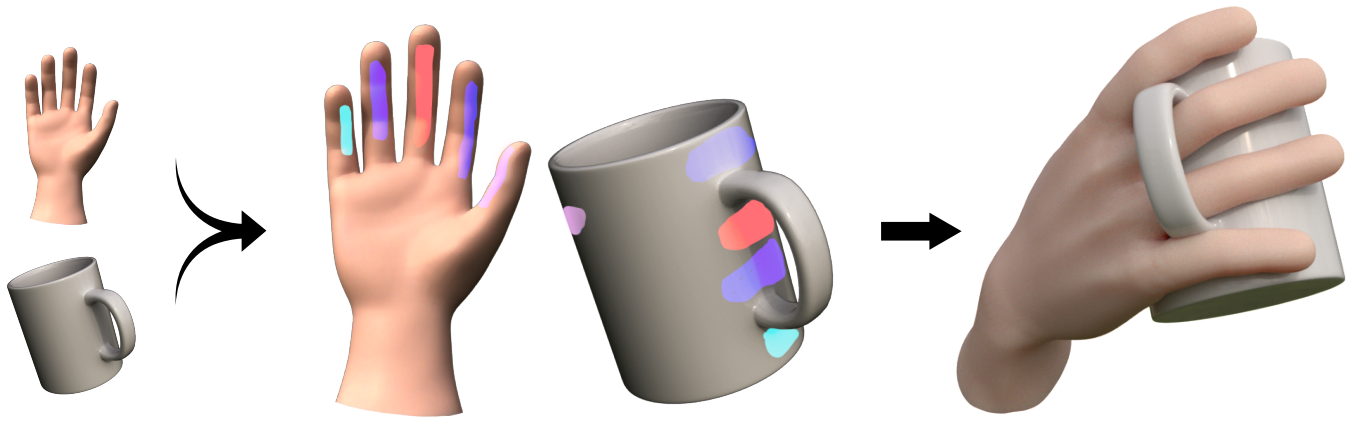
Contact Edit: Artist Tools for Intuitive Modeling of Hand-Object Interactions

ARJUN S. LAKSHMIPATHY, NICOLE FENG, YU XI LEE, MOSHE MAHLER, and NANCY S. POLLARD
Carnegie Mellon University, USA



Posing high-contact interactions is challenging and time-consuming, with hand-object interactions being especially difficult due to the large number of degrees of freedom (DOF) of the hand and the fact that humans are experts at judging hand poses. This paper addresses this challenge by elevating contact areas to first-class primitives. We provide *end-to-end art-directable* (EAD) tools to model interactions based on contact areas, directly manipulate contact areas, and compute corresponding poses automatically. To make these operations intuitive and fast, we present a novel axis-based contact model that supports real-time approximately isometry-preserving operations on triangulated surfaces, permits movement between surfaces, and is both robust and scalable to large areas. We show that use of our contact model facilitates high quality posing even for unconstrained, high-DOF custom rigs intended for traditional keyframe-based animation pipelines. We additionally evaluate our approach with comparisons to prior art, ablation studies, user studies, qualitative assessments, and extensions to full-body interaction.

CCS Concepts: • **Computing methodologies** → **Procedural animation**; **Parametric curve and surface models**.

Additional Key Words and Phrases: hands, contact-driven methods, grasping, posing, inverse kinematics, optimization

Authors' address: Arjun S. Lakshmipathy, aslakshm@andrew.cmu.edu; Nicole Feng, nfeng@andrew.cmu.edu; Yu Xi Lee, yuxilee@andrew.cmu.edu; Moshe Mahler, mmahler@andrew.cmu.edu; Nancy S. Pollard, nsp@cs.cmu.edu, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2023 Copyright held by the owner/author(s).
0730-0301/2023/8-ART1
https://doi.org/10.1145/3592117

ACM Reference Format:
Arjun S. Lakshmipathy, Nicole Feng, Yu Xi Lee, Moshe Mahler, and Nancy S. Pollard. 2023. Contact Edit: Artist Tools for Intuitive Modeling of Hand-Object Interactions. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 20 pages. https://doi.org/10.1145/3592117

1 INTRODUCTION

Posing high contact interactions remains a challenging and time-consuming endeavor, even for seasoned animators. Despite the intrinsic coupling of manipulator and object motion in the physical world, production workflows today largely treat the bodies independently, requiring the animator to pose objects and all of the manipulator joints in a layered, painstaking process utilizing forward and inverse kinematics. While several recent works have endeavored to partially automate the process with exciting results (e.g., [Zhang et al. 2021]), these approaches are rarely adopted in traditional keyframe-based animation pipelines due to their reliance on priors, limited generalization to custom assets, and lack of a built-in feedback loop enabling detailed artistic control over results.

Consequently, manual kinematic posing remains the dominant strategy in production workflows today. This approach enables full artistic control and can work for any interaction; however, changes to the object or hand scale, geometries, kinematic structures, or even basic isometric transformations require complete re-casting of the desired poses, resulting in a laborious and costly process. We argue that the lack of intrinsic pose coupling is ultimately responsible for the majority of these difficulties. To address this challenge, we propose to model this coupling explicitly by providing artist tools to work directly with contact areas.

Contact areas, as the interaction interface between bodies in the real world, present an ideal primitive to model coupling between

hands and objects. Numerous recent works have showcased the benefits of using contacts to drive grasping strategies (e.g., [Brahmbhatt et al. 2019b; Mordatch et al. 2012; Ye and Liu 2012]); however, these techniques are designed more with automation in mind than with the goal of artist control. Without the ability to *edit* arrangements, the value of using contacts as viable primitives rapidly diminishes.

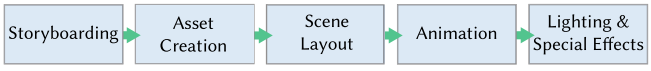
This paper aims to address the aforementioned shortcomings, such that contacts can be integrated into production workflows directly, intuitively, and robustly. Specifically, we:

- propose and demonstrate contact areas as a useful mechanism for specifying high-contact interactions,
- present end-to-end art-directable tools for defining contact areas, directly editing contact areas, and transferring contact areas between surfaces,
- define a novel contact model and foundational algorithms to manipulate and transfer contact areas using the model,
- modify an existing optimization framework to pose deformable, unconstrained, high DOF manipulators,
- will release [on publication] a Maya plugin for use and to facilitate comparisons to our approach.

The techniques proposed in this paper are fully deterministic, not reliant on external priors, computationally inexpensive, robust to high complexity interactions, provide transparent behavior, enable intuitive interfaces, and are designed to facilitate artistic interaction with the animator. We validate these claims by putting our plugin in the hands of real animators to show that high quality results can be generated even with minimal training. Finally, although this paper primarily targets hand-object interaction, we show that the approach generalizes well to other scenes featuring large areas of contact, such as a full-body character sitting down into a chair.

2 PRODUCTION PROCESS OVERVIEW

Modern production workflows for traditional 3D keyframe-based CG media involve a large number of stakeholders but generally follow a standardized pipeline depicted below. More comprehensive overviews are available online[1]:



Each stage is carried out by one or multiple subteams and overseen by an art director, while the entire production is overseen by an executive director. While the pipeline is effective in promoting specialization and is scalable to large organizations, the separation of work does create several important caveats in operations.

Character assets are modeled, rigged, and skinned at the asset creation stage, and are subsequently delivered as final products to the animation stage. Animators do not typically modify these assets, but nonetheless are expected to generate high quality poses and must be able to iterate and explore multiple poses while including input from the art and executive director. The rigging team will create basic and unconstrained control handles to address the need for grasp oriented interaction. These handles are often generic and may not consider where and how the hand will make contact with the object, creating substantial complexity for the animator. Even if a studio provides custom tools to add control handles in an ad-hoc manner, the animator will still need to manipulate the skeletal rig, approximating how the surface meshes of both the object and the hand will interact. The end result is a process which is especially challenging, highly manual, and time-consuming for poses involving multi-asset interaction such as hands and objects. These tasks are particularly daunting for early-career animators, who often end up "fighting the rig" due to their lack of experience.

One potential solution is to enable animators to define interactive poses on meshes directly and instead compute the optimal corresponding rig pose automatically; however, it is *equally* challenging to create robust tools which "just work" on a wide variety of geometries due to fundamental differences in mesh desiderata between the geometry processing and animation communities. For example, the tight coupling between rigging and skinning biases asset modeling towards the construction of coarse meshes of quadrilateral primitives, while many geometry processing algorithms assume, if not outright require, triangle primitives. Although quads can be trivially triangulated, there is no oversight to ensure that these assets have additional desirable properties (e.g. uniform sampling, no triangle inequality violations, etc.). Additionally, in an effort to reduce asset creation time and overall production costs, a growing number of assets today are obtained from alternate sources such as 3D scans or CAD models designed for printing, which have far more degenerate properties than in-house artist models.

Art-directable (AD) tools are expected to operate under such workflows and provide an easy-to-understand interface, produce predictable behavior, enable refinement, be reasonably fast and robust to a wide range of assets, and must be flexible enough to accommodate a wide variety of user and project needs in order to provide value. *End-to-end art-directable (EAD)* tools, which enable animators to create results from scratch entirely under the tool framework, are even more challenging to design; instead, the large majority of tools today are "last-mile" only, requiring the animator to perform a significant amount of upfront work manually in order to utilize them. This paper presents EAD tools for hand-object pose generation through the direct manipulation of contact areas, which are fully compatible with existing posing tools, founded on principled mathematics, easy to understand, and more broadly generalizeable to multi-object interactions.

3 RELATED WORKS

Contact-driven approaches have been employed on numerous occasions in a variety of contexts. We consider three broad categories of related prior work: pose refinement; grasp and manipulation synthesis; and inverse kinematics. We also survey the texture and material transfer literature, which is relevant to our particular method.

3.1 Pose Refinement

One of the most popular uses of contact information is to refine coarse poses, often with the goal of removing unwanted artifacts and improving physical plausibility. For example, contact points have been used to drive physics-based motion synthesis [Liu 2008], refine tracked data into physically plausible solutions [Ye and Liu

[1] https://dreamfarmstudios.com/blog/3d-animation-pipeline/

2012], generate hand motion for full body motion capture through stochastic optimization [Zhao et al. 2013], and improve the quality of hand-hand interactions through compiled databases [Lee et al. 2017]. They have been used in some interactive applications, in one prior work enabling animators to specify contact points to facilitate fusing of independently tracked hand and object motion [Hamer et al. 2011]. All of these approaches use single point contact models.

Area and volumetric representations of contact provide greater solution consistency over larger regions. For example, deformable models have been utilized to correct reconstructed geometries in an effort to resolve self-collisions [Smith et al. 2020]. Regional contact maps have also been inferred [Grady et al. 2021; Jiang et al. 2021] by training on publicly available datasets [Brahmbhatt et al. 2019a, 2020] or in house using motion capture [Taheri et al. 2020] to optimize coarse pose estimates. Like these works, we exploit full contact regions, but address the scenario where ground truth data is not available and may not exist. Instead, we strive to provide the animator with tools to create and manipulate contact regions to create an envisioned result from scratch.

