PDEs Wrapup
AND
Special Topic in Animation

Computer Graphics
CMU 15-462/15-662

Model Equations

- **Fundamental behavior of many important PDEs is well-captured by three model linear equations:**

Solve numerically?

"Laplacian" (more later!)

LAPLACE EQUATION ("ELLIPTIC") $\Delta u = 0$

"what's the smoothest function interpolating the given boundary data"

HEAT EQUATION ("PARABOLIC") $\dot{u} = \Delta u$

"how does an initial distribution of heat spread out over time?"

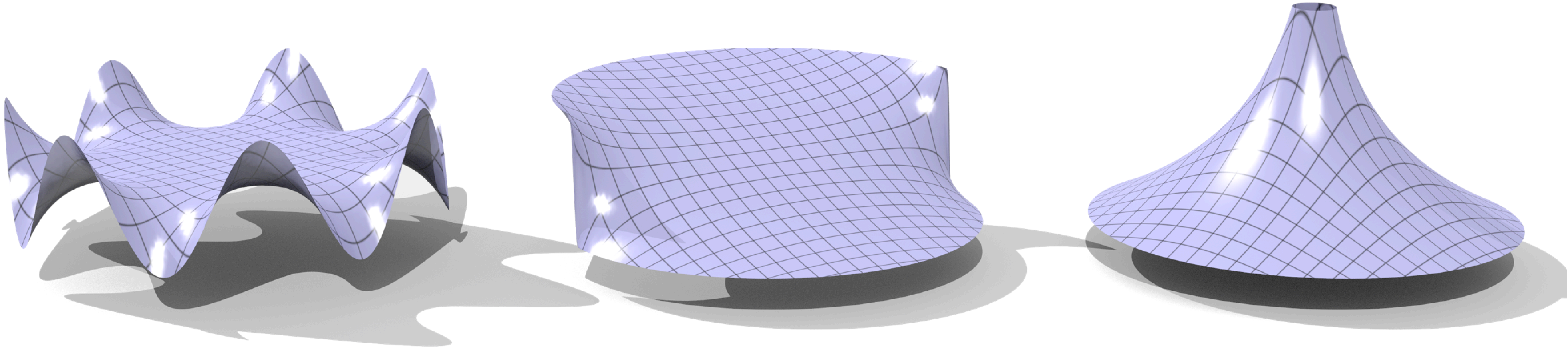
WAVE EQUATION ("HYPERBOLIC") $\ddot{u} = \Delta u$

"if you throw a rock into a pond, how does the wavefront evolve over time?"

[NONLINEAR + HYPERBOLIC + HIGH-ORDER]

Elliptic PDEs / Laplace Equation

- "What's the smoothest function interpolating the given boundary data?"



- Conceptually: each value is at the average of its "neighbors"
- Roughly speaking, why is it easier to solve?
- Very robust to errors: just keep averaging with neighbors!

Image from Solomon, Crane, Vouga, "Laplace-Beltrami: The Swiss Army Knife of Geometry Processing"

Numerically Solving the Laplace Equation

- **Want to solve $\Delta u = 0$**
- **Plug in one of our discretizations, e.g.,**

	$u_{i,j+1}$	
$u_{i-1,j}$	$u_{i,j}$	$u_{i+1,j}$
	$u_{i,j-1}$	

$$\frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} = 0$$

$$\iff u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right)$$

- **If u is a solution, then each value must be the average of the neighboring values (u is a "harmonic function")**
- **How do we solve this?**
- **One idea: keep averaging with neighbors! ("Jacobi method")**
- **Correct, but slow. Much better to use modern linear solver**

Aside: PDEs and Linear Equations

- How can we turn our Laplace equation into a linear solve?
- Have a bunch of equations of the form

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = 0$$

- On a 4x4 grid, assign each cell $u_{i,j}$ a unique index 1, …, 16
- Can then write equations as a 16x16 matrix equation*

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$$\begin{bmatrix}
-4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & -4
\end{bmatrix}
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \\ u_{16} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Compute solution by calling <u>sparse</u> linear solver (SuiteSparse, Eigen, …)
- Q: By the way, what's wrong with our problem setup here? :-)

*assuming neighbors wrap around left/right and top/bottom

Solving the Heat Equation

- **Back to our three model equations, want to solve heat eqn.**

$$\dot{u} = \Delta u$$

- **Just saw how to discretize Laplacian**
- **Also know how to do time (forward Euler, backward Euler, ...)**
- **E.g., forward Euler:**

$$u^{k+1} = u^k + \tau \Delta u^k$$

- **Q: On a grid, what's our overall update now at $u_{i,j}$?**

$$u_{i,j}^{k+1} = u^k + \tfrac{\tau}{h^2}(4u_{i,j}^k - u_{i+1,j}^k - u_{i-1,j}^k - u_{i,j+1}^k - u_{i,j-1}^k)$$

- **Not hard to implement! Loop over grid, add up some neighbors.**

Solving the Wave Equation

- **Finally, wave equation:**

$$\ddot{u} = \Delta u$$

- **Not much different; now have 2nd derivative in time**
- **By now we've learned two different techniques:**
  - **Convert to two 1st order (in time) equations:**

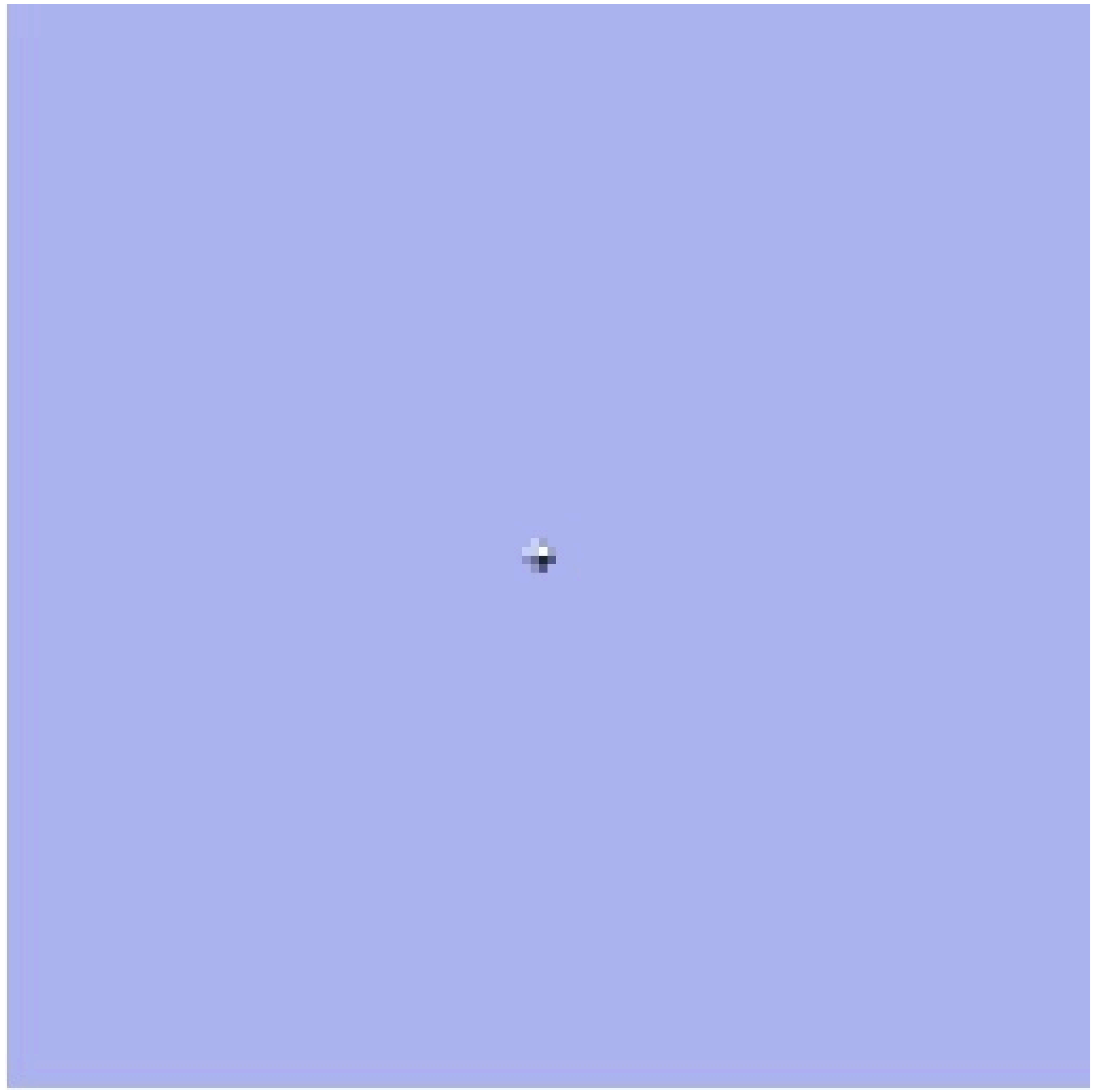
$$\dot{u} = v, \quad \dot{v} = \Delta u$$

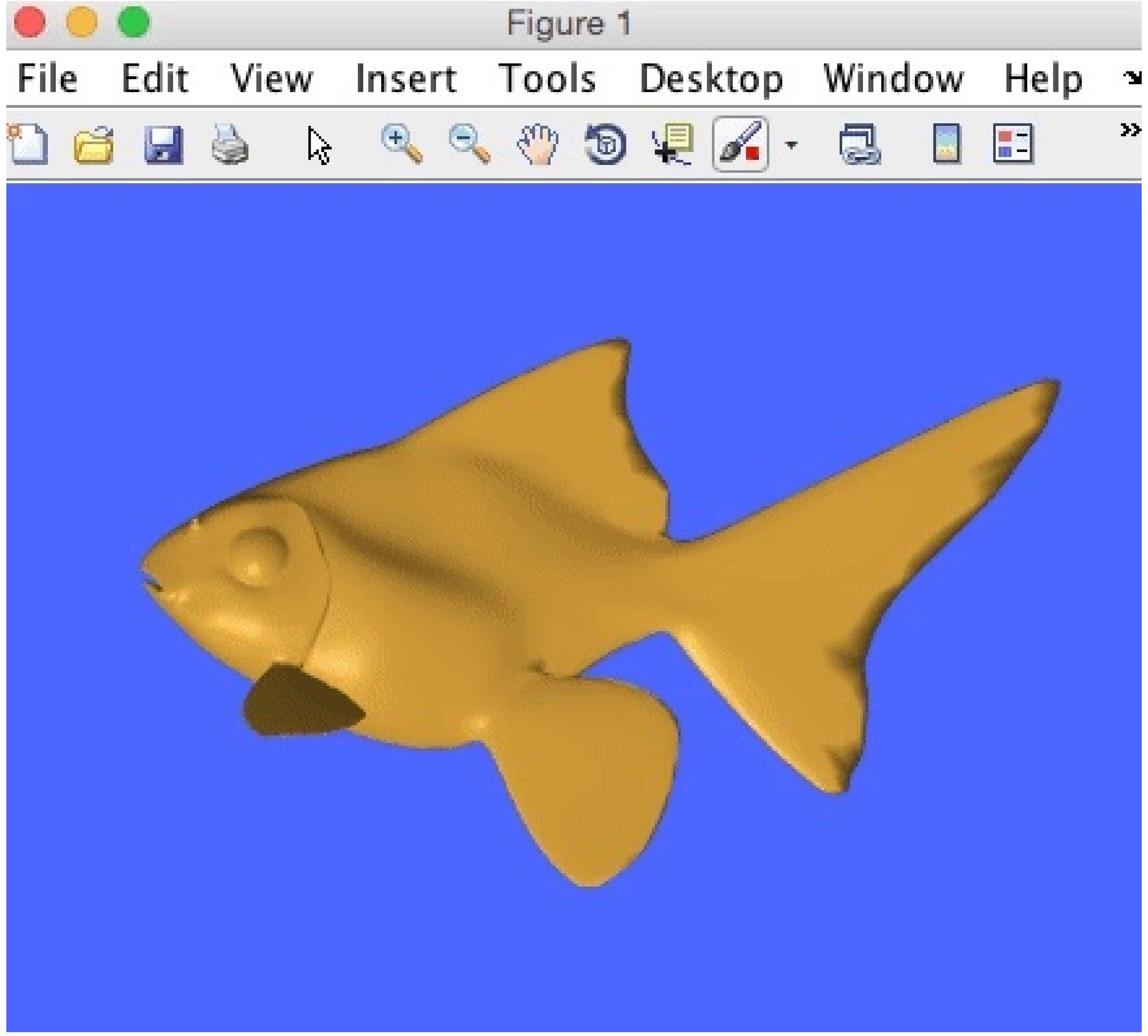
  - **Or, use centered difference (like Laplace) in time:**

$$\frac{u^{k+1} - 2u^k + u^{k-1}}{\tau^2} = \Delta u^k$$

- **Plus all our choices about how to discretize Laplacian.**
- **So many choices! And many, many (many) more we didn't discuss.**

