arrangement allows the exoskeleton to follow the natural trajectory of the digits and be locked in place with only one actuator. The link bar mechanism is simpler than the string-pulley arrangements of previous devices, making the device lower cost and more robust. A later iteration of the device was commericalised by Dexta Robotics. In this version, dorsal mounted motors above the metacarpals drive bars that pull or push normal to the fingernails via slider-crank like arrangements. Iqbal et al.'s [16] HEXOSYS II is perhaps the most similar device to the Dexmo. It also uses a two-part linkage with a single dorsal mounted motor. Springer and Ferrier [5] used the arrangement as well, but with a chain rather than a dorsal motor. Koyama et al. [17] used link-bars, but in an extended arrangement with multiple actuators to support two DOFs for each finger. Alternately, Sarakoglou et al.'s [18] HEXOTRAC uses single actuators but multiple encoders, with DOFs above each joint. This improves tracking and avoids the potential joint misalignment of rigid linkages which can lead to undesirable constraints and internal forces.

Fig. 1. The Dexmo glove, annotated with the joints that are sensed but not actuated (A), sensed and actuated (B) and not sensed or actuated (C) for the Index finger and Thumb. Other fingers have identical arrangements to the Index finger.

Works on active haptic robots commonly use force-position (impedance-based) control systems [19] [1] [20], similar to bilateral force-reflection systems. In this case the haptic device acts as a force display. The simulation computes the forces that should be felt by the user based on the avatar position, and the control loop uses a model of the robot to map these to motor torques. Force computation can be performed with traditional computer graphics principles, such as the god-object algorithm [21]. More exotic implementations have been explored though. For example McNeely et al. [22] used voxel mapping to support 6 DOF haptic feedback, while Popescu et al. [23] used the god-object/HIP algorithm but with hardware acceleration to improve performance.

Carigan and Cleary [24] compare impedance and admittance-based control specifically for haptics. Admittance-based systems sense forces from the user and use these to compute the position of the haptic robot. The robot acts as a position-display. Admittance-based systems have an inner and outer loop, where the outer loop includes the simulation that sets the parameters of the inner loop (the position), and the inner loop consists of a feedback loop between the robot's sensors and drivers in order to achieve this position.

Both schemes have been extensively studied for teleoperation [25] [26] [27]. However, impedance-based control has been most popular for haptic devices until now because it has been cheaper and simpler to design for [20]. This is no longer the case for devices such as the Dexmo. The driver units have coincident DC motors & encoders, making admittance-based controllers for each unit straightforward to design, whereas the under-instrumented linkages make computing an inverse function to map forces to torques more difficult.

As a commerical product, the Dexmo provides a black-box API to its admittance controller (inner loop). The parameters are bend angles, so it is easy to imagine each encoder coupled to its coincident motor with a PID controller. The true implementation is immaterial to the outer loop however. The problem of the outer loop is how to derive parameters in the position-space of the inner loop from the position-space of the simulation, and this is the subject of the current work.

III. Dexmo Haptic Glove

The latest iteration of the Dexmo (Figure 1) is an active device with 21 DOFs, 11 of which are instrumented, and 5 of those are driven. The thumb has three sensed DOFs and the remaining fingers two. The driver units are identical on each finger. They crank a primary bar which pulls or pushes normal to the fingertip via a secondary bar connected to finger cups via two hinge joints.

To control the glove, a simple API is used to set the desired rotation of the primary bar, and the stiffness with which the glove should attempt to drive this pose. The parameter is a normalised value between the maximum and minimum rotations at the extremes of the user's finger flex and extension, measured during a calibration stage.

IV. Pose Estimation

The Dexmo is under-instrumented, in that the distal linkage rotations are unknown. When a hand adopts a circular power grasp however, the fingers follow a predictable trajectory [28]. The angles of the Proximal Interphalangeal (PIP) and Distal Interphalangeal (DIP) joints can be modelled as linear functions of each other and Metacarpal (MCP) flex [29] [30] [31]. Therefore, in the case of the Dexmo and similar devices, the rotation of the primary bar is an approximation MCP flex, or, distance along the power grasp trajectory.

A. Hand Model

The Dexmo SDK provides a graphical hand model. We also define our own mechanical model in Denavit-Hartenberg [32] (DH) Parameters. A number of works model aspects of the human hand. Some characteristics can be modelled reliably, while for others (most commonly the Carpometacarpal (CMC) projections), no models exist and there are only exemplary measurements. We could not find a model of the human hand, parameterised entirely by DH-parameters from the wrist, so we created one based on a number of models presented in the literature [33] [34] [31]. The complete model is provided in the supplementary materials and our code[1].