3.2 Grasp Synthesis

Contacts have played a vital role in the generation of grasps and manipulations. Computing kinematically or dynamically feasible whole hand grasps, especially for dexterous manipulators, is a notoriously challenging problem with a long studied history. Historical works typically considered grasps analytically by examining point contact models with Coulomb friction [Murray et al. 1994; Siciliano and Khatib 2007]. Other techniques include performing global searches for kinematically feasible grasps meeting certain constraints [Pollard 1997], leveraging passive compliance to exploit automatic contact dynamics from physics simulation [Pollard and Zordan 2005], and targeting optimal independent regions within which point contacts can be placed [Roa and Suarez 2009]. Contacts have also been used as error terms in the training of generative models [Christen et al. 2021; Wu et al. 2021]. A number of works target manipulation planning and optimization by switching between contact modes [Cheng et al. 2021] or by optimizing contact placement and forces for physically plausible results [Hazard et al. 2020; Mordatch et al. 2012]. The ϵ metric proposed by Ferrari and Canny for evaluating grasp wrench spaces [Ferrari and Canny 1992] – the most widely used grasp quality evaluation metric – is fundamentally rooted in its analysis at individual contact points.

However, characterization of grasps as single points highly simplifies the complexities of real interactions and cannot account for geometric consistency outside the designated point. This is especially problematic for manipulators with deformable skins, which may utilize large contact regions to grasp objects. Solutions capable of generating high quality results through the integration of differentiable contact dynamics have been presented [Turpin et al. 2022]; however, it is challenging to define a custom grasp through this framework or compute solutions in timescales acceptable for interactive iteration. Instead, the precedents of optimizing poses by matching contacts [Brahmbhatt et al. 2019b; Lakshmipathy et al. 2022] are most relevant to our work; however, these methods focus largely on grasp generation and assume that contact areas defining the grasps are available from external sources. This paper provides the critical missing pieces required to create and edit contact areas from the ground up, thus eliminating reliance on external data completely and enabling the creation of EAD tools. We also introduce a simple, but necessary modification to the optimization formulation presented in previous work to enable our tools to compute solutions for unconstrained high DOF rigs.

3.3 Inverse Kinematics

Posing characters to meet contact targets has long been addressed through inverse kinematics (IK). Zhao and Badler were the first to describe inverse kinematics as a general optimization problem [Zhao and Badler 1994], pointing out the flexibility of formulating optimal pose with an objective function rather than following a specific iterative IK process. This theme has been carried forward in follow-up works which allowed artists to work with many IK handles stably and robustly to pose characters [Yamane and Nakamura 2003]. A relatively recent survey of IK approaches can be found in the work of Harish et. al. [Harish et al. 2016]. Our optimization can be viewed as a solution to the IK problem when the IK targets are contact areas. We present a modification of the optimization approach in previous works [Lakshmipathy et al. 2022] that works reliably and robustly to solve this problem.

3.4 Texture and Material Transfer

Our method of contact parameterization and transfer based on logarithmic maps (i.e. geodesic distance and direction) is similar to methods previously used in texture transfer and mesh editing. Like our method, previous works use the log map to parameterize surface patches with respect to a “spine" and allow user control over the transfer process by specifying a starting point and direction for the spine on the target mesh [Biermann et al. 2002]. However, the existing technique [Biermann et al. 2002] requires both the source and the target mesh to be parameterized over the plane, which renders the method complicated and costly. In contrast, our method performs no intermediate parameterizations: contact patches are instead parameterized directly via surface-computed log maps and patches are transferred by tracing geodesics on the target surface. The log map parameterization has also been used to transfer texture “decals" [Schmidt et al. 2006]; however, the computed log maps are approximations and exhibit accuracy degeneration that depends on the sampling of the underlying surface. This technique [Schmidt et al. 2006] also experiences texture “foldover” when the underlying sampling is too sparse, where discretization artifacts cause the texture fold over itself. We instead compute log maps which are always exact in the polyhedral sense using the MMP algorithm [Mitchell et al. 1987] while still maintaining interactive rates. Additional control is provided by allowing users to parameterize contact patches with respect to an axis of choice, enabling artifacts to be avoided even for difficult transfers. In summary, our approach contributes interactive texture transfer and editing, with log maps that are always exact in the polyhedral sense, and that avoids the need for an intermediate mesh parameterization as well as the high patch distortions and foldover that can come from prior approaches. We

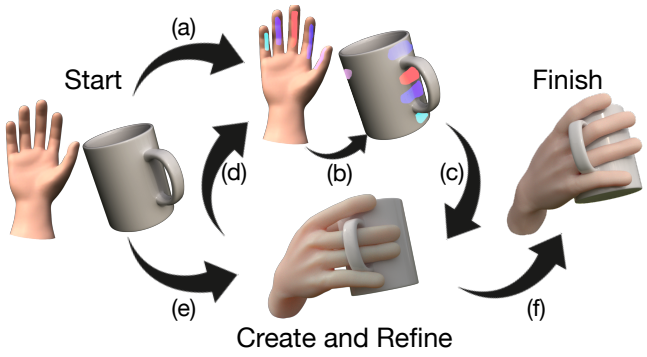


Fig. 1. High level overview of our tool workflow. Starting from a given hand and object, the artist typically starts by (a) creating contact areas on the hand and manipulating them into their final positions before (b) transferring them to the object to generate a corresponding set. Artists can then (c) compute the manipulator pose from the contact areas and (d) use the result as feedback to edit the contact areas on the hand and object in an iterative feedback loop. Artists can alternatively (e) first create a (often coarse) pose using traditional posing tools and designate contact areas afterword. (f) Refinement continues until the final result is acceptable.

provide the artist with the ability to define and change their own axis , but also provide a default axis to start with.

Some recent research also demonstrates flexible use of local log map parameterizations for contact, but defines the contacts differently, including based on a boundary or based on a single representative point [Lakshmipathy et al. 2022, 2021]. Neither existing representation was effective for our purposes, and we compare to them directly in Section 8.1.

Our method of designating transfer via curves also bears some similarity to existing work on cloth layering for arbitrary bodies via an ordered set of "freedom marks" [Igarashi and Hughes 2002] as well as determining equivalent functional maps [Gehre et al. 2018]. Our method can be viewed as extending the literature on designating correspondences via input curves, in this case moreso as a basis for IK.

4 METHODS

Figure 1 illustrates the workflow we aim to enable, which requires the ability to create and manipulate contact areas, transfer areas between surfaces, and compute the appropriate pose. To support this workflow, we present our contact model, define the operations supported by our model, and detail the optimization process used to compute character pose from contact area correspondences.

4.1 Contact Parameterization & Transfer

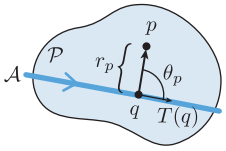
4.1.1 Smooth formulation. We define a contact patch $\mathcal{P} \subset \Omega$ as a closed region on surface Ω. To enable direct editing of contact patches, we must develop a representation that enables patch *transfer* and patch *manipulation*. *Transfer* is the task of computing an as-isometric-as-possible embedding of a patch $\mathcal{P}$ on a source surface Ω into another target surface Ω'. Patch *manipulation* includes any isometric transformations to $\mathcal{P}$ (translation, rotation, etc.) Importantly, our representation must enable interactions with patches to be intuitive, predictable, and fast.

Our first insight is that embedding an open curve is significantly easier than embedding a region or closed loop. Hence we parameterize $\mathcal{P}$ with respect to an open curve $\mathcal{A}$ which we call an "axis". We define axis $\mathcal{A}$ as a piece-wise geodesic curve on Ω containing:

(1) a finite set of points $\{a_1, \ldots, a_m\} \in \Omega$, with a *shortest geodesic* g_i connecting each pair of adjacent points (a_i, a_{i+1}) for $i = 1, \ldots, m-1$;
(2) *turning angles* $\{\phi_i\}_{i=2}^{m-1}$, where each ϕ_i is the angle of rotation from the ending direction of g_{i-1} to the initial direction of g_i, expressed in the tangent space of a_i.

For every point $p \in \mathcal{P}$, we store:

(1) the closest point q on $\mathcal{A}$
(2) the *logarithmic map* $\log_q(p)$ encapsulating the distance r_p from q to p and the initial direction θ_p of a geodesic from q to p expressed in the tangent space of q. The angle θ_p is taken relative to the tangent direction $T(q)$ of $\mathcal{A}$ at q. (If $q = a_i$ for some i, then $T(q)$ is taken as either the initial direction of g_i, or the ending direction of g_{i-1}.) See inset for diagram.



With this parameterization, $\mathcal{P}$ can be transferred from Ω to Ω' by first transferring the axis $\mathcal{A}$. In what follows, let primed quantities denote the corresponding transferred quantity on Ω'. To transfer $\mathcal{A}$, the position of its first point and initial tangent direction must be specified on Ω'; then the position of $\mathcal{A}'$ on Ω' is completely determined by re-tracing its constituent geodesic segments on Ω'. The transferred patch $\mathcal{P}'$ can then reconstructed on Ω' by simply applying the *exponential map*, the inverse of the log map: For each $p \in P$, the corresponding $p' \in \mathcal{P}'$ is obtained by tracing a geodesic starting from q' at an angle θ, taken relative to the tangent of $\mathcal{A}'$ at q', for a distance r.

The reason for parameterizing points $p \in \mathcal{P}$ relative to closest points in $\mathcal{A}$ is so that $\mathcal{A}$ acts as an intuitive control handle roughly corresponding to the "skeleton" of $\mathcal{P}$. Using closest points also distributes these "basis points" along $\mathcal{A}$, which prevents the foldover that might occur in the transferred patch if we only parameterized P with respect to a single point (see Figure 14.)

In general, the embeddings that result from this parameterization will not be truly isometry-preserving; for example, a geodesic ball in a region with positive curvature will have a smaller area than a geodesic ball of the same radius in a region of zero curvature. The embedding will be isometry-preserving if and only if the source region on Ω and target region on Ω' are themselves isometric, which will almost never occur. However, the strength of this method is that it can yield a reasonable embedding that approximately preserves the shape of the patch, whether or not an isometric embedding exists. Such robustness is especially important in artist-driven workflows, where physical and geometric plausibility is usually not strictly followed. Furthermore, the majority of surface regions used as contact regions in practice have low curvature, which inherently minimizes distortion.

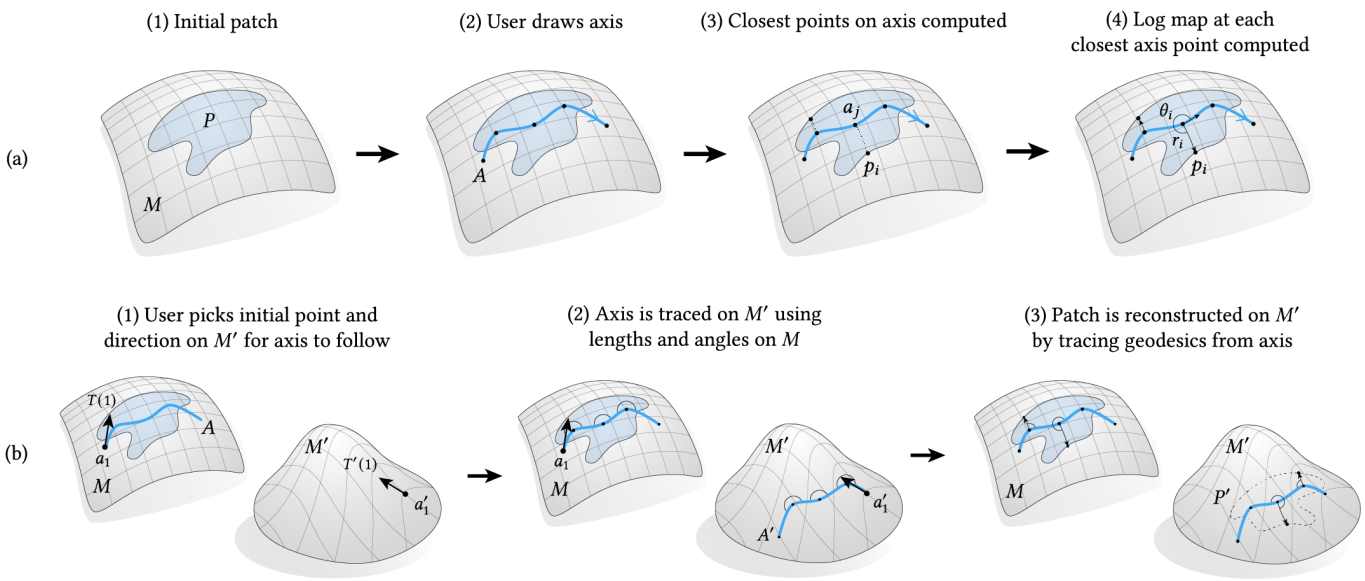


Fig. 2. An overview of contact patch parameterization (a) and patch transfer (b). See Section 4.1 for technical detail.