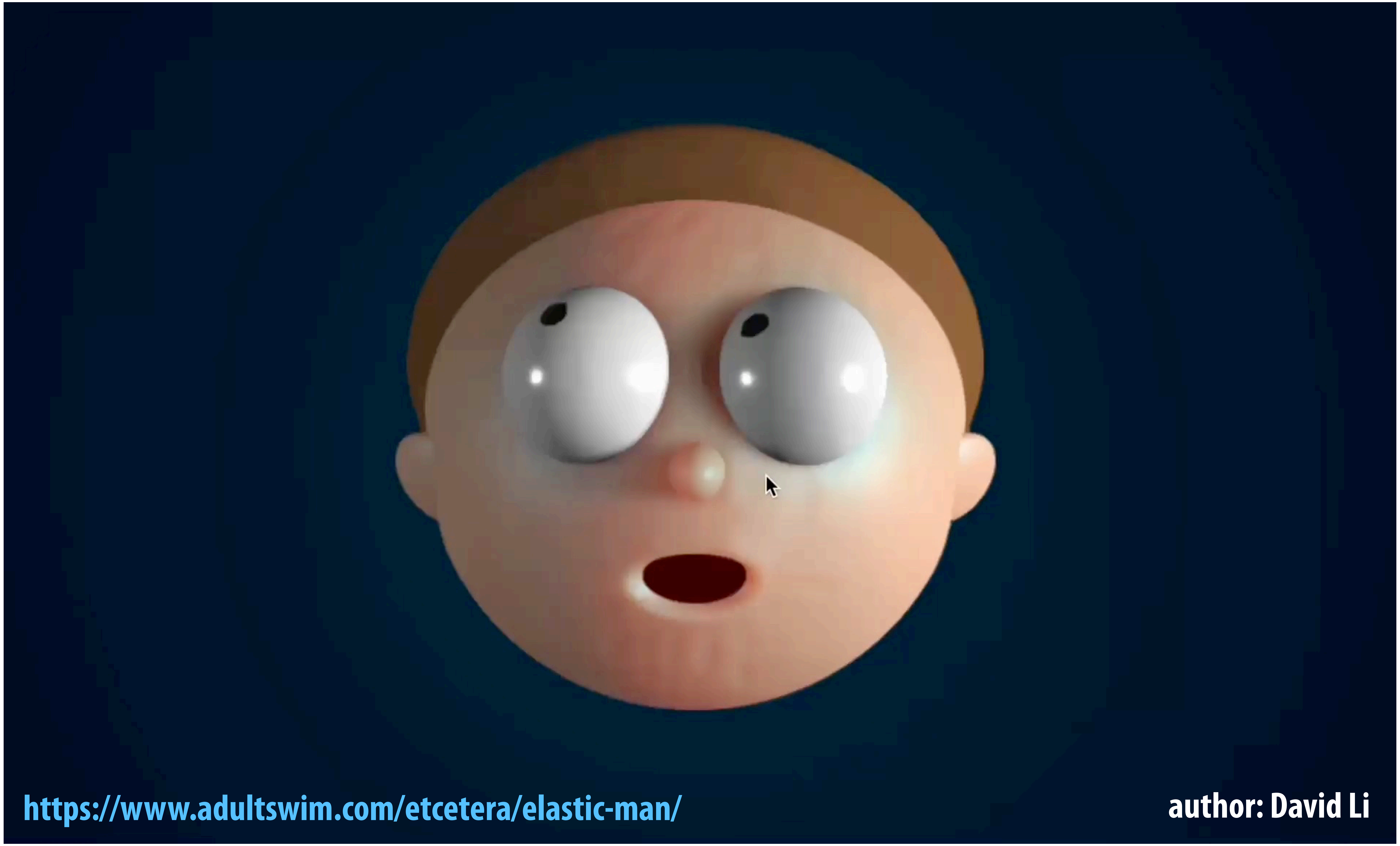
Wave Equation on a Grid, Triangle Mesh





Fish credit: Alec Jacobson (http://www.alecjacobson.com/weblog/?p=4363)

Fun with wave-like equations…



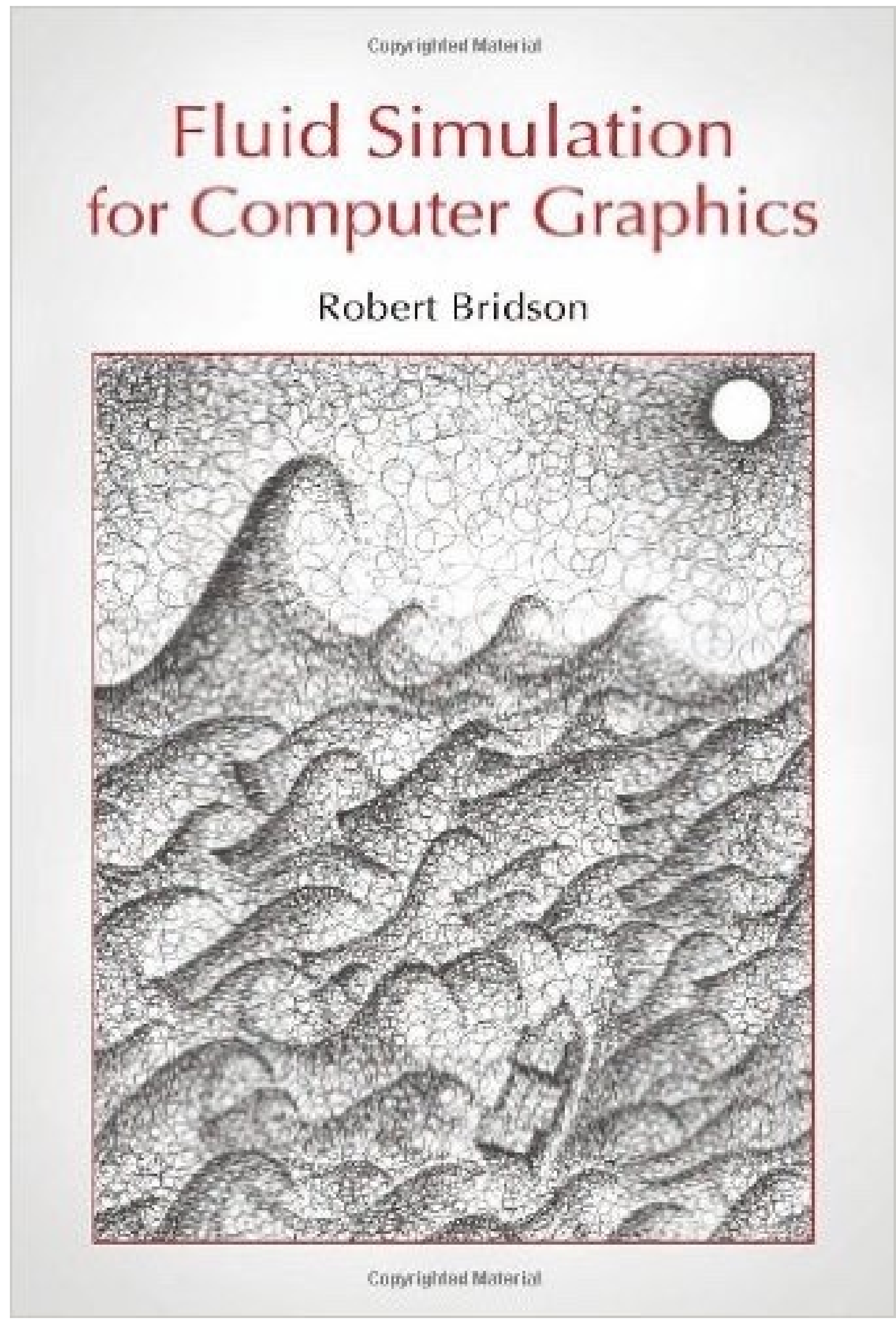
Technique: low-res thin shell simulation (via "position-based dynamics") + Loop subdivision

Wait, what about all that other cool stuff?
(Fluids, hair, cloth, ...)

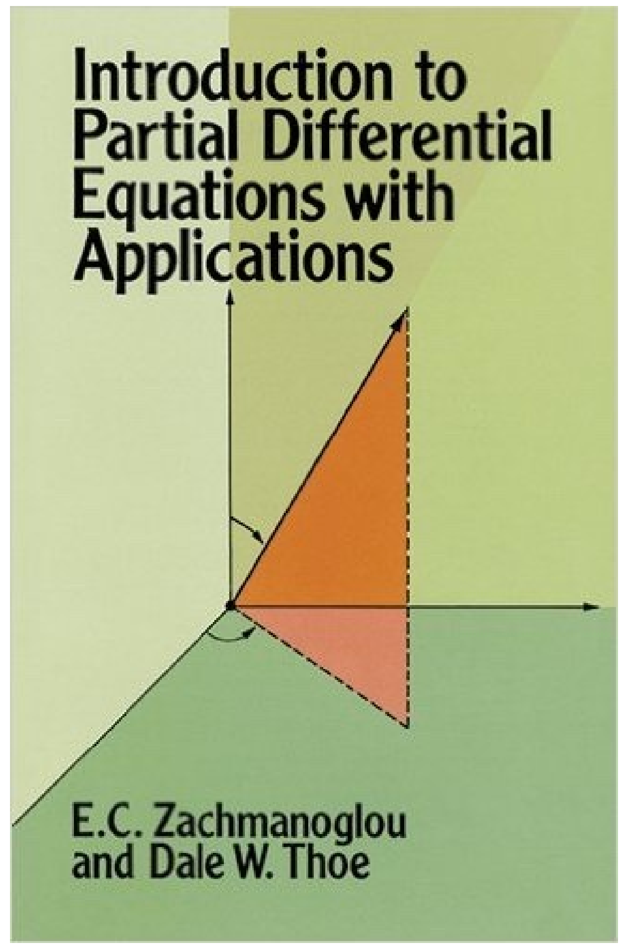
Want to Know More?