B. Forward Model

The virtual hand is controlled by directly setting the normalised rotations from the Dexmo API. The thumb CMC has two dedicated sensors, as does the MCP abduction parameters. The MCP, DIP and PIP of all fingers and the thumb (interphalangal) are set from the primary bar rotation.

A nice property of this approach is that the exact dimensions of the user's hand, and even their dynamic range, are decoupled from the virtual hand. Our above method of estimating the unknowns in the hand pose is relatively simple. Hand pose estimation is a well studied problem however (e.g. [35] [36] [30] [37]). For example, the human palm is capable of hollowing [33]. We have included the joints necessary for this in our virtual hand. While we do not currently drive them, it is conceivable that more advanced - perhaps even context aware - models (e.g. [38]) could improve the accuracy of the pose estimation without any additional sensor DOFs.

[1]Our implementation is available in the CASMS Haptics Repository https://bitbucket.org/account/user/casms/projects/HAP

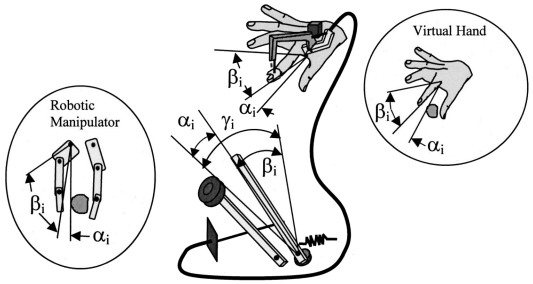

Fig. 2. Springer and Ferrier's position control variables [5]. γ is the virtual object position in the finger's coordinate system.

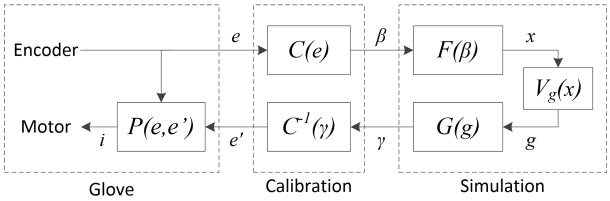

Fig. 3. Dexmo gesture-based outer control loop. g is one of a set of discrete gestures that are translated to admittance parameters by G.

V. Haptic Feedback

The Dexmo's inner loop is proprietary, however its API behaves similarly to the control system proposed by Springer and Ferrier [5]. Like the Dexmo, theirs is parameterised by finger bend angle (β). Their system is based around a *contact drum* shown in Figure 2. As described in Section IV, the finger will follow a predictable trajectory that can be expressed with a single DOF (β). A contact distance from the finger to the virtual object (α) is defined in the same coordinate system, and the drum position which inhibits user motion is set to the virtual object position (γ). In Springer and Ferrier's system, force is defined as a function of torque and implemented via PWM. The force is set per object and applied based on whether the contact distance α is greater or less than zero. This is simlar to the Dexmo, which applies a constant torque when the target position is exceeded: $F \approx a(\beta < \gamma)$, where a is a unitless stiffness parameter provided to the API from the object's material. The Dexmo also allows the comparison to be inverted to make the constraint dorsal.

This parameterisation requires that the geometry be defined in the angular coordinate system of the palm. The control system then for the Dexmo is shown in Figure 3. Encoder measurements (e) are converted (C) to bend angles (β) by the Dexmo SDK. These drive a forward model (F) that computes the hand pose in the VE. For gesture-based interaction, or where geometry is palm fixed (e.g. [23]), the simulation (V_g) needs only to determine a discrete gesture (g) that maps (G) to a target parameter (γ) for the admittance controller (P) (via any internal transforms, e.g. C^{-1}). The Dexmo SDK provides a number of gesture-based examples. However it may not always be easy or possible to define an environment function that computes bend angles.

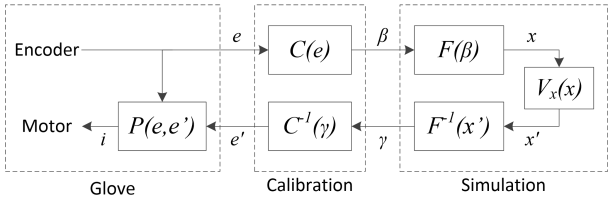

Fig. 4. Ideal position-based outer control loop. x' is a target position for a fingertip that is converted to a bend angle γ for the Dexmo API by F^{-1}.

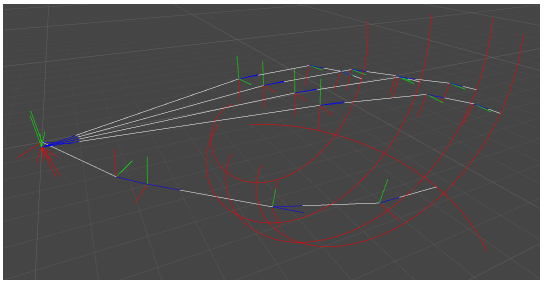

Fig. 5. The sampled curves for all five fingers of our hand model.