The parameterization is also robust to the choice of axis. The exponential map is surjective by the Hopf-Rinow theorem, meaning any point p on a connected surface Ω can be reached from any other point $q \in \Omega$ by following a geodesic from q for the appropriate distance with the appropriate initial direction. Thus a parameterization can be computed for any choice of axis (although some axes will be more intuitive or yield better parameterizations than others – see Section 9).

4.1.2 Discrete implementation. Parameters and the transfer process for the discrete formulation are illustrated in Figure 2. We discretize the surfaces Ω, Ω' as manifold triangle meshes M and M' respectively, both of which may or may not have boundary. Analogous to our setup in the smooth setting, the discrete axis A is specified by:

(1) a finite set of points $\{a_1, \ldots, a_m\}$ on M, with a discrete geodesic g_i connecting each pair of adjacent points (a_i, a_{i+1}) for $i = 1, \ldots, m-1$;
(2) turning angles $\{\phi_i\}_{i=2}^{m-1}$, where each ϕ_i is the angle of rotation from the ending direction of g_{i-1} to the initial direction of g_i, expressed in the tangent space of a_i.

The patch is represented as a set of barycentric points $P = \{p_1, \ldots, p_n\}$ on M. For every point $p_i \in P$, we will compute:

(1) the closest point $a_j \in \mathcal{A}$ to p_i;
(2) the log map $\log_{a_j}(p_i)$, which represents the geodesic distance r_i from a_j to p_i, and the initial direction θ_i of a geodesic from a_j to p_i, expressed in the tangent space of a_j. The angle θ is taken relative to the tangent direction $T(a_j)$ of $\mathcal{A}$ at a_j. $T(a_j)$, also denoted $T(j)$, is defined as the initial direction of the geodesic connecting a_j to a_{j+1}, expressed in the tangent space of a_j; for the axis endpoint a_m, we use the direction from a_m to a_{m-1} instead.

In our plugin, the user need only specify the sequence of points $\mathcal{A}$, while $\{g_i\}_{i=1}^{m-1}$ and $\{\phi_i\}_{i=2}^{m-1}$ are computed from A automatically. More optimally, a typical use has the artist select only the start and end of the axis with all edge crossings along the geodesic connecting those two points automatically added to the axis representation, minimizing requirements for user input while simultaneously providing a dense representation. Tangent vectors are represented as complex numbers in a local coordinate system. Defining a basis for tangent spaces at points on faces and edges is relatively straightforward, since each face and pair of adjacent faces is on its own intrinsically flat; we define tangent spaces at vertices using the definition of previous works [Sharp et al. 2019b]. Importantly, the definition of tangent bases at any discrete component enables our method to be *sampling-agnostic*, which we demonstrate enables smooth manipulation equally well on both coarse and fine meshes. To evaluate the exponential map, we trace discrete geodesics using the scheme described by [Polthier and Schmies 2006].

To parameterize patch P, we first compute the closest point in A to each point $p_i \in P$. Determining the closest point with absolute accuracy is not essential since the closest point only acts as the point at which the log map will be computed and does not affect the accuracy of the log map itself. In our experience, we found that it sufficed to only consider the points $\{a_1, \ldots, a_m\}$ as closest points. Hence, we use the fast but approximate Vector Heat Method [Sharp et al. 2019b] to do closest-point interpolation, which determines the closest point $a_j \in A$ to each point $p_i \in P$.

For each such pair $(p_i, a_j) \in P \times A$, we then compute $\log_{a_j}(p_i)$. Since the log map accuracy directly affects the accuracy of the patch reconstruction, we use an exact polyhedral geodesic algorithm [Mitchell et al. 1987], referred to here as MMP, to explicitly compute the exact geodesic h_i from a_j to p_i. We compute $\log_{a_j}(p_i)$ by computing the initial direction θ_i of h_i in the tangent space of a_j. In general, θ_i may be expressed with respect to an arbitrary polar

axis; we express θ_i with respect to the outgoing direction $T(j)$ of g_j at a_j, i.e. we store $\tilde{\theta}_i = \theta_i / T(j)$. We store $\log_{a_j}(p_i)$ as a single complex number $z_i := r_i e^{i\tilde{\theta}_i}$, where r_i is the length of h_i. Figure 2(a) illustrates the process of patch parameterization.

We must also encode the axis A for reconstruction on M'. For each pair (a_i, a_{i+1}) of subsequent points in A, we store the length of the geodesic segment g_i connecting them. For each point a_i for $i \in \{2, \ldots, m-1\}$, we also store the turning angle ϕ_i at a_i, i.e. the angle between the incoming direction of g_{i-1} at a_i, and the outgoing direction of g_i at a_i, expressed in the tangent space of a_i. These lengths and angles determine the position of A, up to an overall translation and rotation which must be stored explicitly.

We now describe the process of patch transfer. In what follows, primed quantities denote quantities on M' that have been transferred from M. To transfer A to the target surface M', the user selects a point $a'_1 \in M'$. The user also specifies the initial direction $T'(1)$ of the first geodesic segment g'_1 of the transferred axis A'. The axis A is transferred simply by tracing out its constituent geodesics on M', starting from a'_1 and using $T'(1)$ as an initial direction; the rest of the axis is traced using the lengths and turning angles between geodesic segments computed from A on M. After the axis has been transferred, the transferred patch P' is reconstructed on M' by tracing geodesics from the patch parameterization, subject to one caveat. In the context of transfer, we actually want P' to be the *mirror image* of P since $P \subset M$ and $P' \subset M'$ are intended to be in contact with each other with opposing normals. Hence, the angles used for geodesic tracing during axis and patch transfer will be negated compared to the angles computed for their respective parameterizations on M. Figure 2(b) illustrates the process of patch transfer.

As discussed in Section 4.1, users are free to designate a variety of axes A. This flexibility jointly makes interaction with contacts very robust to user input and provides fine-grained control over where and how the patch should be embedded into the target surface. For example, if the user wanted the tentacle in Figure 4 to wrap around a cylindrical object, one intuitive option would be to draw the axis to coincide with the longitudinal axis of the tentacle patch. Choosing different initial directions for that axis creates different wraps.

Our parameterization allows us to reconstruct complex contact patch shapes exclusively through computation of geodesics and geodesic path tracing, which depends only on the mesh being manifold. Even after transfer, users can easily manipulate the patch on M' simply by manipulating the axis A'; the patch is simply re-constructed upon any changes to the axis. Parameterization is the most expensive step of our method due to its reliance on an exact polyhedral method; however, it is a one time process and tends to not take very long in practice. Please see the supplementary video for approximate wall clock timescale on an example. Patch transfer and manipulation only requires the inexpensive operation of geodesic tracing and can be performed in real-time. We expand upon the supported contact patch editing operations in proceeding subsections and provide pseudcode for all algorithms in the appendix.

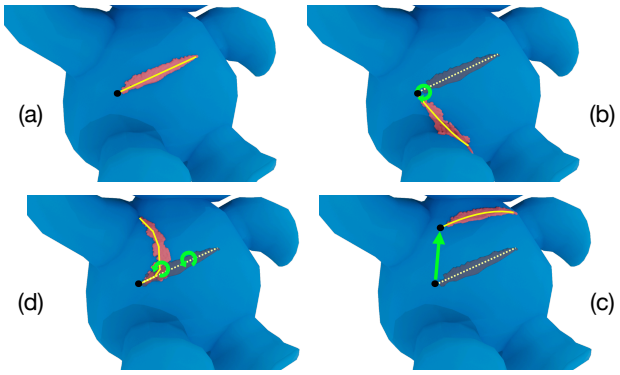


Fig. 3. Illustrations of our editing operations applied to a sample contact. (a) Starting from a contact painted on a coarse mesh using our custom brush, users can (b) rotate, (c) translate, or (d) locally deform the patch directly on the source domain. Black dots indicate axis start points.

4.2 Contact Painting

Contacts can be created either using off-the-shelf brush tools already provided by Maya or by our custom paint brush. We leave the choice up to users but briefly discuss use cases for both options.

On fine meshes or regions with dense sampling, users may find it beneficial to select existing discrete vertices. Vertex selection is flexible, inexpensive, has no parameters beyond the brush size, and is a method familiar to most artists. The drawback, however, is that this method does not enable selection of arbitrary points such as those within faces or in between edges, which leads to problems creating dense contacts on coarse meshes.

In response, we created our own brush tool which enables arbitrary point selection by discretizing a NURBS curve drawn in screen space and projecting the points into mesh space via standard ray-mesh intersection. Our custom brush offers a solution to the coarse mesh problem; however, it requires users to designate possibly unintuitive parameters such as the discretization step size and does not provide variable brush stroke sizes.

For these reasons, we allow heterogeneous creation of contacts using both brushes, which combined permit users to work directly with the original desired triangulated mesh, even if the mesh is irregularly sampled or has poor triangulation in general.

4.3 Contact Editing

Our model supports three editing operations which aim to be as isometric as possible: translation, rotation, and transfer. We additionally support local deformation (bending) of the patch, as well as hierarchical composition by linking multiple axes together. Since the location of the patch is determined by its parameterization with respect to its axis, to edit the patch we need only edit the axis. Defining all patch editing operations as operations on the axis simplifies the editing process conceptually and computationally. Users can directly manipulate large and arbitrarily shaped contact regions using only the axis as a simple, low-dimensional control handle. All patch editing operations, which can be viewed in the supplementary video, execute at interactive rates. Pseudocode for each operation is available in the appendix.

4.3.1 Translation. We define patch translation as rigidly moving a patch along a specified path with as little rotation as possible with respect to the path. Mathematically, this definition corresponds to *parallel transport* of the patch along the path of translation. In practice, we found that simply transporting the patch along the shortest geodesic connecting the start point to the end point of the path of translation produced more predictable behavior than parallel-transporting the patch along the user-drawn path of translation. The process of patch translation is implemented as follows:

(1) the user drags the patch across the surface;
(2) the shortest geodesic between the start and end point of the dragging path is computed using MMP;
(3) the initial direction $\vec{d}'$ at the first point a'_1 of the patch axis is transported along this shortest geodesic, using *discrete parallel transport* [Sharp et al. 2019b];
(4) the axis is re-traced at the end point, and finally the patch is reconstructed using its axis-based parameterization.

In the third step, discrete parallel transport of a vector with respect to a path involves applying a change-of-basis to the vector to keep it parallel as it passes through different local coordinate systems, which amounts to a sequence of simple complex arithmetic operations. See Section 5.2 in [Sharp et al. 2019b] for details.

4.3.2 Rotation. Rotation is the task of rotating the whole patch rigidly around a designated pivot point. Rotating the patch around a'_1 amounts to simply choosing another direction for initial direction d' of the transferred patch axis A'. Although it is possible to similarly define rotation around any other point in A' by constraining the axis tangent direction at that point, the pivot point is always chosen to be a'_1 for simplicity.