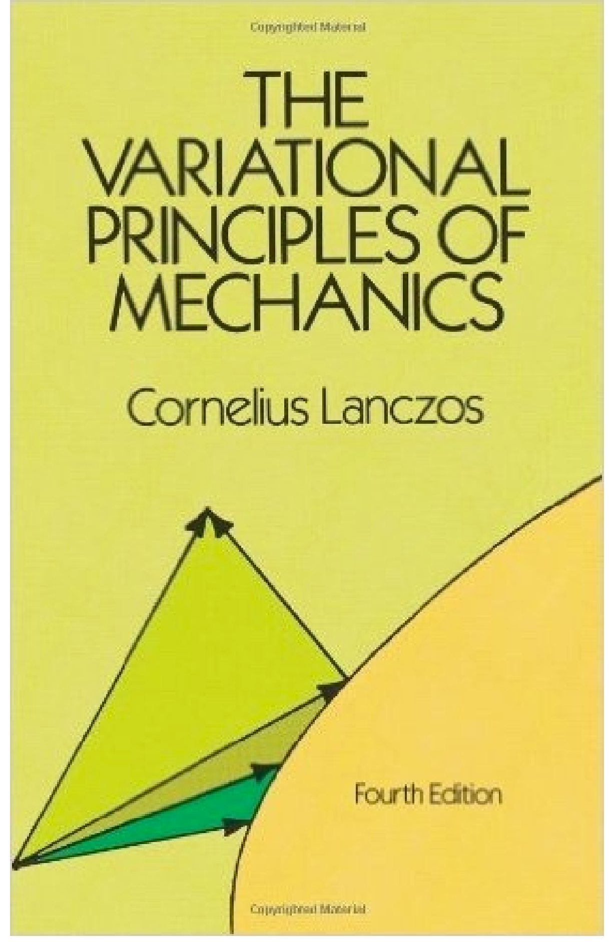
- There are some good books:
- And papers:

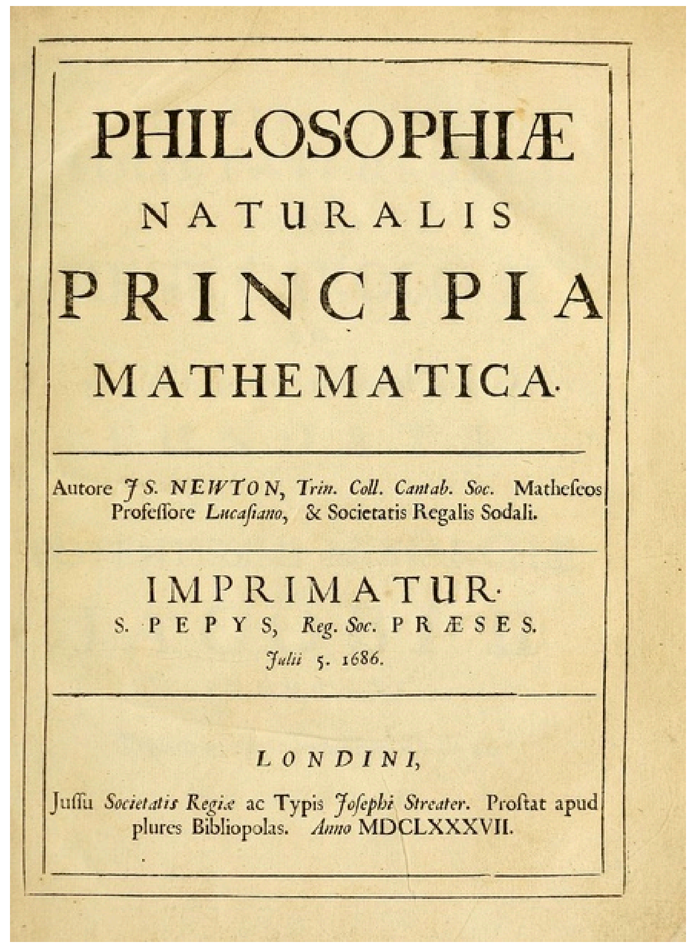
http://www.physicsbasedanimation.com/



- Also, what did the folks who wrote these books & papers read?







And that is the end of the official course material on PDEs!

(but watch for a guest appearance of the Heat Equation in the next section :)

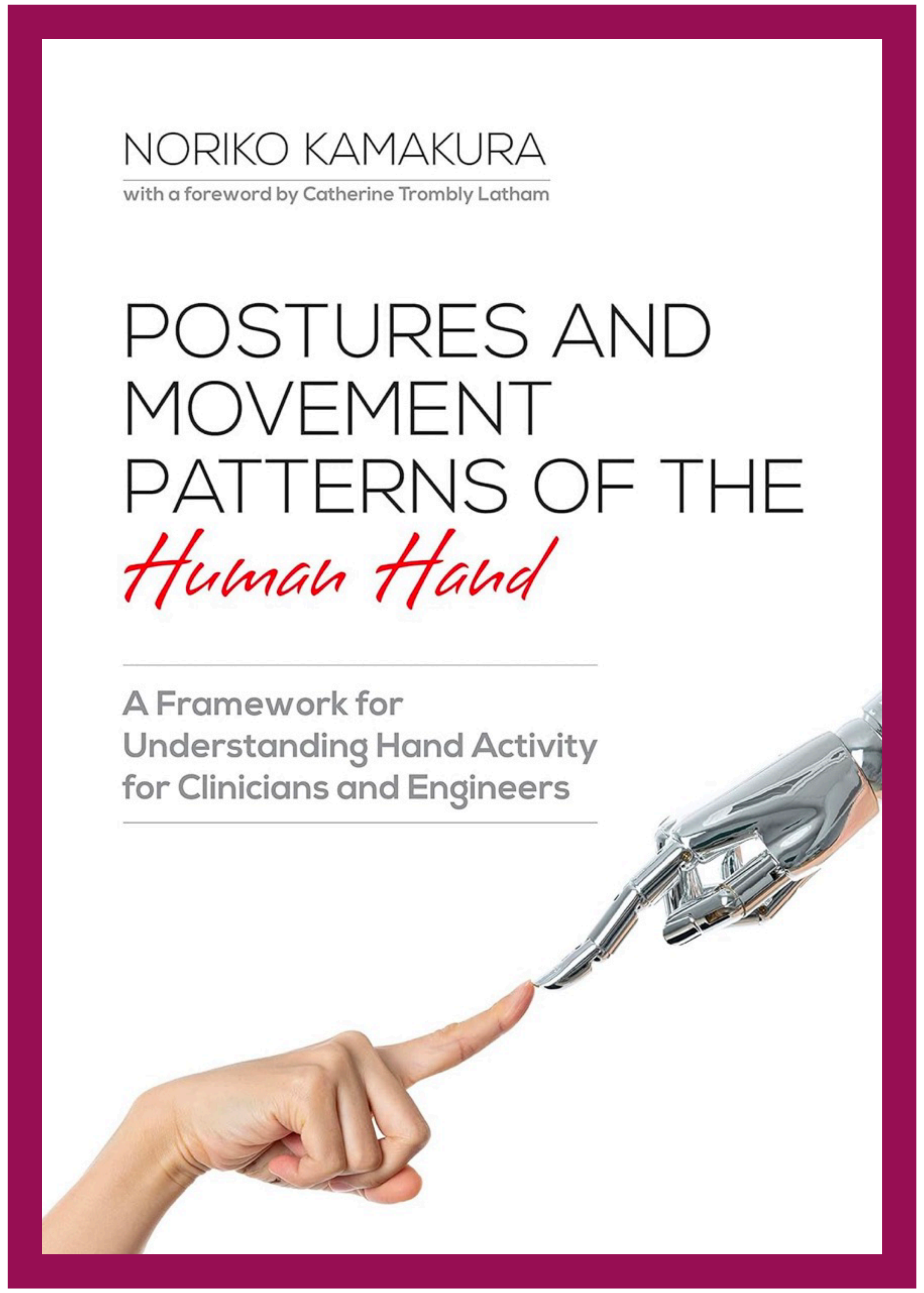
Contact Edit: Artist Tools for Intuitive Modeling of Hand-Object Interactions

Arjun S. Lakshmipathy, Nicole Feng, Yu Xi Lee, Moshe Mahler, Nancy S. Pollard

Carnegie Mellon University


input
Contact Edit
final pose

Area Contacts

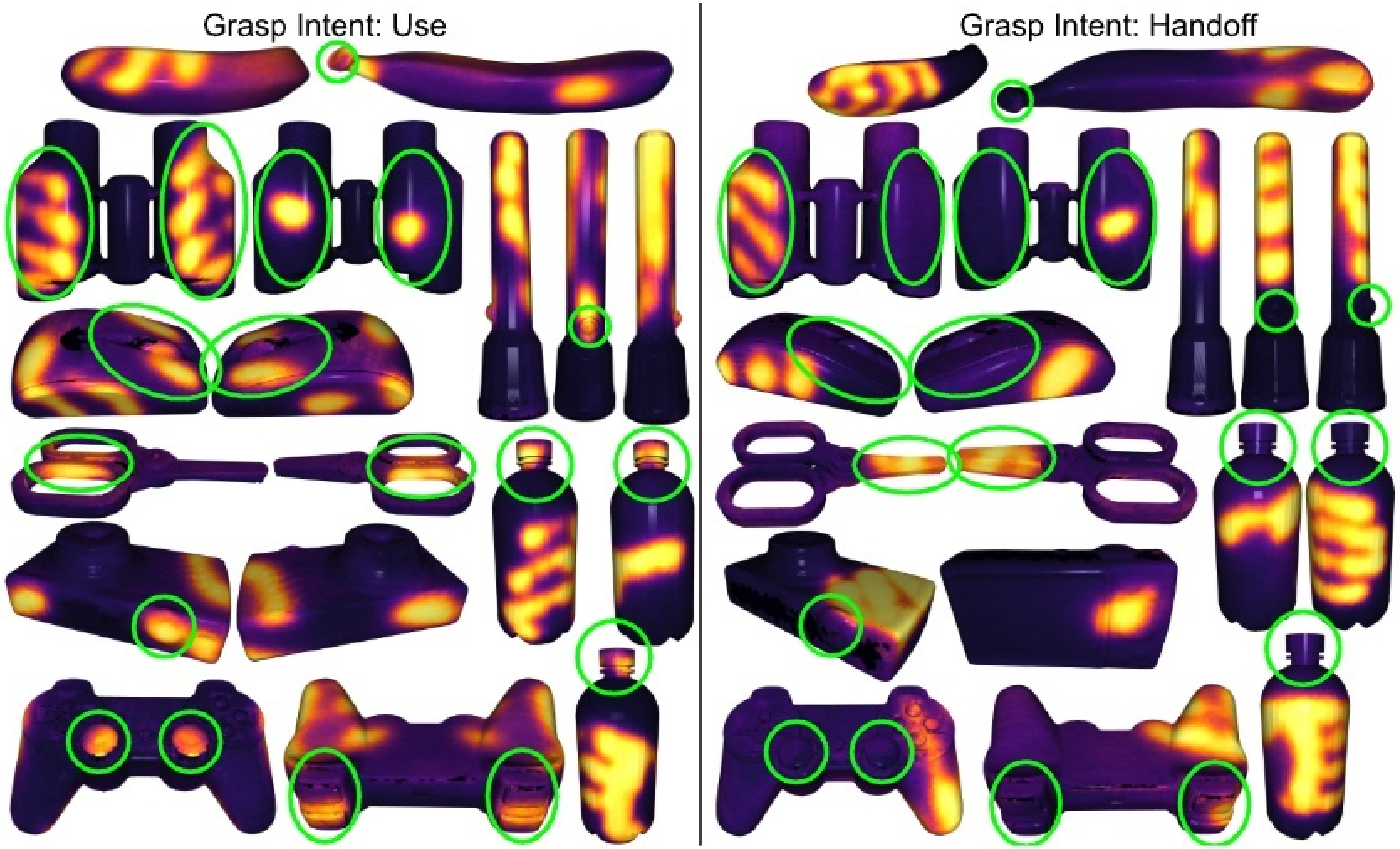


Kamakura, Noriko. *Postures and Movement Patterns of the Human Hand: A Framework for Understanding Hand Activity for Clinicians and Engineers*. Universal-Publishers, 2022.