VI. Position-based Control

For VE design, it would be preferable to use traditional computer graphics techniques to perform collision detection, and rely on the system to compute γ from a target pose (Figure 4).

This however requires F^{-1}: the transform of a position in simulation space (x') to admittance controller space (γ). Recall from Section IV that one of the benefits of indirectly driving the virtual hand, is that the pose estimation model can be arbitrarily complex. This is more visually attractive. However it could make solving F^{-1} analytically difficult, or even impossible. An alternative would be to create a constrained IK system that models the linkages of the hand and robot. Another would be to turn it into an optimisation problem and use a numerical solver. The constrainted nature of our problem however facilitates a far simpler solution - a look-up table.

A. Sampling the workspace

In a pre-processing step, we sample the forward model F with different driver parameters (β). We store the resulting fingertip positions (x) in 3D space. For each vertex, an attribute contains the driver parameters that achieved that position. The vertices are connected to form a set of lines and thus a curve. Now in order to determine the parameter γ required to achieve a target position x', we simply have to lookup the barycentric coordinates of that position on the curve, use them to interpolate the driver parameters. Figure 5 shows these sampled trajectories for the five fingers of our DH hand model. Each finger has 128 samples and linearly interpolates between them.

B. Asymmetry

The Dexmo has an asymmetric number of sensor-motor DOFs. While the metacarpals are instrumented and actuated, the finger spread is only sensed. There is no point in sampling sensor-only DOFs in the lookup table, as the glove will be unable to effect any change for these. Instead, we transform the geometry based on the pose of these sensor-only DOFs. The geometry then represents at any given time the potential workspace of the drivers in Cartesian space and implicitly, we only try to drive parameters that will actually affect the hand pose. The same technique of transforming the workspace of each actuator allows us to incorporate not only additional degrees of freedom, but other arbitrary transforms. For example, to convert a rig from a left hand to a right hand, all we do is apply a scale of -1 in the x-axis.

Vertex positions encode the real-world range of motion, so the most complex embedding would be a tetrahedral mesh (3 DOF). Driver parameters are stored as vertex attributes, for which any number are supported.

This approach allows us to use an arbitrarily complex hand model and controller. The multiple degrees of freedom and their relationships - real or assumed - are encoded in the fingertip positions. The approach allows the same model used for visual feedback to be used to compute haptic feedback, maximising sensory coherence, and works equally well whether the model is real-world scale or not. It is not clear however what would happen if free and driven joints were interleaved in the same kinematic chain.

VII. Validation

We developed our system out of necessity and so did not have an alternative with which to compare it. Instead to assess its practicality we built it into haptic VEs utilising a variety of approaches towards haptic rendering and simulation.

A. Goal Computation

With an implementation of F^{-1} we need to modify the simulation function V to compute a target position. We experimented with two ways to do this.

1) God Object Mode: The fingertip from the forward model is considered a haptic proxy and is provided with a god-object. A closest-point query between the god-object and the curve identifies the target position.

2) Intersection Point Mode: The curve is tested directly against the world geometry. The intersection point is considered the target position.

We find intersection point mode to be the best approach. This is because the god-object has more degrees-of-freedom than the curve and its deriative in curve-space can vary with time, relative position & world geometry. This makes the mode less well conditioned than the direct intersection approach, which depends only on the hand pose and curve shape, and so is more dynamically stable. The effect is confounded by latency, as the longer between outer loop updates the larger the jump in parameter can be, resulting in oscillations.

B. Simulation

As an admittance-based device, the simulation must be driven by the hand pose. We experimented with two ways to do this.

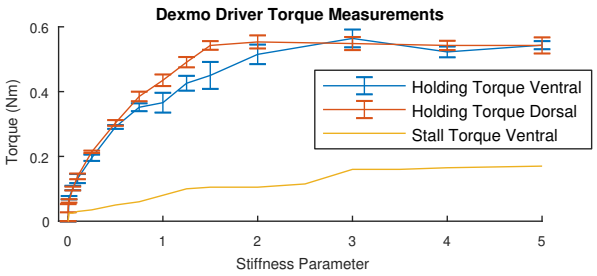

Fig. 6. Driver controller Characteristic for a range of Stiffnesses

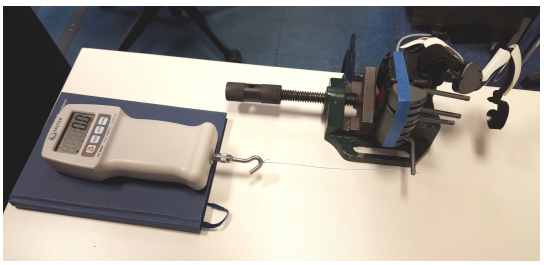

Fig. 7. Measuring the torque of a Dexmo finger driver. The attachment point is 50 mm from the pivot.