4.3.3 Local Deformation. While the editing operations defined so far are intended to be as-rigid-as-possible transformations acting on the whole patch, we define local deformation as transforming only part of the patch rigidly, which corresponds to rotating part of the axis $\{a_i, a_{i+1}, \ldots, a_m\} \subset A$ around the point $a_i \in A$ while keeping the rest of A fixed. Like all other patch editing operations, local deformations do not change the connectivity of the axis. Unlike global editing operations, however, this local rotation requires that we must update the turning angle ϕ_i at a_i.

4.3.4 Hierarchical Composition. We allow the user to edit a system of multiple patches in a hierarchical manner, by "parenting" (possibly multiple) child patches to a parent patch. For each child patch, we use MMP to connect its axis to the axis of its parent with a geodesic. We choose to connect the first points in the axis for simplicity, but note that connections can alternatively be made from the end point or any intermediate point if desired. The connected axes are equivalent to a single axis, with the only difference being that axis points may now possess multiple successors in the case of branching. We store the turning angles and segment lengths corresponding to each newly-added axis point. Therefore, whenever the axis of the parent is updated, we can re-trace each child axis by applying the usual axis-tracing procedure to the single connected axis. Using the connected axes ensures that the position of each child patch relative to its parent patch is approximately maintained throughout any operations applied to the parent patch.

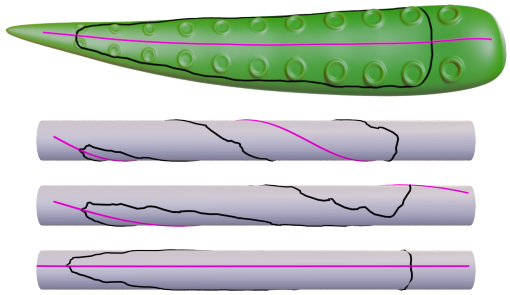


Fig. 4. By drawing an appropriate axis (magenta), the user can easily specify that the patch on the tentacle should be transferred to wrap around a cylindrical object. Changing the start angle of the axis changes the wrap.

4.4 Kinematic Hand Posing

After corresponding contacts on the object and hand have been created and paired, we adopt a modified formulation of the optimization in [Lakshmipathy et al. 2022] to compute the hand pose that best matches the contacts, while simultaneously accounting for surface geometries and user edits. Specifically, we start with the existing formulation:

$$\begin{aligned} \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \quad & \textstyle\sum_{i=0}^{N} \ (\lambda_d \Gamma_{D,i} + \lambda_n \Gamma_{N,i}) + \sum_{j=0}^{J} \ \lambda_p \Gamma_{P,j} \\ \text{s.t.} \quad & \boldsymbol{\theta_L} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta_U} \end{aligned} \tag{1}$$

where $\boldsymbol{\theta}$ is the degree of freedom vector, N is the total number of corresponding contact vertices, $J = |\boldsymbol{\theta}|$, $\boldsymbol{\theta_L}$ and $\boldsymbol{\theta_U}$ define the lower and upper bounds of $\boldsymbol{\theta}$ respectively, $\Gamma_{D,i}$ penalizes for distance from contacts, $\Gamma_{N,i}$ attempts to align opposing surface normals, and $\Gamma_{P,j}$ encourages solutions near a rest pose or artist-defined prior.

Unfortunately, applying Eq. 1 as-is *does not work* for unconstrained, high-DOF systems characteristic of standard animation rigs. Unlike robot hands whose DOFs are constrained by real-world considerations, animation rigs are intended to provide as much artistic flexibility as possible. A common approach is to model all child joints as full 3-DOF ball joints and the root as a full 6-DOF joint. Note that production rigs are typically far more complicated than this formulation, often sporting more complex IK and NURBS handles. Even under this simple formulation, however, we found that adopting Eq. 1 as-is produced mediocre results with highly undesirable characteristics such as contorted fingers and significant self-intersections. See Figure 11c for one such example.

To address this problem, we make two modifications. First, we drop the root DOF contributions to Γ_P. Second, we select a much higher λ_p than described in the literature. By introducing the aforementioned modifications, we are able to reliably position the manipulator near the contacts while restraining alterations to the rest pose. The revised formulation is thus:

$$\begin{aligned} \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \quad & \textstyle\sum_{i=0}^{N} \ (\lambda_d \Gamma_{D,i} + \lambda_n \Gamma_{N,i}) + \sum_{j=5}^{J} \ \lambda_p \Gamma_{P,j} \\ \text{s.t.} \quad & 0 \leq \boldsymbol{\theta} \leq 2\pi \end{aligned} \tag{2}$$

The optimization with these surprisingly simple modifications worked very well for us, scaling to rigs possessing north of fifty degrees of freedom, showing some robustness to starting position of the root, and retaining the benefits of cheap computation cost and full differentiability. We formulate and solve Eq. 2 using the NLOPT optimization library [Johnson 2017].

5 APPLICATIONS, EXPERIMENTS, AND RESULTS

We illustrate the benefits of our approach using a variety of objects and 4 custom manipulators: a human hand, a long-fingered witch hand, a three-fingered alien hand, and a tentacle. All experiments were conducted using our Maya plugin coupled with the geometry-central library [Sharp et al. 2019a].

5.1 Static Grasps

Figures 5 and 6 illustrate some high quality poses synthesized using our framework. Generating poses such as these using a traditional process is time consuming and iterative; however, our approach allows the user to quickly set up the desired shot by specifying contact locations, after which the corresponding manipulator pose is automatically calculated in a manner that accounts for both rig kinematics and skin deformation.

5.2 Rapid Grasp Transitions

Figure 7 illustrates a sequence of transitions between multiple pitching grips, each of which was constructed through direct manipulation of contacts on the surface of a baseball. Each transition required exactly three manipulation operations: two translations and one rotation to go from the knuckleball to the fastball, and three translations to move from the fastball to the curveball. The entire sequence was generated by an untrained animator using reference photos in less than an hour after initial scene setup. Subtle details, such as closing the gaps between fingers, were made by creating corresponding contact patches on different fingers of the same mesh. The full sequence can be viewed in the supplementary video.

5.3 Hierarchical Contact Transfer

Artists can also paint multiple contacts onto the hand and then map them to an object in one broad sweep using hierarchical contact patch transfer. Figure 8 shows an example. Contacts on the hand are mapped to different areas of the bunny with a few mouse clicks. Our as-isometric-as-possible transfer process adapts well to the different geometric features on the bunny surface.

Figure 9 illustrates another example of this type of workflow, this time showing adaptation to changes in object scale – a unique benefit to our approach. This sequence was constructed using 4 keyframes in under 45 minutes by an inexperienced animator. To do so, we transferred the contacts from the prior experiment to an identical starting point and axial direction of the sphere at each scale, and use the solution from the previous keyframe as a prior for the subsequent frame to reduce optimization time. The animated result can be seen in the supplementary video. Note that while alternative techniques such as drawing from pre-compiled taxonomies (e.g. [Li et al. 2007; Sanso and Thalmann 1994]) may yield similar results in the case of human hands, our method does not require any pre-compiled hand pose priors and makes no assumptions of hand kinematics – artists could easily replicate the same procedure on a non-anthropomorphic hand, for example, where taxonomies may be impossible to compile.

5.4 Difficult Grasps on Higher Genus Topologies

Routing multiple fingers through holes and around handles is an especially challenging task due to the large number of inter-object and self-intersections generated during traditional posing, many of which would require movement of the palm and a subsequent total repositioning of the fingers in the updated state to resolve. Figure 10 shows two examples. Using contacts allows the animator to designate which parts of the handles are desirable for contact and automatically generate a pose for the palm optimized for the desired grasp. Different fingers can also be routed towards specific handle regions to reduce self-intersections, while automated computation of the palm location can determine the optimal base pose position required to produce such a routing.

5.5 Ablation Studies

To demonstrate the necessity of each term in Eq. 2, we performed an ablation study in which each optimization term is individually removed. Figure 11 illustrates the results of the study. As anticipated, removal of any terms results in catastrophic failure.

Removal of Γ_D removes all incentive to progress towards the contacts, resulting in failure to even reach the desired object. Removal of Γ_N permits negligence of the geometry at the contact point locations, which creates ambiguity in distinguishing the object interior and exterior. This omission caused the tentacle to find a solution which conformed to the inside of the pole during part of the wrap rather than the outside. Finally, removal of Γ_P commonly results in highly undesirable pose artifacts such as contorted fingers and large regions of self-intersection. We note that this omission is particularly problematic in the context of unconstrained problems where there is otherwise no quantitative regularization to prevent unnatural behaviors from emerging.

5.6 Extensions

Although we have primarily considered manipulator-object interaction in this paper, the general-purpose nature of our contact formulation permits application to arbitrary contact-rich contexts.

Figure 12 illustrates an example in which various parts of a full body are designated to make contact with the chair and floor. To animate this sequence from the initial standing position after specifying the contact sets, we start by fixing all joints and solving for the position of the root joint at the center of the pelvis. Next, we unlock the lower body and solve for the leg positions which align the feet with the floor. Finally, we unlock the left and right arm subsystems and solve in order to bring the arms inward. Another use case is to minimize foot movement during the in-betweens of the sitting procedure, which we can do by solving for the foot contacts while the remainder of the upper body continues with the sitting interpolation.

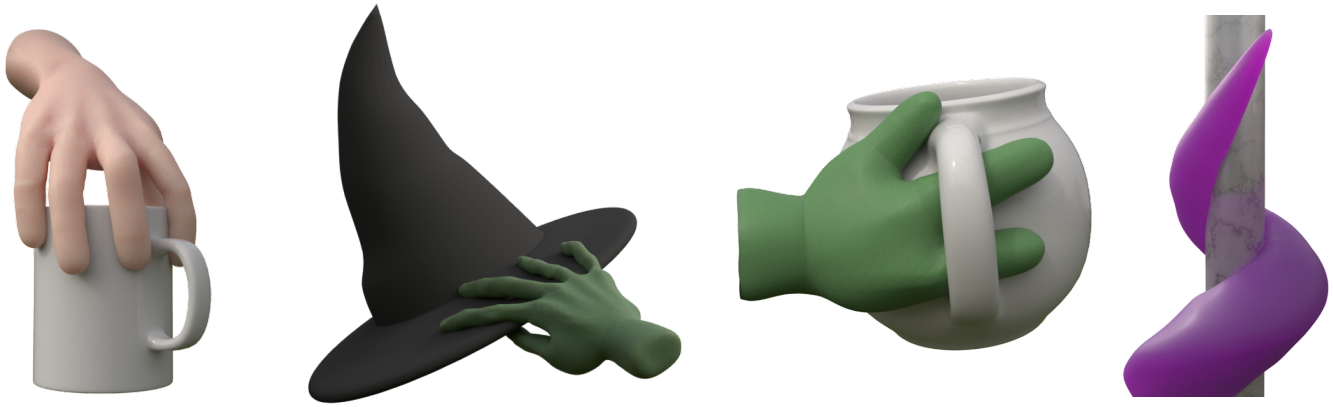


Fig. 5. Poses created using our contact edit framework across kinematically and morphologically diverse manipulators in realistic in-situ contexts.

Fig. 6. Additional poses generated by our solver across various manipulators and geometries.

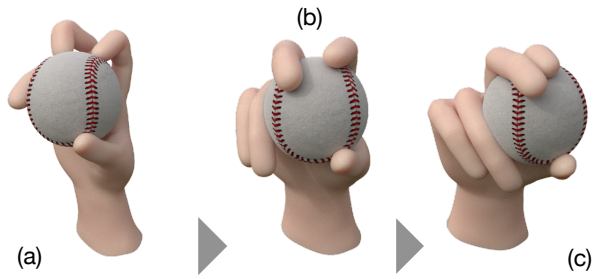


Fig. 7. Our technique can create smooth transitions between multiple baseball pitching grips, including a (a) knuckleball, (b) two-seam fastball, and (c) curveball.