Kamakura N, Matsuo M, Ishii H, Mitsuboshi F, Miura Y. Patterns of static prehension in normal hands. American Journal of Occupational Therapy. 1980

Existing Work on Hands and Contacts

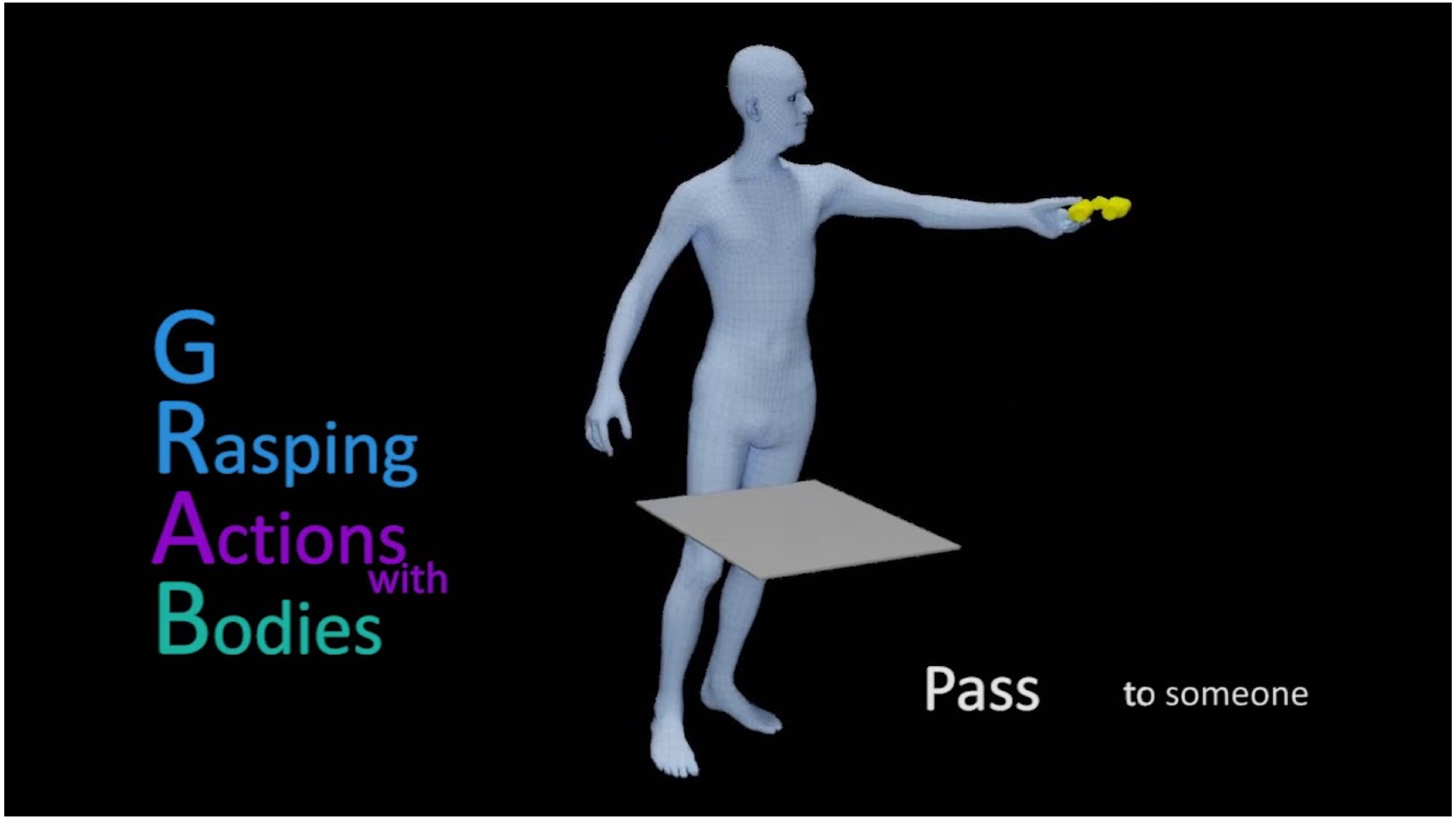
ContactDB



https://mlatgt.blog/2019/06/06/contactdb-analyzing-and-predicting-grasp-contact-via-thermal-imaging/

Existing Work on Hands and Contacts

GRAB



https://grab.is.tue.mpg.de/

Existing Work on Hands and Contacts

ARCTIC

ARCTIC

A Dataset for Dexterous Bimanual Hand-Object Manipulation

Zicong Fan[1,2], Omid Taheri[2], Dimitrios Tzionas[2], Muhammed Kocabas[1,2], Manuel Kaufmann[1], Michael J. Black[2], Otmar Hilliges[1]

[1]ETH Zürich, Switzerland
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany

arctic.is.tue.mpg.de

Advanced Interactive Technologies

ETH zürich

MAX PLANCK INSTITUTE FOR INTELLIGENT SYSTEMS

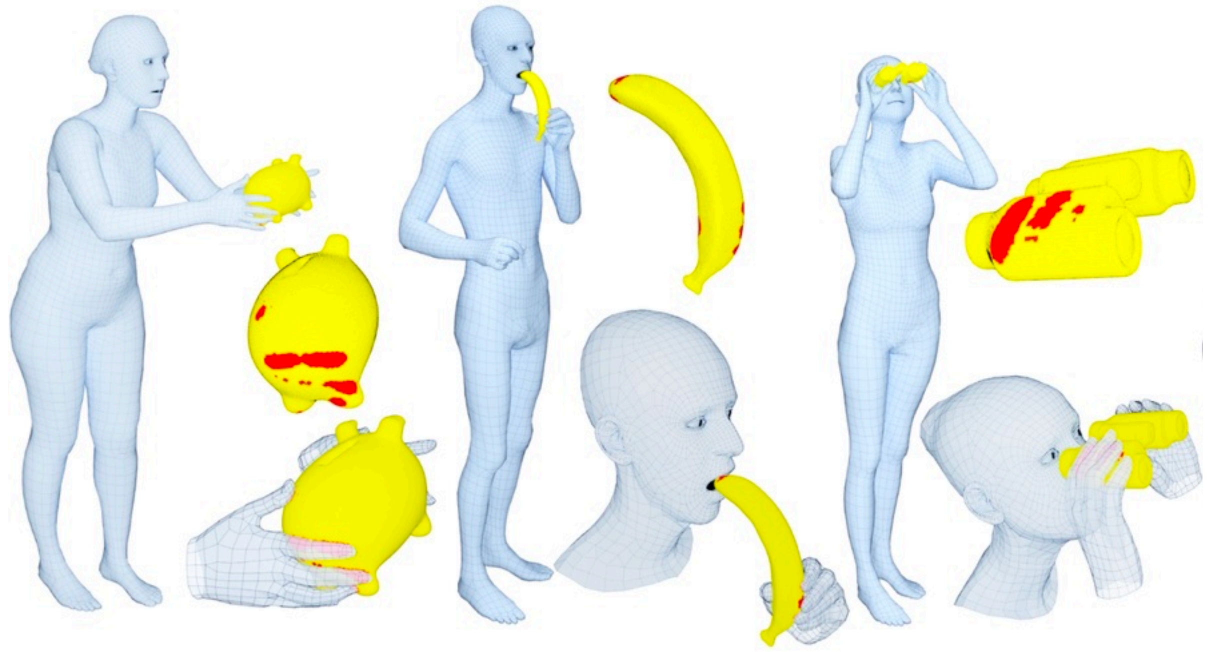
https://arctic.is.tue.mpg.de/

Existing Work on Hands and Contacts

ContactDB

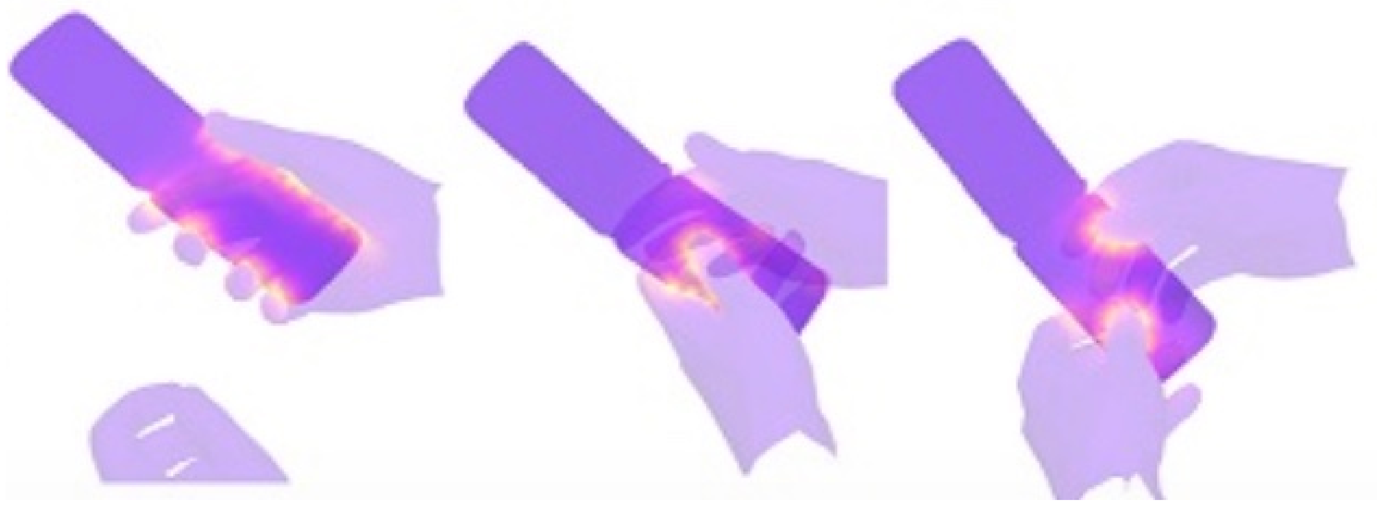
[Brahmbhatt et. al, 2019]

GRAB



[Taheri et. al, 2020]

ARTIC



[Fan et. al, 2023]

ContactGrasp

[Brahmbhatt et. al, 2019]