1) Force Reflection: To apply forces back to the simulation we estimate the forces applied by the user's fingers. The Dexmo does not implement Hooke's Law but rather applies a constant torque according to a parameter a, until the actuator returns to its target pose (Section V). To map between a and real-world force, we sampled the stiffness of the controller with a force gauge (Figures 6 & 7). Holding Torque is the maximum torque necessary to backdrive the actuator, while the Stall Torque is the torque driven at zero speed. We fit a function (Equation 1) to stiffness samples between $0-1.5$ to describe the true torque ($R^2 = 0.99, RMSE = 0.021$). We use the Holding Torque as this is the largest torque the user can exert.

$$\tau(a) = 0.2016a^3 - 0.614a^2 + 0.802a - 0.04523 \qquad (1)$$

Computing force from the torque requires the power-grasp radius. This can be determined as part of the pose estimation, as the distance between the MCPs joint and the fingerip. This is only accurate so far as the hand model represents the dimensions of the user's hand however. Establishing those dimensions is not part of the calibration. We accept some error in exchange for a simpler calibration procedure, but the error is bounded by the range of human hand dimensions.

2) Collision-based Simulation: The problem with force-reflection is that underinstrumented & underactuated devices limit grasp isotrophy (the ability for finger joints to accurately apply forces and torques) [39]. This makes stable grasping through force and torque closure difficult. Compounding this is the lack of friction as forces are applied from infinitesimally small points. Friction forms an important component of force closure. Techniques such as the Friction Cone Algorithm are almost always necessary, as without friction at least four points of contact are required for stable grasping [40].

An alternative to force-reflection is to rely on geometric collision constraints. Haptic feedback is provided via inter-

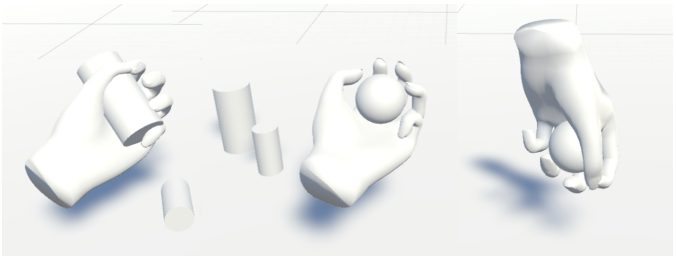

Fig. 8. Grasps of handheld objects using geometric colliders

section tests, which in theory constrain the hand pose to the bounds of the virtual object as in Section VII-A2. Additionally though, a geometric model of the hand participates in the physical simulation. The virtual objects' behaviour are driven by collisions with hand model. This results in more stable grasps with higher manipulability, as the hand geometry provides the user with additional degrees of freedom and larger numbers of contact points.

The drawback is that because the device is underactuated, simulated contacts cannot be accurately rendered to the user. It is easy then for the user to severely violate collision constraints provoking high energy responses. The signifiance of these will depend on the simulation. PhysX is an impulse-based simulation that is vulnerable to large forces, whereas the unconditionally stable Position-Based Dynamics approach would be more forgiving [41]. A second drawback is low fidelity force rendering. Simulated impulses are not reflected back into the glove. Therefore only when an object is over-constrained will the users recieve significant force feedback; otherwise, collisions will be resolved completley within a single frame, and the haptic feedback will depend mostly on the framerate (the maximum penetration per frame).

Figure 8 shows the geometric hand from the Dexmo SDK grasping a number of simple objects (the PhysX engine requires dynamic bodies be convex). All grasps rely on geometric collisions to a different degree. Overhand grasps would be possible with force-reflection only, but not with the power grasp assumption of the Dexmo, while others (e.g. lifting from below) are only possible with geometric collisions.

C. Force Rendering

We can invert the samples taken in Section VII-B1 and fit another function (Equation 2) in order to compute stiffness values with which to drive specific torques/forces to the user ($R^2 = 0.95, RMSE = 0.437$). In this case we use the Stall Torque as this is the maximum that can be actively exerted by the device. As described in Section VII-B1, the torque is proportional only to the Stiffness parameter. To render force we compute Stiffness given a target torque, and simply set the position to be above or below the current pose to provoke its actuation.

$$a(\tau) = -203.3\tau^3 + 163.8\tau^2 + 1.826\tau - 0.03613 \qquad (2)$$

Figure 9 shows the fidelity of the Dexmo force rendering based on Equation 2 across its dynamic range. Figure 10 shows