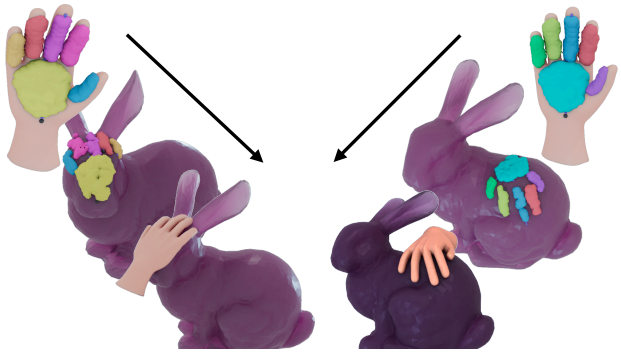


Fig. 8. Bulk transfer of a hierarchically organized identical set of contacts on the hand complies to local geometric features on the head (left) and body (right) of the Stanford bunny.

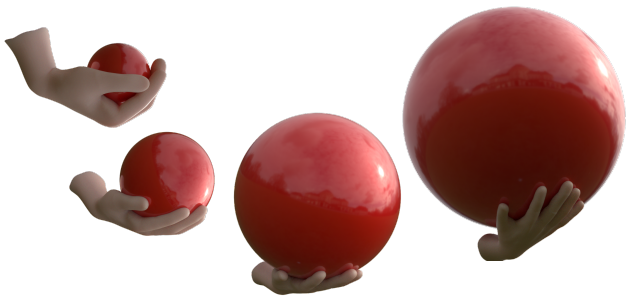


Fig. 9. Demonstration of a manipulator adapting its grip in response to variance in object scale.

6 USER STUDIES

We conducted a small user study to gauge the effectiveness of our plugin in the hands of real animators.

6.1 Subject Recruitment and Distribution

Five subjects over age 18 were recruited from a combination of channels including internal university mailing lists, in-class announcements, and referrals. Subjects were expected to have basic working proficiency in Maya, including the ability to control rigs and change camera views. Three out of five subjects came exclusively from an art background, while two had some technical background.

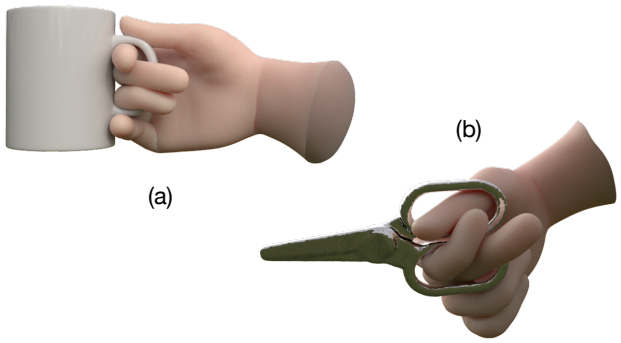


Fig. 10. Challenging grasps generated using our method (a) curling the fingers around the cluttered interior of a genus-1 mug handle and (b) between the two holes of a pair of genus-2 scissors.

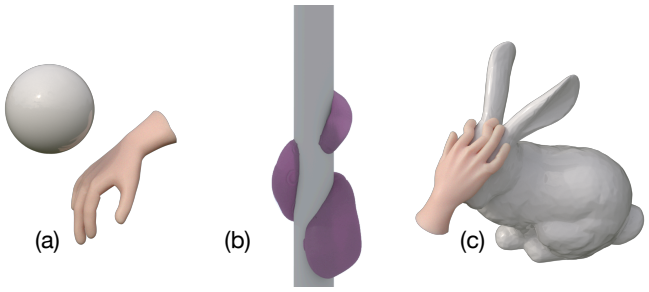


Fig. 11. Catastrophic failures resulting from the removal of optimization terms, including (a) distance, (b) normal, and (c) prior penalties.

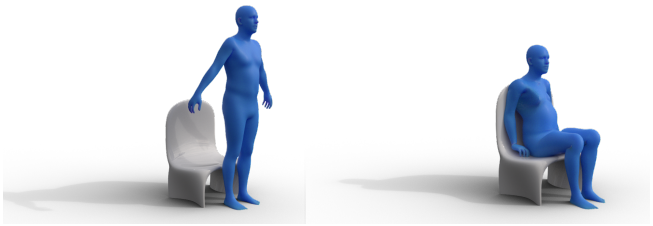


Fig. 12. Result of applying our method to designate contacts on a full body (a) in preparation for sitting in a chair and (b) settled into the final pose.

6.2 Setup

Participants were asked to partake in three one hour sessions and were compensated with a $20 Amazon gift card at the end of each session for a total of $60 per subject. Participants were asked to watch a 9 minute tutorial video prior to attending the first session but were not informed of session details in advance. All studies were conducted in-person with a proctor present during each session. Each session was conducted on a single workstation running an Intel i7 3.1 GHz processor with 8 GB of RAM. Sessions were screen-recorded without audio to limit exposure of personally identifiable

Fig. 13. All user study items

information (PII). Participants were additionally provided a Maya scene with a skinned and rigged hand asset, a 3D digital scan of a physical object, and the physical object itself as reference material. 3D scans including geometry and textures were collected using an Artec Space Spider[2]. Participants were permitted full internet use during each session and were permitted to ask unlimited questions. Figure 13 includes all objects used during the studies. For simplicity, and to reduce the learning burden, we provided subjects access only to the default vertex selection paintbrush already included in Maya due to the high polygon counts of the 3D scanned meshes.

In the first session, participants were given an object by the proctor and asked to create as many good quality, natural looking poses as possible using the provided hand rig and object mesh. Participants were then asked to recreate the same poses on a best effort basis in the second session using view-only replicas of their original poses. Participants were tasked with performing the first and second sessions either using our plugin or existing Maya tooling only. The order of tool usage was randomized to reduce ordering bias. In the third session, participants repeated the first session procedure but were allowed to select any object in Figure 13 and use our tool as much or as little as desired. In an effort to simulate a production workflow, participants were not allowed to modify the provided hand rig, skinning weights, object or hand geometry, or asset scales; however, subjects were allowed to perform rigid transformations to the object or rotate any of the hand joints. Each subject was played an audio recording of directions at the start of each session. Surveys were sent to subjects at the completion of each session.

Subjects were given access to the tutorial video and a textual reference during each session in which our plugin was used. Although participants were told to create as many high quality poses as possible or replicate as many poses as possible in the previous session using the plugin, our primary intention in the first plugin session was to provide subjects with hands-on experience with our tool, and in particular the processes of creating, manipulating, and clearing contacts, using and observing the optimization process, and navigating the general UI layout. To test the intuitiveness of plugin usage, we further randomly subdivided the first plugin session into

[2]https://www.artec3d.com/portable-3d-scanners/artec-spider

two pools: participants in the first pool were provided guided suggestions by the proctor for exploiting plugin features, while those in the second pool only received answers to queries.

Rather than requiring users to designate an axis during contact creation time, we provided users with an automatically computed initial default axis to use with any newly created contact patch. We tried two methods for computing a default axis. In the first method, we simply connect the two furthest points on the patch with a geodesic. In the second, we use an approximation of the patch's medial axis as the default axis: we use the heat method [Crane et al. 2017] to compute a smoothed distance field u from the boundary of the patch, compute the gradient ∇u, and record points where ∇u changes direction by an angle of more than $\pi/2$ as points on the (smoothed) medial axis. These points were then greedily connected into a curve using Dijkstra's algorithm.

6.3 Results

Our findings were exciting and in some cases surprising. Although the subject pool was small, the response was generally positive. We elaborate upon detailed observations, positive feedback, and criticisms in the proceeding paragraphs.

One subject opted to use our plugin as a "last-mile" solver by painting contacts after creating an initial pose manually, while the remainder, as expected, used our plugin as a "coarse" solver by instead painting contacts in the beginning, solving for the pose automatically, and making minor adjustments manually at the end, if at all. 100% of subjects opted to paint contacts on the hand first and subsequently transferred them to the object. Three subjects opted to change the auto-generated default axis to an orientation consistent across all contacts. The remaining two, who opted to keep the default axis in all cases, opted to bring the object close to the hand before transferring contacts. All subjects could easily tell when transfer results were undesired and were able to quickly alter orientations via contact rotation. Subjects who heavily used contact rotation and translation did so with fine-grained movements and expressed satisfaction with the result. Local deformation was used sparingly. Three subjects opted to start from scratch for each new pose, while two chose to return to a template contact distribution for re-use across multiple poses and expressed interest in having an existing library for common regions such as fingertips or upper palm regions. All subjects that made fine-grained adjustments post optimization did so through manual rig adjustment rather than creating or modifying contacts. With the exception of one subject who created very sparse contact sets, users never modified any weighting hyperparameters. All subjects opted to visualize the optimization progress to understand how the final solution was generated and, more interestingly, how they could modify variables such as object or contact placement to produce even more desirable results.

Despite the large variance in usage across even a limited subject pool, the response from all subjects in the third session was positive. One subject opted to use the plugin exclusively during the third session, and the remainder chose to use both toolsets together. 100% of subjects claimed the plugin would be useful to other animators and found the tutorial video alone sufficient for using the tool regardless of whether guided suggestions were provided in the introductory session. The response to optimizer-generated poses was generally positive – most subjects were very satisfied, while some opted to make minor changes after completion.

Dissatisfaction and criticisms were consistently associated with one of three issues: solutions to difficult optimization problems such as those which involved very large wrist movement, contact transfers to object locations with vastly different curvature than the source region, or generic UI layout and the lack of hotkeys. Subjects who understood optimization limitations deliberately tried to push the limits of the solver, while those who encountered some unwanted artifacts usually just modified the object transform slightly and re-solved. We also found that neither of our default axis generation methods were really successful – regardless of the default axis provided, subjects either preferred to create their own axis or opted to move forward in spite of, rather than aided by, the generated default axis.

While the subject pool was small, these results do suggest that our tool is intuitive and predictable, and more importantly that those results held true across users with different existing skill, users with limited technical background, and in a variety of usage patterns. We found it especially encouraging that many subjects approached us directly after the final session and asked when the plugin would be available publicly, and even moreso that the inquires came from heavy Maya users. A number of subject-generated poses, along with more detailed results comparing usage time split per subject between our plugin and existing tooling, are available in the appendix.

7 QUALITATIVE ASSESSMENTS

To assess the quality of user study results, we additionally conducted a round of qualitative assessments.

7.1 Subject Recruitment and Distribution

We published and distributed a survey online to a total of 65 individuals with a background or interest in animation via a combination of internal mailing lists, individual outreach, and class announcements. We also solicited the individual response of an expert animator with ten years of experience. Subjects were required to be over age 18. All subject information, with the exception of the expert animator, was kept anonymous.

7.2 Setup

We collected 12 poses generated by users from the previous study which used our plugin in a substantial capacity and 12 poses created using fully manual FK tooling. Poses were selected based on overall quality by an internal review process. All selected poses are available in the appendix.

Each pose generated using the plugin was paired with another using only manual tooling primarily by user study subject ID, secondarily by item, and lastly at random with results from other subject IDs. All pairs of corresponding images are available in the appendix. Importantly, we note that the decision to primarily group by subject ID rather than object was to calibrate for individual skill level, both with respect to existing experience and the level of familiarity with our plugin vs. existing tooling. Since poses using the

plugin were largely sourced from the third session (e.g. where subjects were allowed to choose their object), nearly all pairs present non-corresponding items. Our results therefore assess the general ability of an individual subject with and without access to our plugin, which we consider more meaningful than comparing results between subjects.