Grasp’d

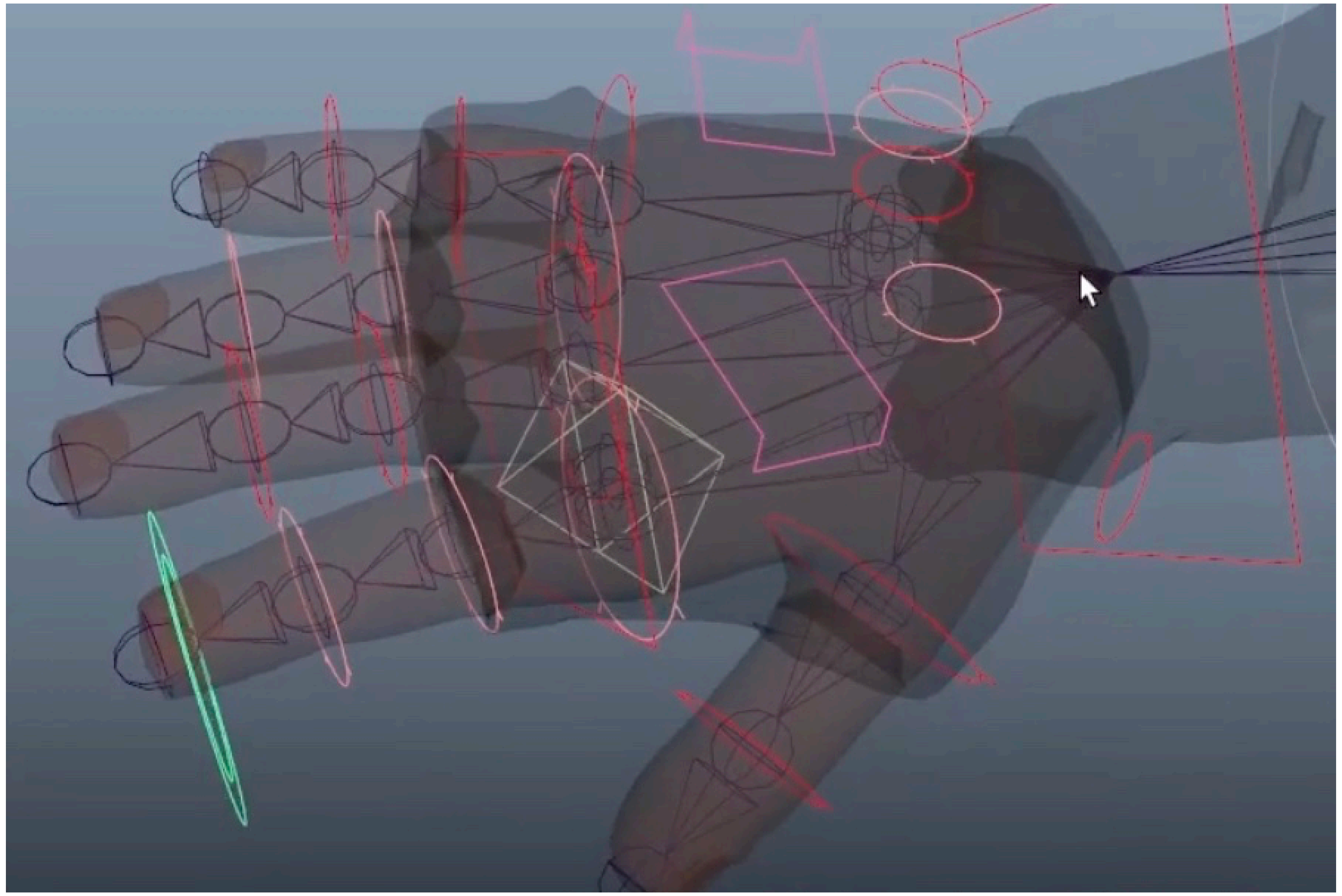
[Turpin et. al, 2022]

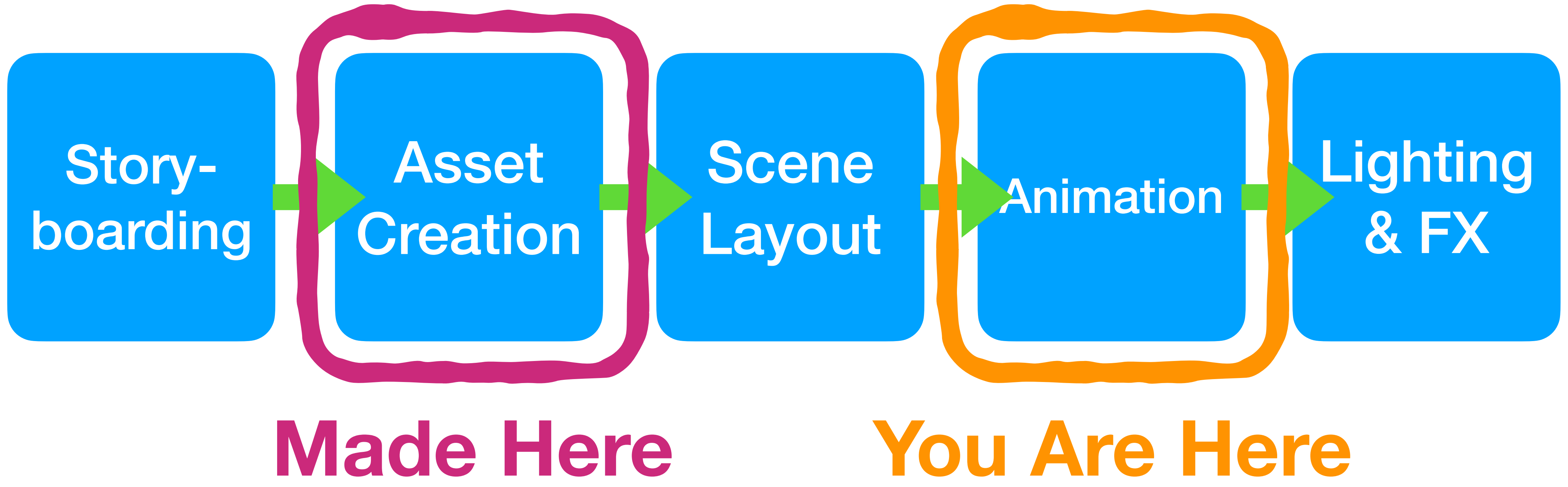
Challenges in Existing Posing Techniques

- Hierarchy-Induced Reconfigurations
- Gap Closure Difficulties
- Challenging for early-career animators

Common Solutions

Option 1: Make Complex Rig





Option 2: Inverse Kinematics

The
Requirement

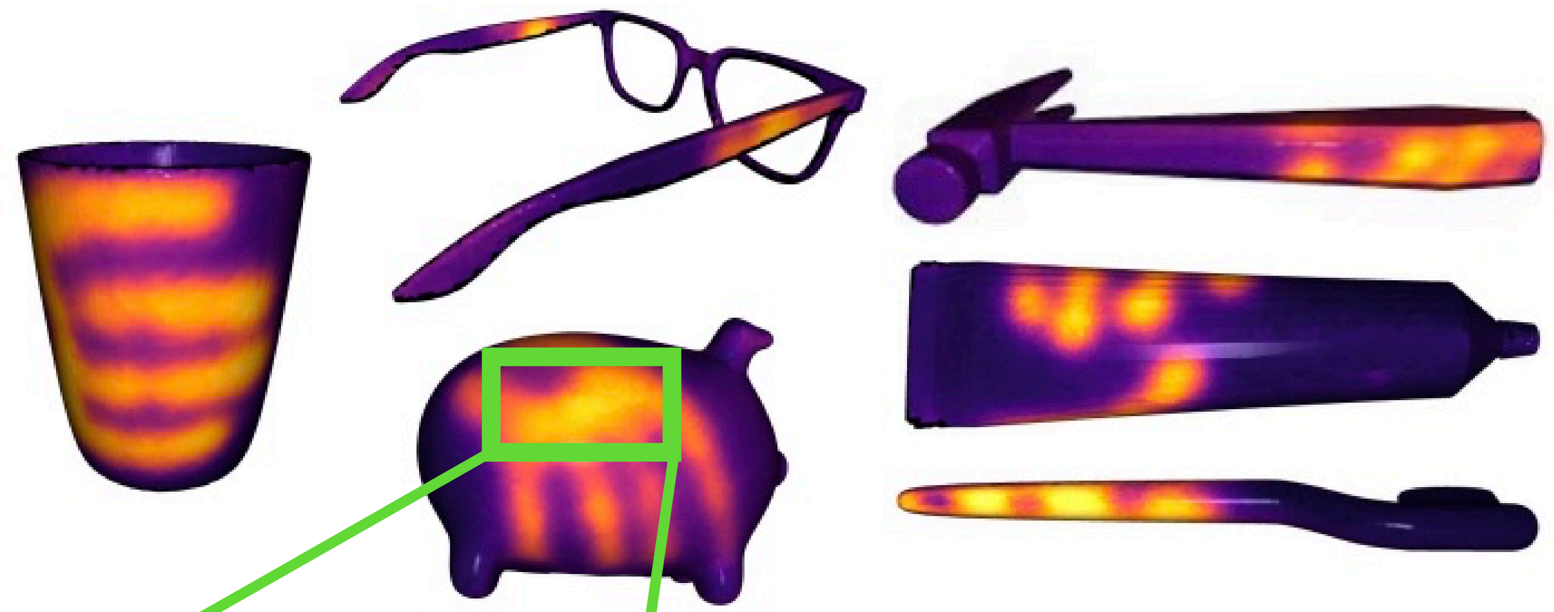
You

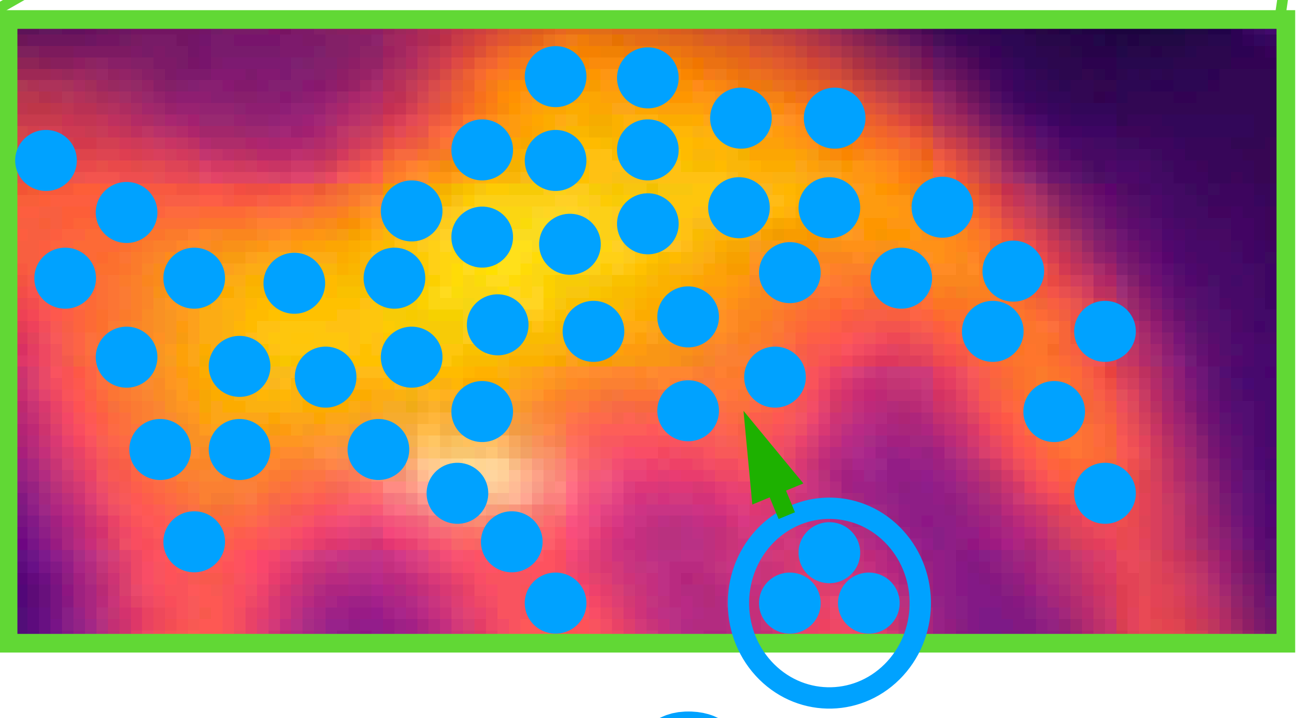
- We wanted to give artists tools to work with contact areas as an alternative design tool

Contact areas can improve IK

Understanding Contact Areas

[Brahmbhatt et. al, 2019]