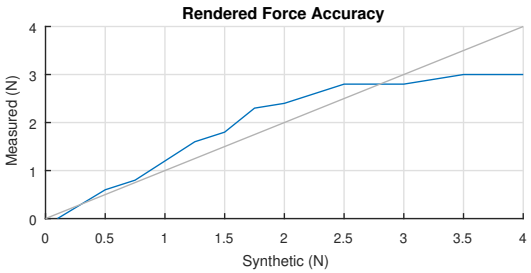

Fig. 9. Actual force exerted by Dexmo for intended forces provided to the controller

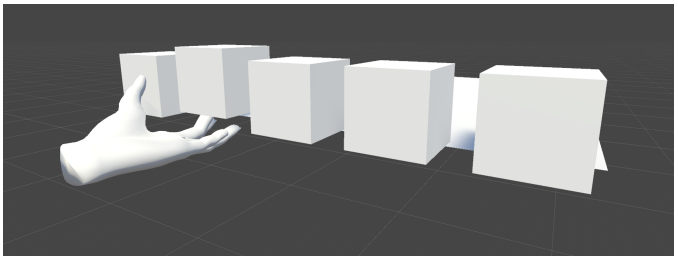

Fig. 10. Virtual environment with synthetic weights used to drive Dexmo in force rendering mode

a VE with a set of virtual objects used to demonstrate force rendering in VR.

PhysX is an impulse-based simulation engine. Therefore the force for each finger was computed from the integral of the impulses applied to the finger's colliders by the simulation, divided by the period of the frame. In this case, the force was applied directly, as this best matched the geometric arrangement of our VE. However, as the Dexmo is underactuated, this is unlikely to be a good solution for all VEs. Sarac et al. [42] proposed a number of rendering strategies that could be applied to differnt VEs. Ultimatley though, we advise against the use of force-reflection.

It is well known from teleoperation that in latent force-reflection systems, user input in the form of position, and system response in the form of force, can become out of phase [43] [44] [45]. In this case the user experience is undermined and the system can become unstable, which we experienced during our tests. The typical haptic loop frequency is 1000 Hz. Though our technique can approach this (Section IX), latency is limited by the slowest component. In this case, the Unity framerate that we measured at ∼100 Hz, and the Dexmo update rate we constrained to 30 Hz on the advice of Dexta Robotics (personal communication).

Due to the oscillations caused by the low bandwidth, we could not measure constant forces exerted by virtual weights, however we were able to measure the maximum forces at the oscillation peaks. These are shown in Figure 11. Such measures will have some inertial component, however as can be seen they are minimal and the response is consistent with the expected behaviour of the controller, based on its slightly above unity gain shown in Figure 9.

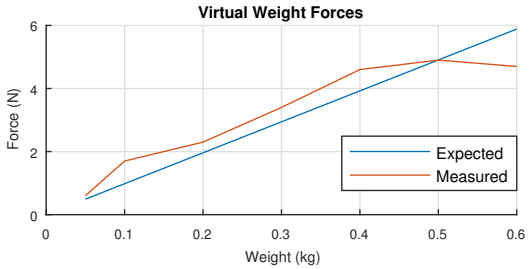

Fig. 11. Maximum forces exerted for virtual weights

VIII. Implementation

We implemented our system in Unity 2017.3 with Dexmo's managed client library.

A. Hand Model

Both the Dexmo graphical model and our DH model were implemented as hierarchies of scene-graph nodes. A function would take the bend parameters and apply proportional rotations to the transforms as described in Section IV. In the Dexmo API sensor and driver parameters are in the same space - normalised between the extremes of user flex and extension.

B. Lookup Table

We identified which nodes belonged to kinematic chains downstream of the actuators, and therefore should be sampled in the lookup tables. A function set the model parameters and measured the endpoint of a chain, representing the tip of the corresponding finger.

Whenever the hand model changes, we construct for each finger a set of vertices/lines as described in Section VI. We use the Unity messaging system to repeatedly update all transforms within a single function call allowing us to do this in one instant at design time. For the purposes of the lookup construction, F is a function call that sets the normalised rotation parameters of the hand controller script, updates all the transforms, and reads back the fingertip positions in Unity world coordinates. The curve is stored in the local space of first actuated node's parent. For the Dexmo, this is the finger spread, or, MCP abduction, joint.

C. Collision Detection

For collision detection, we iterate over each line forming the sampled curve, testing it against potential colliders with a raycast. The ray is from the line's origin and the ray-intersection distance determines whether or not the segment intersects. Unity provides raycast tests for all colliders. The first intersection that lies within a segment is considered to be the target point. If the tests reach the end of the curve, there is considered to be no intersection. When there is no intersection, the stiffness of the finger controller is set to zero. Potential colliders are identified with a broadphase pass. Curves are encompassed with bounding boxes, and we rely on Unity to identify potentially intersecting colliders that are then tested as above.