Survey subjects were presented with 12 pairs of images in A/B format spanning each of the following categories:

- Overall visual appeal
- Realism / physical plausibility
- Complexity
- Natural appearance

Answer choice and category order was fully randomized to reduce ordering bias. Subjects were required to provide a response to all questions in order to receive credit for completion.

7.3 Results

Table 1 tabulates the results of the study, where each answer choice is awarded a score of +1 for a win, 0 for a loss, and 0.5 for a tie. Selections were required to have a vote differential of 4 or more to be considered the majority choice.

Table 1. Qualitative study results

Qualitative Study Results		
	Plugin Score	Manual Score
Overvall Visual Appeal	8	4
Physical Plausibility	6	6
Complexity	4	8
Natural Appearance	6.5	5.5

The expert opinion was relatively consistent with aggregate results. Specifically, 9/12 plugin poses were preferred for visual appeal, 8/12 for physical plausibility, 4/12 for complexity, and 10/12 for natural appearance. Overall, poses generated by the plugin performed as well, if not better, than *fully manual* posing in a majority of categories. Importantly, we note that the loss in complexity is expected since users were exposed to the plugin for the first time during the previous sessions. We note that poses such as those in Figure 10 are possible to generate, but require more experience. However, performance in the remaining categories clearly indicate that our tools have a substantial impact on pose quality.

8 COMPARISONS

8.1 Contact Model Comparisons

We compare our axis-based contact model to those of previous works [Lakshmipathy et al. 2022, 2021], which are boundary-based methods that store bounding contact points only and utilize a single-point log map embedding to respectively. Our model is optimized for usage as an artist tool, with the goal of minimizing the time taken both to digest an original user specification as well as to perform downstream editing and optimization.

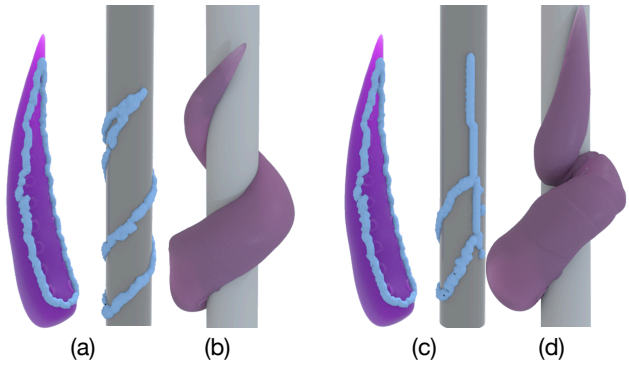


Fig. 14. Contact model behavior during transfer of a tentacle contact to a cylinder using (a) Our contact model and (b) its generated solution. (c) The contact model from [Lakshmipathy et al. 2022] and (d) its generated solution

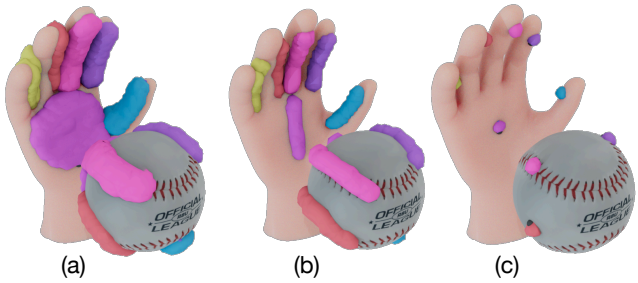


Fig. 15. Contact sets used as the basis for pose solution comparisons using (a) Our axis based area model (b) A curve-only model and (c) A single point model

It is possible to parameterize only the patch boundary and recover its interior via flood fill [Lakshmipathy et al. 2021]; however, since our brush tools already enable easy user selection of areas, all points in the interior will be used for downstream optimization, and flood fill is slow to run on fine meshes, it is faster to keep the whole region rather than re-compute the interior per edit, especially since our method is fast enough to perform real-time editing operations without compromising robustness. Bypassing flood fill also enables our operations to be sampling-agnostic, which provides consistent editing behavior and performance across different triangulations.

The axis model is also significantly more robust than the single-point log map embedding [Lakshmipathy et al. 2022], which notably suffers the same drawbacks of far-region distortion and foldover as prior works in the texture transfer literature. As illustrated in Figure 14, these limitations can result in unexpected complete breakdowns of editing and transfer operations in perfectly reasonable use cases, ultimately leading to unpredictable behavior.

8.2 Posing Method Comparisons

We also perform a comparison of the solutions generated by our optimization framework utilizing contact areas against both traditional single point IK and a “curve-only" variation inspired by the Line of

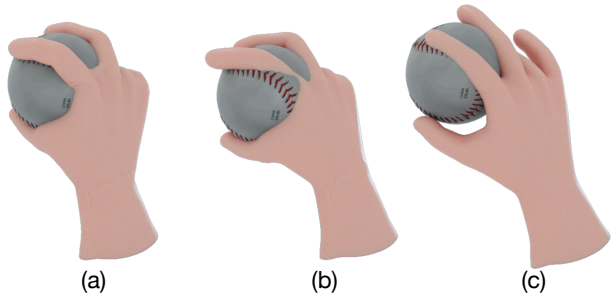


Fig. 16. Poses computed using (a) Our axis-based area model (b) A curve-only model and (c) A single point model

Action [Guay et al. 2013]. Figure 15 illustrates the contact sets used as the basis for each solution. Full contact areas were approximated as single points and curves respectively on a best-effort basis. We found that responses were fairly consistent regardless of curve or point placement in ambiguous cases such as the large palm contact. Corresponding object points were selected individually in the single point formulation and were transferred using our axis-based solution in the case of the curve and area formulation, where the curve itself was used as the axis in the former case. All solutions were computed from same start pose using the same solver and number of iterations, but note that weighting coefficients λ_d, λ_n, and λ_p had to be altered to account for differences in point density. Specifically, the values used are:

Table 2. λ weighting values used to generate results in Figure 16.

λ Weighting Values			
	λ_d	λ_n	λ_p
Single Point	3.5	6.1	10.0
Curve Only	1.0	1.2	10.0
Full Area	1.0	1.0	10.0

Figure 16 illustrates the poses computed under each formulation, while Table 2 provides the weighting coefficients used the generate the results. We note that both the single point and curve-only formulations resulted in solutions which produced highly unnatural finger configurations or significant penetration between the palm and the object respectively. The single point formulation also required extensive testing of contact point placement and hyperparameter tuning even to produce a somewhat similar result and was highly sensitive to changes, while curve and area formulations were less sensitive to changes. Importantly, we note that the choice of λ-values for the area based contacts not only produced the result in Figure 16, but also **every single result in the entire paper, and all but one result in the appendix**. The curve formulation also notably converged to similar solutions regardless of how the palm curve was oriented, assuming the number of points comprising the curve stayed the same.

We hypothesized and verified that the curve and area formulations produce similar solutions if contacts are confined to regions with an implicitly defined orientation (e.g. fingers only). The failure of the curve formulation to generalize to regions such as the palm,

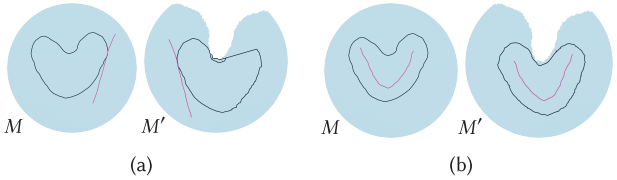


Fig. 17. (a) Our transfer method can produce large unwanted distortion if the axis (magenta) is drawn such that the patch parameterization depends on geodesics expected to travel outside of a non-convex domain. (b) However, desired results can still be obtained by simply choosing the axis to lie entirely within the patch region.

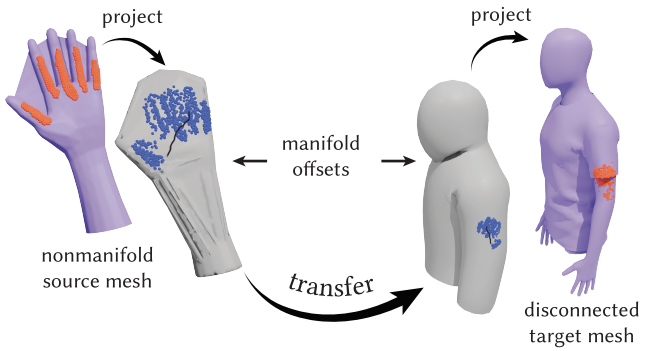


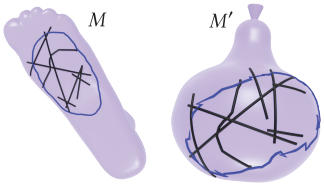
Fig. 18. Although the geodesic tracing step of our transfer process is not directly applicable to nonmanifold and disconnected surfaces, contact transfer can still be achieved between such surfaces via the use of a manifold offset mesh. Parameterization and transfer occurs between the offset surfaces, and patches are transferred between the original and offset surfaces via projection. Offset surfaces in this figure were constructed using the "shrinkwrap" modifier in the open-source 3D software Blender.

however, underscores a general limitation: given two curves of the same length, it is not possible to non-uniformly weight the importance of one curve over another without either artificially making one curve sparser than the other or introducing more hyperparameters to adjust the weighting. In contrast, an area formulation implicitly provides non-uniform weighting capability by design. We note this observation to be particularly valuable in the context of artist tools since users already make this designation during creation of the contact areas themselves, which improves overall intuitive understanding of tool usage in comparison to tweaking weighting hyperparameters. Additionally, the results demonstrate that, more generally, an abundance of correspondences is valuable in addressing several well-known drawbacks in IK solvers.

9 DRAWBACKS AND LIMITATIONS

Patch transfer can fail if there exist points p on the patch and q on the axis such that $d_M(p, q) \neq d_{M'}(p', q')$, i.e. the geodesic distance between p and q on M does not equal the distance between their corresponding points p', q' on M'; Figure 17a illustrates such a failure. However, the user can draw an alternate axis that yields the expected results (Figure 17b).

Parameterization quality may also degrade for certain choices of axes, for example axes with too few control points to adequately parameterize a patch with a complex shape, or axes that deviate significantly from the general shape of the patch. In practice, our method produces reasonably robust transfers even for a highly non-intuitive choice of axis (inset).



Our contact model is currently not well-suited for grasps that do not have sizeable areas of contact with the target object as well as grasps that span concavities or holes, such as the teeth of a comb or the holes of a button. While possible to model the contact as a set of many small patches, patch transfer and manipulation will be more susceptible to error. We also note that our optimization method, as illustrated by Figure 16(c), struggles when contact information is sparse, particularly in disambiguation of multiple solutions or resolution of uncanny artifacts.

Our method of contact parameterization and transfer is not designed for contact regions that belong to multiple disjoint surfaces, such as a figurine with separate layers of clothing. However, transferring such contacts is still possible under our framework. One option is to separately parameterize and transfer each portion of the contact region that belongs to a different surface. Alternatively, one can construct a single offset surface that encloses all components of either the source or target surface, project the contact region to/from the offset surface, and perform contact parameterization and transfer between the offset surfaces. On the other hand, transfer via a proxy surface often induces additional inaccuracy (see Figure 18). Robustly handling multi-component surfaces remains a challenge for future work.

Our solver produces locally optimal solutions only and tends to struggle with finding solutions which require large amounts of wrist movement. User study subjects expressed little concern with this limitation in practice since modifying the object transform or moving the wrist or object into a more favorable start position required minimal effort; however, increased sensitivity to initialization renders our approach more problematic in automation-focused settings.