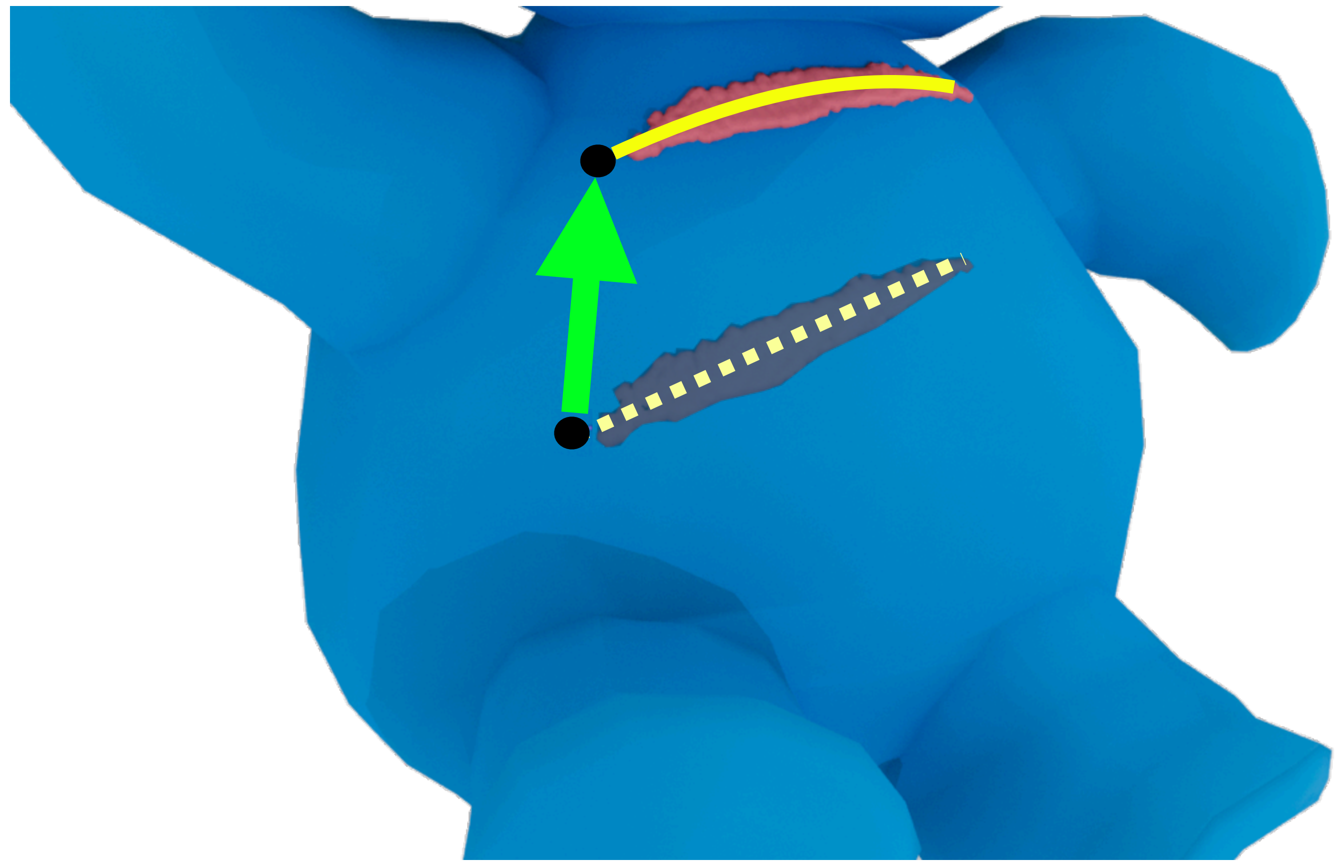
?

Realtime Performance

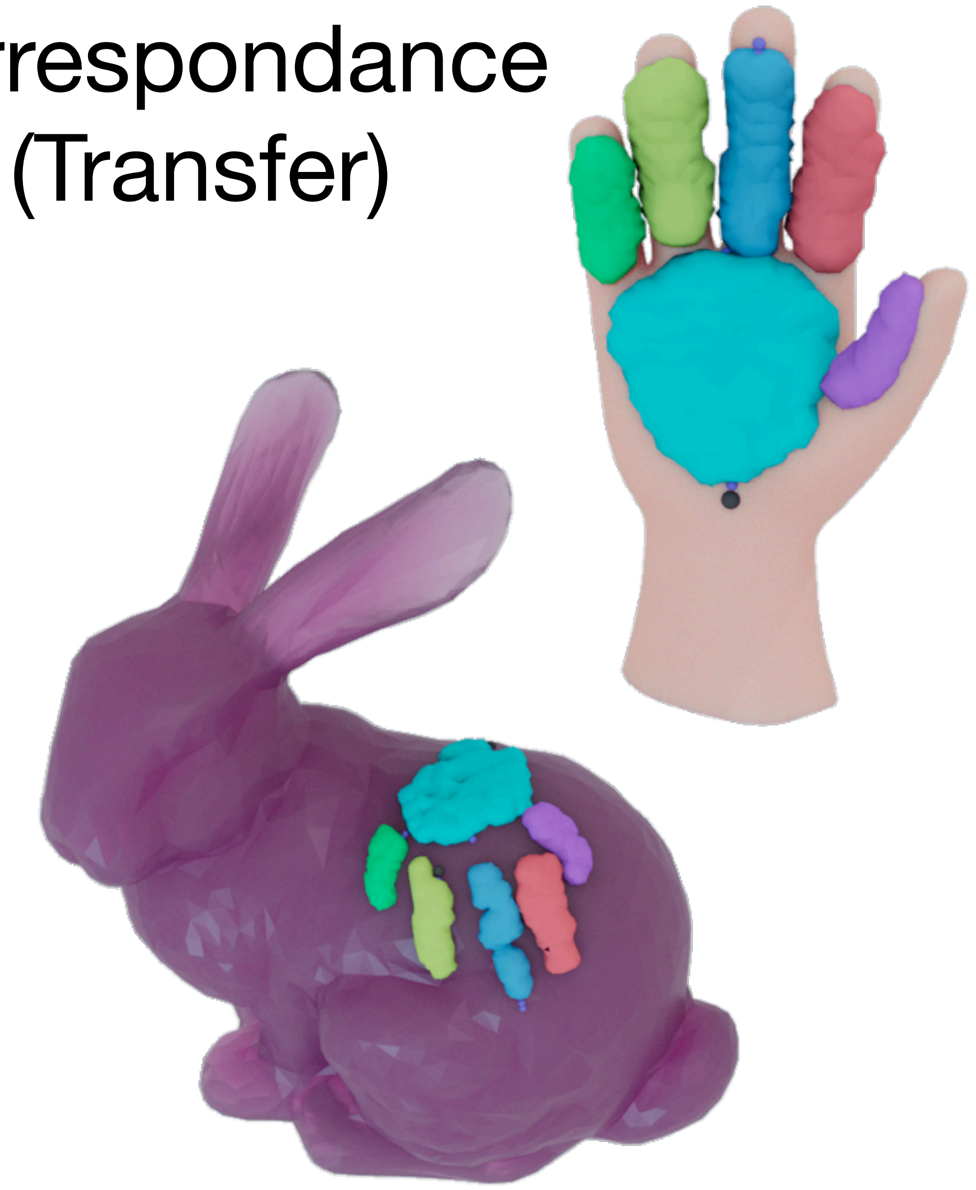
Predictable

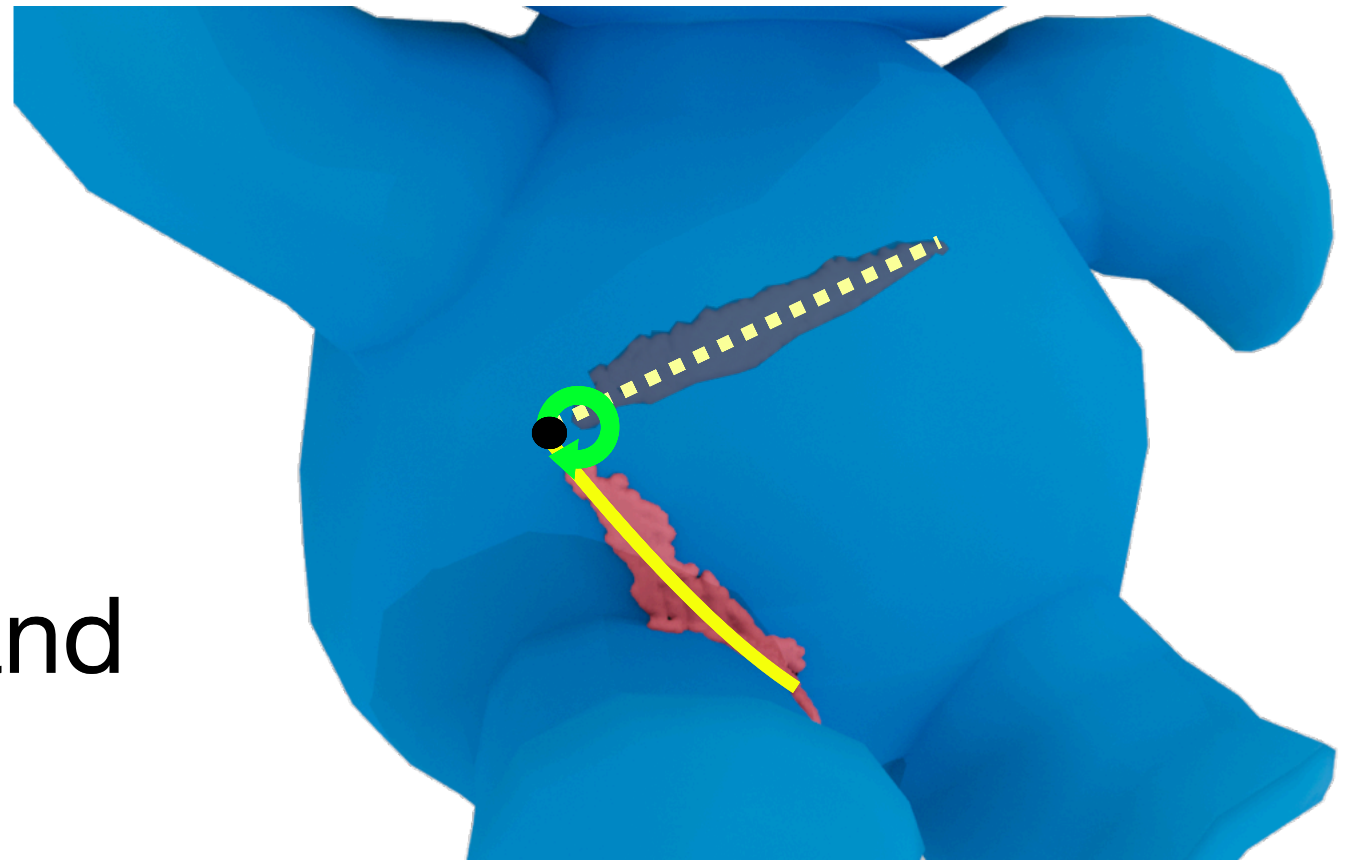
Easy to Understand

Relocation (Translation)