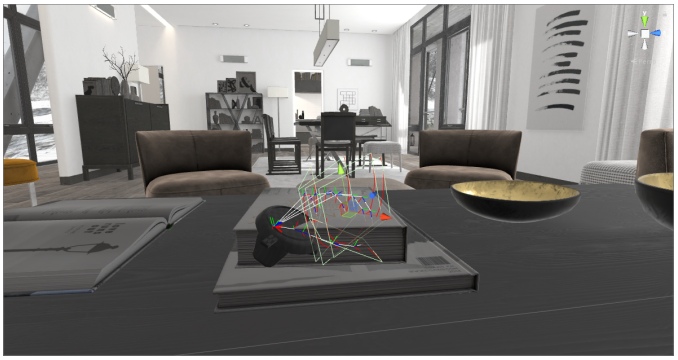

Fig. 12. Virtual hand in a haptically-enabled architectural visualisation virtual environment grasping a book, with visualisations of the hand DH-model, finger trajectories and bounding boxes for the collision detection broadphase

IX. Performance

Our problem is grounded in the real-world biophysics of the hand, which gives an upper bound on complexity. For example, with a finger length of 10 cm and an MCP range of 130°, 128 sample points would achieve a sampling spatial resolution of approximately 1.7 mm (and recall that the parameters are interpolated between these). This allows us to take a brute-force approach more often than would otherwise be practical. Characterising the performance of our technique is difficult. Individual raycasts are typically fast, but the number of potential colliders has a multiplicative effect on the processing time. This makes our approach highly dependent on the scene and 3rd party physics engine.

We validated our approach by integrating the Dexmo into a VR architectural visualisation environment (Figure 12). Haptic feedback was enabled for the entire environment and its contents by creating static Mesh Colliders from the scene's graphical geometry. The scene contained 898,675 triangles across 668 colliders. Updating the haptic parameters for all five fingers took approximately 1.19 ms per frame, when grasping as shown in Figure 12. Of this 0.31 ms was spent performing the actual raycasts. PhysX, which Unity uses as its physics engine, spent 0.01 ms in the broad phase.

It is important to remember that what we describe constitutes the outer loop of the control system. Therefore these execution times will not affect stiffness. They will however determine the highest spatial frequencies that can be displayed as they limit the change in stiffness over time.

X. Conclusion

The Dexmo is a haptic glove designed to provide kinesthetic feedback. Like a number of recent designs, it has been optimised for portability and ease-of-use. One way this has been done is to use link-bars instead of more delicate and expensive pulley-string systems. Such arrangements however introduce challenges in pose estimation and control system design. The Dexmo uses an admittance-based controller, where an inner loop drives a target pose set by an outer loop incorporating the simulation. The inner loop can operate at high rates, but requires the outer loop to specify positions in an angular coordinate system, rather than forces in a Cartesian system.

The problem of estimating angular position is underconstrained. Models with more DOFs than the glove can sense must be constructed, and assumptions made about the relationships between the glove parameters and hand model in order to estimate hand pose. While this is doable for pose estimation, inverting it to compute driver parameters is non-trivial. Traditionally, abstractions-based on palm-fixed geometry and abstract measures such as curvature would be used. However this complicates the design of the VE.

We propose a new way to compute the target pose for such underconstrained haptic devices. Our approach is based around sampling the forward model to build a lookup table. The lookup table is queried by expressing it as geometry in the Cartesian space of the simulation. Here it can be transformed to express the workspace of the actuators at any time based on additional sensed or assumed properties. Basic collision detection and geometric queries can be performed on the geometry to solve for a target pose, and the required actuator parameters to achieve it. The actuator parameters can then be fed to the inner-loop.

While we have only tested it on the Dexmo, the technique should generalise to similar devices, including those that can actuate in multiple embedded dimensions. Further, while the Dexmo SDK uses the same parameter space for its controller as it does to report finger pose, this is not a requirement of our technique. To compute the actual pose for use by this inverse function we tested two algorithms. We find the open-loop intersection point to be superior to the god-object, as it is the most stable. However, both algorithms were straightforward. Our embedded trajectory describes to the application the potential workspace of the actuator. In the future it, is worth exploring algorithms that can better take advantage of this to provide the optimal kinesthetic cues given these constraints. For controlling the simulation, we experimented with force-reflection and collision constraints. We found collision constraints to be superior because they provided additional DOFs with which to establish contact points. This improved the manipubility of the simulation beyond what is possible with force-reflection, due to the limited grasp isotrophy of the underinstrumented device.

New haptic devices are being optimised for portability and ease-of-use in an effort to aid adoption and accessibility. To achieve this goal VE developers need equally accessible control systems with which to drive them. Our experimental VEs validate the embedded 3D lookup table for controlling admittance-based haptics, by demonstrating its practicality in real VEs. We hope that this method can contribute to the continued development and adoption of accessible, low cost haptics.