Our current contact model also assumes manifold meshes, which is needed for the log map computations and tracing geodesics. Strictly speaking, however, we need only require that the mesh be locally manifold, rather than globally manifold. Specifically, each point on the axis on the source mesh needs to be manifold during contact parameterization, and the contact region on the target mesh must be manifold in order to reliably trace geodesics. Alternatively, one can perform mesh repair, or project onto/from a manifold offset mesh such as in Figure 18.

Other shortcomings involve the workflow of working with contacts. This issue was especially prevalent in non-human hands since the animator was expected to internally visualize the contact set without being able to replicate the pose themselves in the physical world. Maintaining finger spacing during transfer proved challenging as well, although hierarchical transfer partially combats the problem.

Two areas of future work are to reduce the time taken to draft contacts and extend our pose drafting capabilities to both full animations and VR / AR. Libraries of contact sets could assist with the former; however, full prediction is a notoriously challenging problem which requires a more extensive exploration into learning-based methods. The latter would likely require formulation of new interfaces, as our posing tools currently do not account for items such as interpolation, motion smoothness, or mid-air contact painting. We are excited by applications such extensions will enable.

10 CONCLUSION

In summary, we have presented EAD artist tools which enable intuitive modeling of high complexity, contact-rich manipulator-object interactions through the direct manipulation of contact areas. To do so we proposed a novel contact model formulation centered on a log map-based parameterization computed with respect to a user-chosen axis, which we demonstrate to be scalable to large contact regions, highly robust to the choice of axis and surface sampling variations, and capable of providing real-time, robust, and predictable updates in responses to edits including dragging, rotation, hierarchical composition, and transferring between surfaces. We have also introduced an optimization framework that is scalable to high DOF manipulators through two modifications of an existing formulation. Combined, our framework provides the first direct and responsive means of working with contact regions, which we demonstrate can lead to the synthesis of high quality results while allowing animators to retain full artistic control. We are confident that the foundations laid in this paper will serve as a strong basis for future studies and are hopeful that the combination of our distributed Maya plugin and emphasis on practicality will spur interest and encourage participation from the broader community in this line of work.

ACKNOWLEDGMENTS

The authors would like to thank Helena Yang, Selena Sui, and Robin Zhang for their helpful contributions in the development of the plugin. We would also like to acknowledge all anonymous user study and survey subjects for their participation, and the anonymous reviewers for their helpful suggestions in revising our manuscript. We also thank Meta Platforms Inc. for access to the 3D scanner. This material is based upon work supported by the AI Research Institutes program supported by NSF and USDA-NIFA under AI Institute for Resilient Agriculture, Award No. 2021-67021-35329 and NSF award CMMI-1925130.

REFERENCES

H. Biermann, I. Martin, F. Bernardini, and D. Zorin. 2002. Cut-and-Paste Editing of Multiresolution Surfaces. 21, 3 (jul 2002), 312–321. https://doi.org/10.1145/566654.566583

S. Brahmbhatt, C. Ham, C. C. Kemp, and J. Hays. 2019a. ContactDB: Analyzing and Predicting Grasp Contact via Thermal Imaging. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://contactdb.cc.gatech.edu

S. Brahmbhatt, A. Handa, J. Hays, and D. Fox. 2019b. ContactGrasp: Functional Multi-finger Grasp Synthesis from Contact. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

S. Brahmbhatt, C. Tang, C. D. Twigg, C. C. Kemp, and J. Hays. 2020. ContactPose: A Dataset of Grasps with Object Contact and Hand Pose. In *The European Conference on Computer Vision (ECCV)*. https://contactpose.cc.gatech.edu

X. Cheng, E. Huang, Y. Hou, and M. T. Mason. 2021. Contact Mode Guided Sampling-Based Planning for Quasistatic Dexterous Manipulation in 2D. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 6520–6526. https://doi.org/10.1109/ICRA48506.2021.9560766

S. Christen, M. Kocabas, E. Aksan, J. Hwangbo, J. Song, and O. Hilliges. 2021. D-Grasp: Physically Plausible Dynamic Grasp Synthesis for Hand-Object Interactions. *arXiv preprint arXiv:2112.03028* (2021).

K. Crane, C. Weischedel, and M. Wardetzky. 2017. The Heat Method for Distance Computation. *Commun. ACM* 60, 11 (Oct. 2017), 90–99. https://doi.org/10.1145/3131280

C. Ferrari and J. Canny. 1992. Planning optimal grasps. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*. 2290–2295 vol.3. https://doi.org/10.1109/ROBOT.1992.219918

A. Gehre, M. M. Bronstein, L. Kobbelt, and J. Solomon. 2018. Interactive Curve Constrained Functional Maps. *Computer Graphics Forum* (2018). https://doi.org/10.1111/cgf.13486

P. Grady, C. Tang, C. D. Twigg, M. Vo, S. Brahmbhatt, and C. C. Kemp. 2021. ContactOpt: Optimizing Contact to Improve Grasps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. http://www.pgrady.net/contactopt/

M. Guay, M.P. Cani, and R. Ronfard. 2013. The Line of Action: An Intuitive Interface for Expressive Character Posing. *ACM Trans. Graph.* 32, 6, Article 205 (nov 2013), 8 pages. https://doi.org/10.1145/2508363.2508397

H. Hamer, J. Gall, R. Urtasun, and L. Van Gool. 2011. Data-driven animation of hand-object interactions. In *2011 IEEE International Conference on Automatic Face Gesture Recognition (FG)*. 360–367. https://doi.org/10.1109/FG.2011.5771426

P. Harish, M. Mahmudi, B. Callennec, and R. Boulic. 2016. Parallel inverse kinematics for multithreaded architectures. *ACM Transactions on Graphics (TOG)* 35, 2 (2016), 1–13.

C. Hazard, N.S. Pollard, and S. Coros. 2020. Automated design of robotic hands for in-hand manipulation tasks. *International Journal of Humanoid Robotics* 17, 01 (2020), 1950029.

T. Igarashi and J. F. Hughes. 2002. Clothing Manipulation. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology* (Paris, France) *(UIST '02)*. Association for Computing Machinery, New York, NY, USA, 91–100. https://doi.org/10.1145/571985.571999

H. Jiang, S. Liu, J. Wang, and X. Wang. 2021. Hand-Object Contact Consistency Reasoning for Human Grasps Generation. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

S.G. Johnson. 2017. The NLopt nonlinear-optimization package. http://ab-initio.mit.edu/nlopt

A. Lakshmipathy, D. Bauer, C. Bauer, and N. S. Pollard. 2022. Contact Transfer: A Direct, User-Driven Method for Human to Robot Transfer of Grasps and Manipulations. arXiv:2110.15532 [cs.RO]

A. Lakshmipathy, D. Bauer, and N. S. Pollard. 2021. Contact Tracing: A Low Cost Reconstruction Framework for Surface Contact Interpolation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Y. Lee, S. Lee, and S.H. Lee. 2017. Multifinger interaction between remote users in avatar-mediated telepresence. *Computer Animation and Virtual Worlds* 28 (05 2017), e1778. https://doi.org/10.1002/cav.1778

Y. Li, J. L. Fu, and N. S. Pollard. 2007. Data-Driven Grasp Synthesis Using Shape Matching and Task-Based Pruning. *IEEE Transactions on Visualization and Computer Graphics* 13, 4 (2007), 732–747. https://doi.org/10.1109/TVCG.2007.1033

C. K. Liu. 2008. Synthesis of Interactive Hand Manipulation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 163–171.

J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. 1987. The Discrete Geodesic Problem. *SIAM J. Comput.* 16, 4 (1987), 647–668. https://doi.org/10.1137/0216045 arXiv:https://doi.org/10.1137/0216045

I. Mordatch, Z. Popović, and E. Todorov. 2012. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*. 137–144.

R. M. Murray, S. S. Sastry, and L. Zexiang. 1994. *A Mathematical Introduction to Robotic Manipulation* (1st ed.). CRC Press, Inc., USA.

N.S. Pollard. 1997. Parallel algorithms for synthesis of whole-hand grasps. In *Proceedings of International Conference on Robotics and Automation*, Vol. 1. 373–378 vol.1. https://doi.org/10.1109/ROBOT.1997.620066

N. S. Pollard and V. B. Zordan. 2005. Physically Based Grasping Control from Example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '05)*. Association for Computing Machinery, New York, NY, USA, 311–318. https://doi.org/10.1145/1073368.1073413

K. Polthier and M. Schmies. 2006. Straightest Geodesics on Polyhedral Surfaces. In *ACM SIGGRAPH 2006 Courses* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 30–38. https://doi.org/10.1145/1185657.1185664

M. A. Roa and R. Suarez. 2009. Computation of Independent Contact Regions for Grasping 3-D Objects. *IEEE Transactions on Robotics* 25, 4 (2009), 839–850. https://doi.org/10.1109/TRO.2009.2020351

R. M. Sanso and D. Thalmann. 1994. A Hand Control and Automatic Grasping System for Synthetic Actors. *Computer Graphics Forum* (1994). https://doi.org/10.1111/1467-8659.1330167

R. Schmidt, C. Grimm, and B. Wyvill. 2006. Interactive Decal Compositing with Discrete Exponential Maps. 25, 3 (jul 2006), 605–613. https://doi.org/10.1145/1141911.1141930

N. Sharp, K. Crane, et al. 2019a. geometry-central. www.geometry-central.net.

N. Sharp, Y. Soliman, and K. Crane. 2019b. The Vector Heat Method. *ACM Trans. Graph.* 38, 3 (2019).

B. Siciliano and O. Khatib. 2007. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg.

B. Smith, C. Wu, H. Wen, P. Peluse, Y. Sheikh, J. K. Hodgins, and T. Shiratori. 2020. Constraining Dense Hand Surface Tracking with Elasticity. 39, 6, Article 219 (nov 2020), 14 pages. https://doi.org/10.1145/3414685.3417768

O. Taheri, N. Ghorbani, M. J. Black, and D. Tzionas. 2020. GRAB: A Dataset of Whole-Body Human Grasping of Objects. In *European Conference on Computer Vision (ECCV)*. https://grab.is.tue.mpg.de

D. Turpin, L. Wang, E. Heiden, Y.C. Chen, M. Macklin, S. Tsogkas, S. Dickinson, and A. Garg. 2022. Grasp'D: Differentiable Contact-Rich Grasp Synthesis for Multi-Fingered Hands. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI* (Tel Aviv, Israel). Springer-Verlag, Berlin, Heidelberg, 201–221. https://doi.org/10.1007/978-3-031-20068-7_12

Y. Wu, J. Wang, Y. Zhang, S. Zhang, O. Hilliges, F. Yu, and S. Tang. 2021. SAGA: Stochastic Whole-Body Grasping with Contact. (12 2021).

K. Yamane and Y. Nakamura. 2003. Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on visualization and computer graphics* 9, 3 (2003), 352–360.