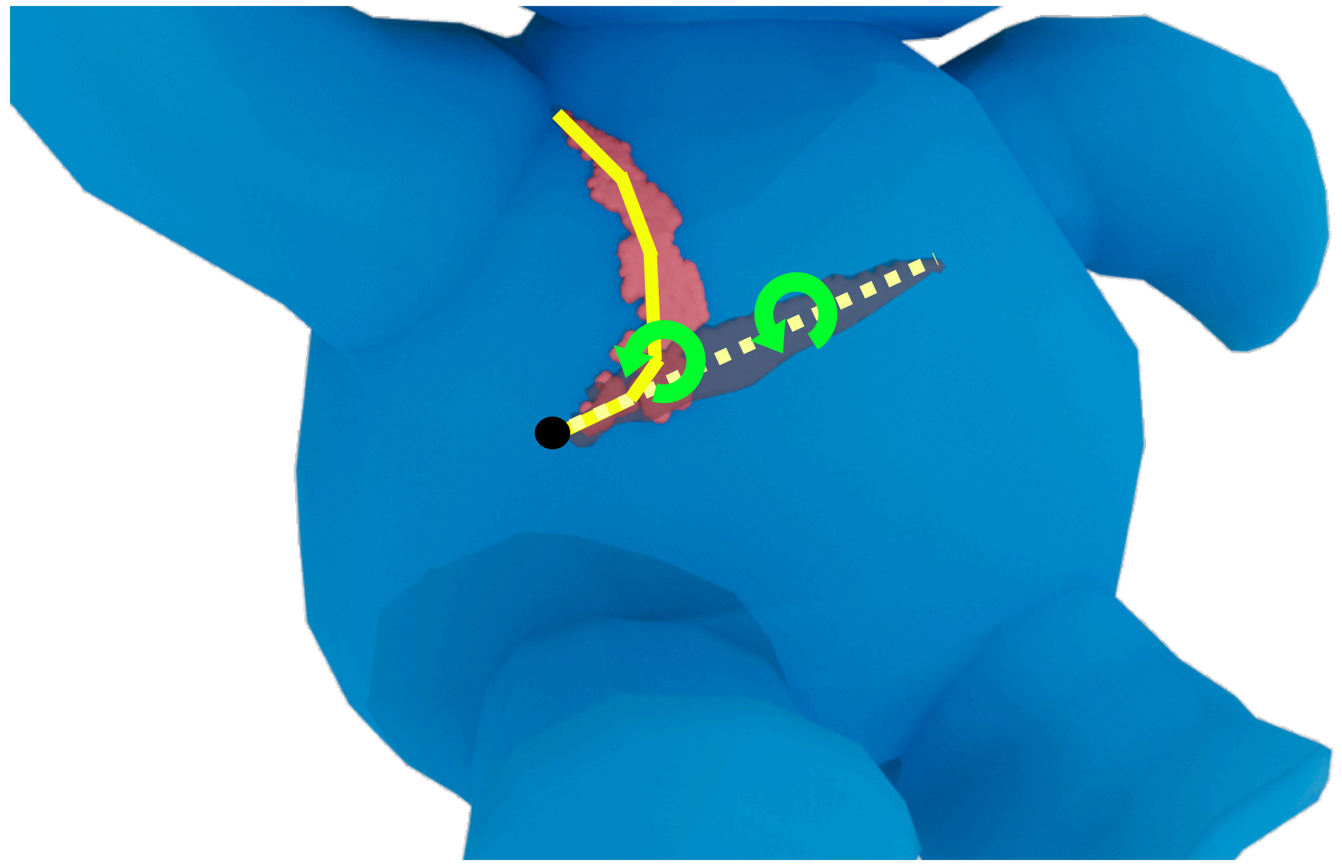


Correspondance (Transfer)



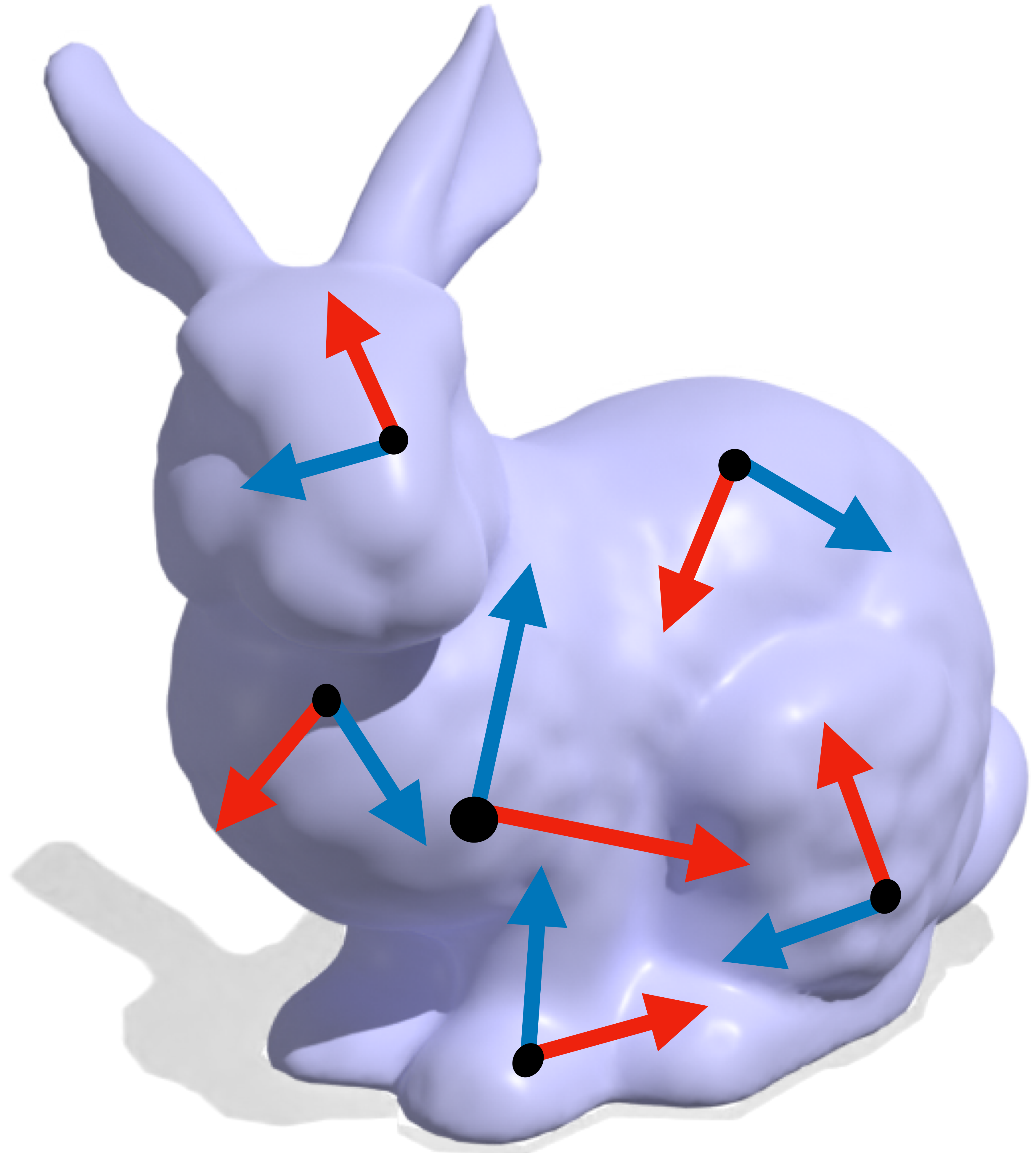


Reorientation (Rotation)

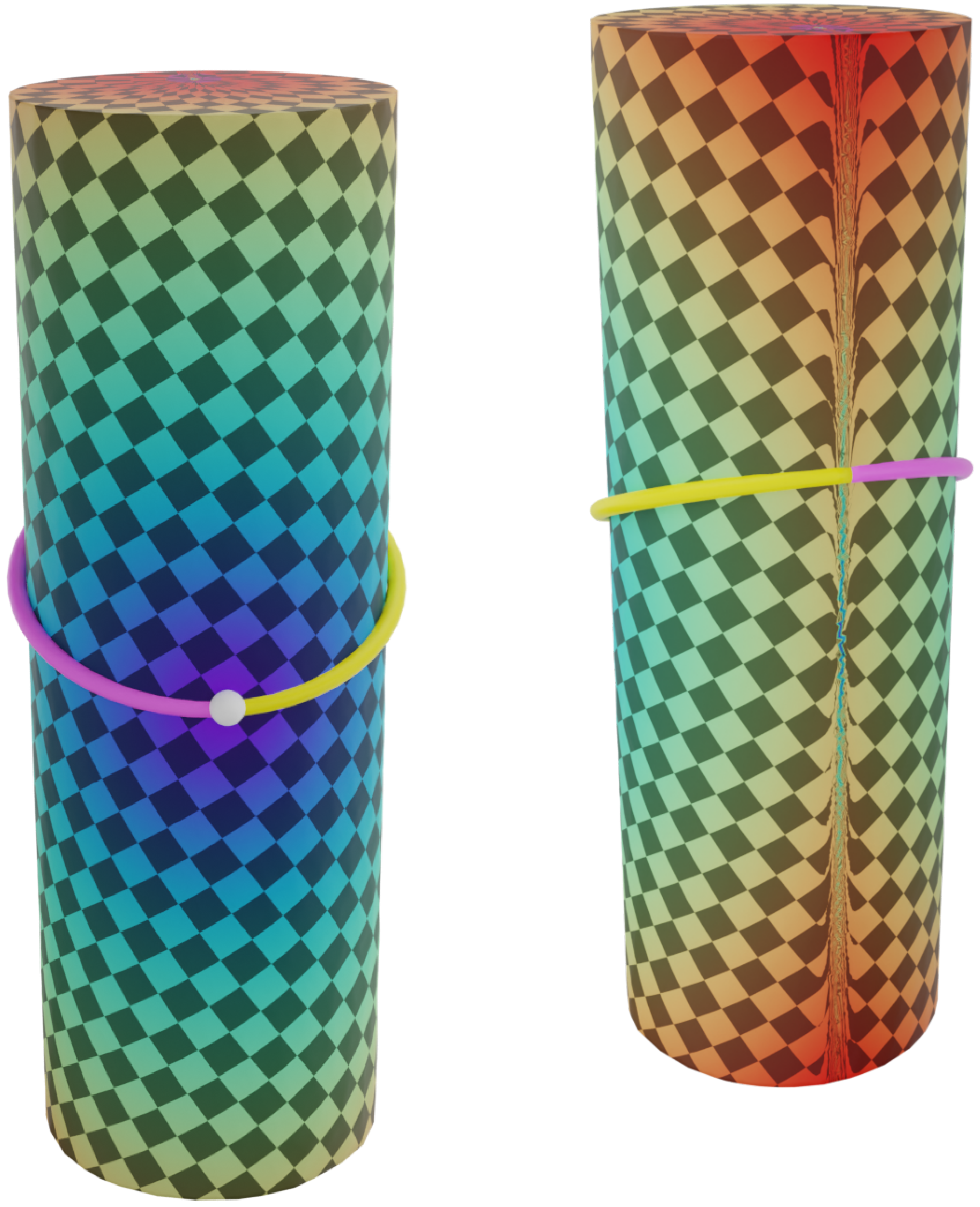


Bending (Deformation)