References

[1] G.C. Burdea. Haptics issues in virtual environments. *International Conference on Computer Graphics*, pages 295–302, 2000.

[2] S. D. Laycock and A. M. Day. Recent developments and applications of haptic devices. *Computer Graphics Forum*, 22(2):117–132, 2003.

[3] DextaRobotics. https://www.dextarobotics.com, 2019. [Online; accessed 08-May-2019].

[4] Xiaochi Gu, Yifei Zhang, Weize Sun, Yuanzhe Bian, Dao Zhou, and Per Ola Kristensson. Dexmo: An Inexpensive and Lightweight Mechanical Exoskeleton for Motion Capture and Force Feedback in

VR. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, pages 1991–1995, 2016.

[5] Scott L. Springer and Nicola J. Ferrier. Design and Control of a Force-Reflecting Haptic Interface for Teleoperational Grasping. *Journal of Mechanical Design*, 124(2):277, 2002.

[6] Charles F Rose, Peter-pike J Sloan, and Michael F Cohen. Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation. *Computer Graphics Forum*, 20(3):239–250, 2001.

[7] Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522, 2004.

[8] Massimo Bergamasco, Antonio Frisoli, and Carlo Alberto Avizzano. Exoskeletons as Man-Machine Interface Systems for Teleoperation and Interaction in Virtual Environments. In *Advances in Telerobotics*, number STAR 31, page 6176. Springer, Berlin, Heidelberg, 2007.

[9] Mohammad Mozaffari Foumashi, Marco Troncossi, and Vincenzo Parenti Castelli. State-of-the-Art of Hand Exoskeleton Systems. Technical report, 2011.

[10] T. M.W. Burton, R. Vaidyanathan, S. C. Burgess, A. J. Turton, and C. Melhuish. Development of a parametric kinematic model of the human hand and a novel robotic exoskeleton. *IEEE International Conference on Rehabilitation Robotics*, 2011.

[11] Zhou Ma and Pinhas Ben-Tzvi. RML glove-an exoskeleton glove mechanism with haptics feedback. *IEEE/ASME Transactions on Mechatronics*, 20(2):641–652, 2015.

[12] Takahiro Endo, Haruhisa Kawasaki, Tetsuya Mouri, Yasuhiko Ishigure, Hisayuki Shimomura, Masato Matsumura, and Kazumi Koketsu. Five-fingered haptic interface robot: HIRO III. *IEEE Transactions on Haptics*, 4(1):14–27, 2011.

[13] Inrak Choi, Elliot W. Hawkes, David L. Christensen, Christopher J. Ploch, and Sean Follmer. Wolverine: A wearable haptic interface for grasping in virtual reality. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:986–993, 2016.

[14] Manuel Aiple and Andre Schiele. Pushing the limits of the CyberGrasp for haptic rendering. In *2013 IEEE International Conference on Robotics and Automation*, pages 3541–3546. IEEE, 5 2013.

[15] Claudio Pacchierotti, Stephen Sinclair, Massimiliano Solazzi, Antonio Frisoli, Vincent Hayward, and Domenico Prattichizzo. Wearable Haptic Systems for the Fingertip and the Hand: Taxonomy, Review, and Perspectives. *IEEE Transactions on Haptics*, 10(4):1–1, 2017.

[16] Jamshed Iqbal, Omar Ahmad, and Ahsan Malik. HEXOSYS II - Towards realization of light mass robotics for the hand. *Proceedings of the 14th IEEE International Multitopic Conference 2011, INMIC 2011*, pages 115–119, 2011.

[17] Tatsuya Koyama, Ikuo Yamano, Kenjiro Takemura, and Takashi Maeno. Development of Multi-Fingered Exoskeleton Haptic Device using Passive Force Feedback. *IEEE International Conference on Intelligent Robots and Systems*, 7(4):2905–2910, 2002.

[18] Ioannis Sarakoglou, Anais Brygo, Dario Mazzanti, Nadia Garcia Hernandez, Darwin G. Caldwell, and Nikos G. Tsagarakis. Hexotrac: A highly under-actuated hand Exoskeleton for finger tracking and force feedback. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:1033–1040, 2016.

[19] Mandayam A. Srinivasan and Cagatay Basdogan. Haptics in virtual environments: Taxonomy, research status, and challenges. *Computers & Graphics*, 21(4):393–404, 1997.

[20] Kenneth Salisbury, Francois Conti, and Federico Barbagli. Haptic rendering: Introductory concepts. *IEEE Computer Graphics and Applications*, 24(2):24–32, 2004.

[21] C.B. Zilles and J.K. Salisbury. A constraint-based god-object method for haptic display. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 146–151. IEEE Comput. Soc. Press, 1995.