Y. Ye and C. K. Liu. 2012. Synthesis of detailed hand manipulations using contact sampling. *ACM Trans. Graph.* 31, 4 (2012), 1–10. https://doi.org/10.1145/2185520.2185537

H. Zhang, Y. Ye, T. Shiratori, and T. Komura. 2021. ManipNet: Neural Manipulation Synthesis with a Hand-Object Spatial Representation. 40, 4, Article 121 (jul 2021), 14 pages. https://doi.org/10.1145/3450626.3459830

J. Zhao and N. I. Badler. 1994. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics (TOG)* 13, 4 (1994), 313–336.

W. Zhao, J. Zhang, J. Min, and J. Chai. 2013. Robust Realtime Physics-Based Motion Control for Human Grasping. 32, 6, Article 207 (nov 2013), 12 pages. https://doi.org/10.1145/2508363.2508412

APPENDIX

A PSEUDOCODE

We assume the following atomic functions:

- ExactGeodesic(p, q) – computes the exact discrete geodesic between two barycentric points p and q on the same triangle mesh M. Our plugin uses MMP.
- InitialDirection(g) – returns the initial direction of a geodesic g in the tangent space of its origin point, expressed as a complex number.
- EndingDirection(g) – returns the ending direction of a geodesic g in the tangent space of its ending point, expressed as a complex number.
- TraceGeodesic(p, z) – trace a geodesic starting from a barycentric point p, along a direction and distance specified by the complex number z.
- ClosestPoints(P, A) – computes the closest point $a_j \in A$ to each point in $p_i \in P$, returning a vector C_P such that $\mathsf{C}_P[i] = j$. Our plugin uses the Vector Heat Method [Sharp et al. 2019b].

Algorithm 1 ParameterizeAxis(A)

Input: An array $A = [a_1, \ldots, a_m]$ of barycentric points on some triangle mesh M.

Output: A vector $\Phi_A \in \mathbb{C}^m$ where $\Phi_A[i] = \phi_i$, the turning angle at a_i; a vector $\mathsf{L}_A \in \mathbb{R}^m$ where $\mathsf{L}_A[i]$ is the geodesic distance between a_i and a_{i+1}; and a vector $\mathsf{T}_A \in \mathbb{C}^m$ where $\mathsf{T}_A[i] = t_i$ is the "tangent direction" at a_i.

$\Phi_A, \mathsf{L}_A, \mathsf{T}_A \leftarrow 0^m$ ▷ *initialize to length m*
$s_i \leftarrow 0$ ▷ *track the direction of the incoming geodesic*
for $i = 1$ to $m - 1$ **do**
  $g_i \leftarrow$ ExactGeodesic(a_i, a_{i+1})
  $t_i \leftarrow$ InitialDirection(g_i) ▷ *tangent direction of A at a_i*
  $s_i \leftarrow$ EndingDirection(g_i)
  $\mathsf{L}_A[i] \leftarrow$ Length(g_i)
  $\mathsf{T}_A[i] \leftarrow t_i$
  if $i > 1$ **then**
    ▷ *Compute turning angle at interior points a_i*
    $\Phi_A[i] \leftarrow t_i / s_i$
▷ *Special case for defining the tangent direction at the endpoint.*
$\mathsf{T}_A[m] \leftarrow s_i$
return $\Phi_A, \mathsf{L}_A, \mathsf{T}_A$

Algorithm 2 ParameterizePatch(P, A, T_A)

Input: A set $P = \{p_1, \ldots, p_n\}$ of barycentric points on some triangle mesh M_1; an array $A = [a_1, \ldots, a_m]$ of barycentric points on the same mesh M_1; and a complex vector $\mathsf{T}_A \in \mathbb{C}^m$ representing the tangents of A.

Output: A map $\mathsf{E}_P^A : P \to A \times \mathbb{C}$ that maps each $p_i \in P$ to a tuple (j, z_i) such that a_j is the point on A closest to p_i, $|z_i|$ is the geodesic distance between p_i and a_j, and $\arg(z_i)$ is the direction at which a geodesic must leave a_j to arrive at p_i, expressed in the tangent space of a_j.

▷ *Compute closest point on A for each point in P*
$\mathsf{C}_P \leftarrow$ ClosestPoints(P, A)
for $p_i \in P$ **do**
  $j \leftarrow \mathsf{C}_P[i]$ ▷ *index of closest point*
  $a_j \leftarrow A[j]$ ▷ *get the closest point on A*
  $h_i \leftarrow$ ExactGeodesic(a_j, p_i)
  $r_i \leftarrow$ Length(h_i) ▷ *get distance from a_j to p_i*
  ▷ *Compute direction in tangent space of a_j*
  $\theta_i \leftarrow$ InitialDirection(h_i)
  ▷ *Compute direction relative to the tangent direction of A*
  $\theta_i \leftarrow \theta_i / \mathsf{T}_A[j]$
  ▷ *Encode distance & direction as a complex number*
  $z_i \leftarrow r_i e^{i\theta_i}$
  $\mathsf{E}_P^A[p_i] \leftarrow (j, z_i)$
return E_P^A

Algorithm 3 ReconstructAxis$(\Phi_A, \mathsf{L}_A, a'_1, t'_1)$

Input: A vector $\Phi_A \in \mathbb{C}^m$ where $\Phi_A[i] = \phi_i$, the turning angle at $a_i \in A$; a vector $\mathsf{L}_A \in \mathbb{R}^m$ containing the lengths of the geodesic segments comprising the same axis A; an initial barycentric point a'_1 on some target triangle mesh M_2; and an initial direction t'_1 as a unit-magnitude complex number.

Output: An array $A' = [a'_1, \ldots, a'_m]$ of barycentric points on M_2; and a vector $\mathsf{T}_{A'} \in \mathbb{C}^m$ whose ith element is the "tangent direction" at a'_i.

$A' \leftarrow []$ ▷ *initialize empty array*
$\mathsf{T}_{A'} \leftarrow 0^m$ ▷ *initialize empty vector*
▷ *Place the first two points of A'.*
Append(A', a'_1) ▷ *add the first point of A'*
$z \leftarrow \mathsf{L}_A[1] t'_1$ ▷ *form distance & direction for geodesic tracing*
$g'_1 \leftarrow$ TraceGeodesic(a'_1, z)
$a'_2 \leftarrow$ EndingPoint(g'_1)
Append(A', a'_2) ▷ *add the second point of A'*
$\mathsf{T}_{A'}[1] \leftarrow t'_1$ ▷ *record axis tangent*
▷ *Record ending direction of g'_1 at a_2*
$s_i \leftarrow$ EndingDirection(g'_1)
▷ *Trace the rest of A'.*
for $i = 2$ to $m - 1$ **do**
  ▷ *Negate turning angle to get mirror image of A*
  $\phi'_i \leftarrow \overline{\Phi_A[i]}$
  ▷ *Form distance & direction for geodesic tracing*
  $z \leftarrow \mathsf{L}_A[i] * s_i * \phi'_i$
  $g'_i \leftarrow$ TraceGeodesic$(A'[i], z)$
  $a'_{i+1} \leftarrow$ EndingPoint(g'_i)
  Append(A', a'_{i+1}) ▷ *add the next point of A'*
  $\mathsf{T}_{A'}[i] \leftarrow$ InitialDirection(g'_i) ▷ *record axis tangent*

22: ▷*Record ending direction of g_i at a_{i+1}*
23: $s_i \leftarrow \text{EndingDirection}(g'_i)$
24: **return** $A', \mathsf{T}_{A'}$

Algorithm 4 ReconstructPatch$(A', \mathsf{T}_{A'}, P, \mathsf{E}_P^A)$

Input: A set $P = \{p_1, \ldots, p_n\}$ of barycentric points on some triangle mesh M_1, and its parameterization E_P^A with respect to an axis A; a transferred axis A' corresponding to A, and its tangents $\mathsf{T}_{A'}$ on some target triangle mesh M_2.
Output: A set $P' = \{p'_1, \ldots, p'_n\}$ of barycentric points on the target triangle mesh M_2.

1: $P' \leftarrow \{\}$ ▷*initialize empty set*
2: **for** $p_i \in P$ **do**
3: $(j, z_i) \leftarrow \mathsf{E}_P^A[p_i]$ ▷*get info from patch parameterization*
4: $t_j \leftarrow \mathsf{T}_{A'}[j]$ ▷*get tangent direction at a'_j*
5: $z_j \leftarrow \overline{z_j}$ ▷*negate angle so P' is the mirror image of P*
6: $z \leftarrow \overline{z_i}/t_j$ ▷*take direction relative to tangent direction of A'*
7: $p'_i \leftarrow \text{TraceGeodesic}(A'[j], z)$ ▷*trace geodesic on M_2*
8: $\text{Insert}(P', p'_i)$ ▷*record where the geodesic ended as p'_i*
9: **return** P'

Algorithm 5 TransferPatch(P, A, a'_1, t'_1)

Input: A set $P = \{p_1, \ldots, p_n\}$ of barycentric points on some triangle mesh M_1 representing the source patch; an array $A = [a_1, \ldots, a_m]$ of barycentric points also on M_1 representing a user-drawn axis; an initial point a'_1 on some target mesh M_2; an initial direction t'_1 as a unit-magnitude complex number.
Output: A set $P' = \{p'_1, \ldots, p'_n\}$ of barycentric points on M_2, representing the mirror image of an approximately-isometric embedding of the source patch into M_2.

1: $(\Phi_A, \mathsf{L}_A, \mathsf{T}_A) \leftarrow \text{ParameterizeAxis}(A)$
2: $\mathsf{E}_P^A \leftarrow \text{ParameterizePatch}(P, A, \mathsf{T}_A)$
3: $A' \leftarrow \text{ReconstructAxis}(\Phi_A, \mathsf{L}_A, a'_1, t'_1)$ ▷*transfer axis to M_2*
4: $P' \leftarrow \text{ReconstructPatch}(A', \mathsf{T}_{A'}, \mathsf{E}_P^A)$ ▷*transfer patch to M_2*
5: **return** P'

B ADDITIONAL USER STUDY METRICS

Table 3 tabulates the usage metrics of our plugin by subject. Times reported only account for "useful" work done by the subject in progression towards a final pose result. Default tooling usage is largely comprised of adjusting hand joints - time spent adjusting object transforms are generally negligible. Time spent reading documentation, asking questions, drafting intermediate poses, and any other "stall" events are omitted. Subjects averaged between 4.625 minutes and 17.667 minutes per pose, with Subject 3 being the fastest and 2 the slowest. All subjects spent more time using the plugin than existing tooling, albeit for different reasons (e.g. experimentation of capabilities, fine grained contact adjustment, creating and destroying contacts at different locations, etc.).

Table 3. More detailed breakdowns of usage times of our plugin vs. existing tooling per subject

Plugin Usage Results		
Subject #	Usage Time (Plugin / Default Tooling)	Poses Created (Plugin Only / Default Tooling Only / Both)
1	29 mins / 13 mins	0 / 0 / 3
2	39 mins / 14 mins	0 / 0 / 3
3	20 mins / 17 mins	0 / 2 / 6
4	22 mins / 8 mins	1 / 0 / 2
5	28 mins / 0 mins	3 / 0 / 0

We examine subject 3 for an approximate tooling efficiency comparison. Subject 3 largely used the tool as a "coarse solver" by exploiting a recurring set of contacts. The subject avoided creating new contacts per pose, and instead kept returning to a saved default set. The subject transferred contacts to the object, computed a coarse pose, and then used manual tooling to tweak the coarse estimate into the final pose. The subject required approximately an equal amount of time creating each pose using both our plugin in combination with existing FK posing tools and existing posing tools only; however, we found the plugin-augmented poses to be substantially higher quality. Therefore, we argue that our tool in its current state can be considered valuable not necessarily for raw throughput speed, but rather, as validated by qualitative assessment results, for its impact on quality.

C SUBJECT-GENERATED RESULTS

The remainder of this document contains images of poses generated by user study subjects and distributed online during qualitative assessments. Corresponding asterisk colors indicate paired A/B images in the qualitative assessment survey. Survey participants were asked to select between the two paired images in each category outlined in Section 7.2. An example question prompt is illustrated in Figure 21.

Fig. 19. Subject results selected for distribution in qualitative studies using our plugin in a substantial capacity.

Fig. 20. Comparison pose results distributed in the qualitative studies survey generated using fully manual FK tooling.

Fig. 21. Example qualitative assessment survey question, where the highlighted red box indicates subject selection.