[22] William A. McNeely, Kevin D. Puterbaugh, and James J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*, pages 401–408, 1999.

[23] V. Popescu, G. Burdea, and M. Bouzit. Virtual reality simulation modeling for a haptic glove. *Proceedings Computer Animation 1999*, pages 195–200, 1999.

[24] Cr Carignan and Kr Cleary. Closed-loop force control for haptic simulation of virtual environments. *Haptics-e*, 1(2):1–14, 2000.

[25] Keyvan Hashtrudi-zaad and Septimiu E Salcudean. Analysis of Control Architectures for Teleoperation Systems with Impedance / Admittance Master and. *The International Journal of Robotics Research*, 20(6):419–445, 1999.

[26] I. Aliaga, A. Rubio, and E. Sanchez. Experimental Quantitative Comparison of Different Control Architectures for Master-Slave Teleoperation. *IEEE Transactions on Control Systems Technology*, 12(1):2–11, 2004.

[27] Erick J. Rodriguez-Seda, Dongjun Lee, and Mark W. Spong. Experimental Comparison Study of Control Architectures for Bilateral Teleoperators. In *IEEE Transactions on Robotics*, volume 25, pages 1304–1318, 12 2009.

[28] Salvador Cobos, Manuel Ferre, and Rafael Aracil. Simplified human hand models based on grasping analysis. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 610–615, 2010.

[29] Zong Ming Li and Jie Tang. Coordination of thumb joints during opposition. *Journal of Biomechanics*, 40(3):502–510, 2007.

[30] Christopher-Eyk Hrabia, Katrin Wolf, and Mathias Wilhelm. Whole Hand Modeling using 8 Wearable Sensors: Biomechanics for Hand Pose Prediction. *Proceedings of the 4th Augmented Human International Conference on - AH '13*, pages 21–28, 2013.

[31] Salvador Cobos, Manuel Ferre, M.A. Sanchez Uran, Javier Ortego, and C. Pena. Efficient human hand kinematics for manipulation tasks. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2246–2251. IEEE, 9 2008.

[32] Jacques Denavit and Richard Scheunemann Hartenberg. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Transactions ASME Journal of Applied Mechanics*, 23:215221, 1955.

[33] Frank P.J. Van Der Hulst, Simon Schätzle, Carsten Preusche, and Andr Schiele. A functional anatomy based kinematic human hand model with simple size adaptation. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5123–5129, 2012.

[34] Bryan Buchholz and Thomas J. Armstrong. A kinematic model of the human hand to evaluate its prehensile capabilities. *Journal of Biomechanics*, 25(2):149–162, 1992.

[35] Matteo Bianchi, Paolo Salaris, and Antonio Bicchi. Synergy-based hand pose sensing: Optimal glove design. *International Journal of Robotics Research*, 32(4):407–424, 2013.

[36] Matteo Bianchi, Paolo Salaris, and Antonio Bicchi. Synergy-based hand pose sensing: Reconstruction enhancement. *International Journal of Robotics Research*, 32(4):396–406, 2013.

[37] P. Cerveri, E. De Momi, N. Lopomo, G. Baud-Bovy, R. M.L. Barros, and G. Ferrigno. Finger kinematic modeling and real-time hand motion estimation. *Annals of Biomedical Engineering*, 35(11):1989–2002, 2007.

[38] Nasser Rezzoug and Philippe Gorce. Prediction of fingers posture using artificial neural networks. *Journal of Biomechanics*, 41(12):2743–2749, 2008.

[39] Mark R. Cutkosky. On Grasp Choice, Grasp Models, and the Design of Hands for Manufacturing Tasks. *IEEE Transactions on Robotics and Automation*, 5(3):269–279, 1989.

[40] Ignacio Galiana and Manuel Ferre. *Multi-finger Haptic Interaction*. Springer Series on Touch and Haptic Systems. Springer London, London, 2013.

[41] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 4 2007.

[42] Mine Sarac, Massimiliano Solazzi, Miguel A Otaduy, and Antonio Frisoli. Rendering Strategies for Underactuated Hand Exoskeletons. *IEEE Robotics and Automation Letters*, 3(3):2087–2092, 7 2018.

[43] B. D O Anderson. The small-gain theorem, the passivity theorem and their equivalence. *Journal of the Franklin Institute*, 293(2):105–115, 1972.

[44] Robert J. Anderson and Mark W. Spong. Bilateral Control of Teleoperators with Time Delay. *IEEE Transactions on Automatic Control*, 34(5):494–501, 1989.

[45] Gunter Niemeyer and J.-J.E. Slotine. Stable adaptive teleoperation. *IEEE Journal of Oceanic Engineering*, 16(1):152–162, 1 1991.