Complexities of Surfaces



No Global Coordinate System



Large Path Discontinuities

The Axis Model

$$exp_q(p) := T_q(r_p,\theta_p)$$

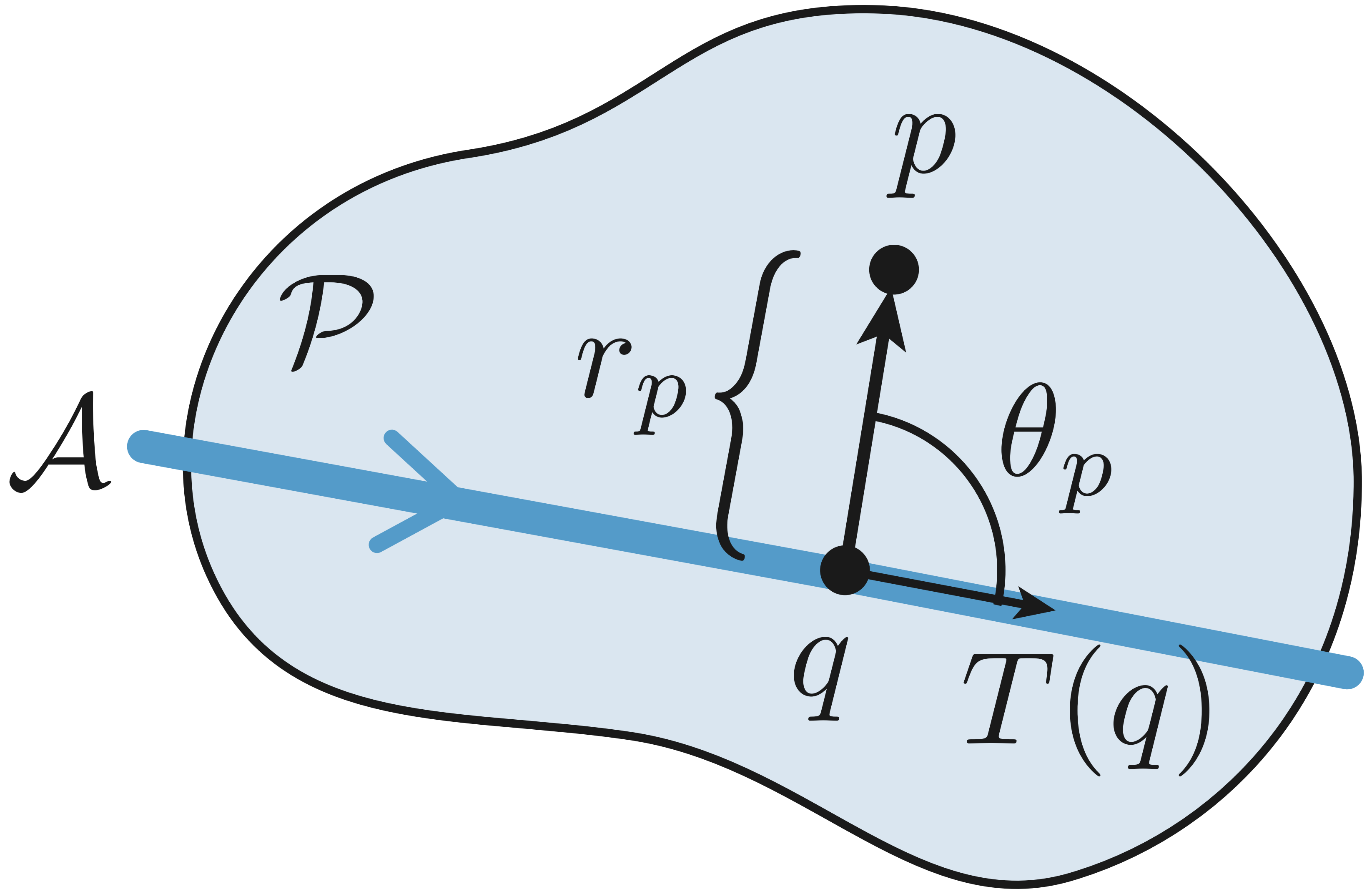
Exponential map
coordinates of p
in tangent basis of q

$$\{a_0,\dots,a_M\} \epsilon \mathcal{A}(xis)$$

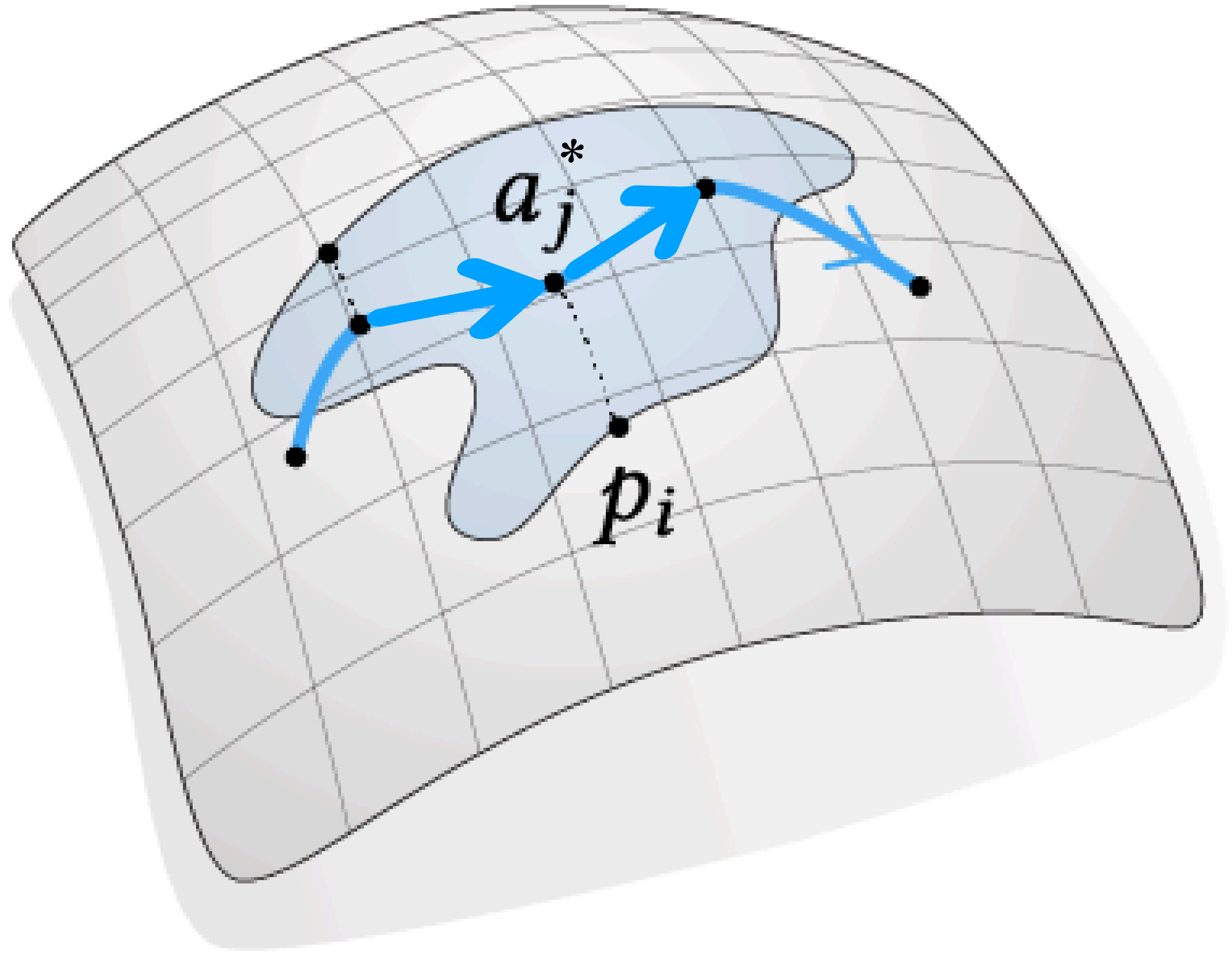
$$a_{j+1} = a_j + (d_j,\phi_j)$$

d_j := distance

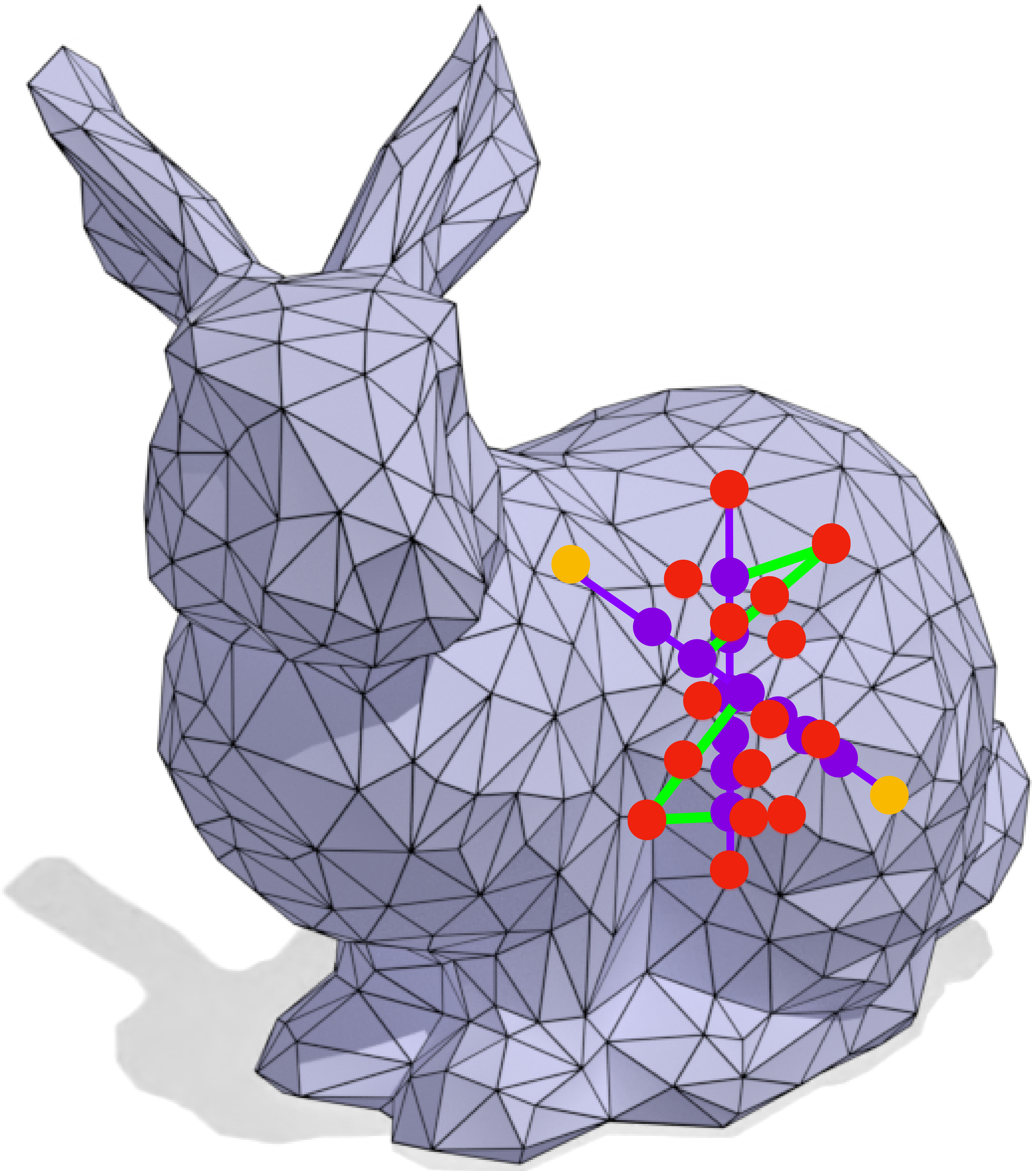
ϕ_j := turning angle



q = a_j^*



Contact Area Initialization



The Vector Heat Method

NICHOLAS SHARP, YOUSUF SOLIMAN, and KEENAN CRANE, Carnegie Mellon University

This paper describes a method for efficiently computing parallel transport of tangent vectors on curved surfaces, or more generally, any vector-valued data on a curved manifold. More precisely, it extends a vector field defined over any region to the rest of the domain via parallel transport along shortest geodesics. This basic operation enables fast, robust algorithms for extrapolating level set velocities, inverting the exponential map, computing geometric medians and Karcher/Fréchet means of arbitrary distributions, constructing centroidal Voronoi diagrams, and finding consistently ordered landmarks. Rather than evaluate parallel transport by explicitly tracing geodesics, we show that it can be computed via a short-time heat flow involving the *connection Laplacian*. As a result, transport can be achieved by solving three prefactored linear systems, each akin to a standard Poisson problem. To implement the method we need only a discrete connection Laplacian, which we describe for a variety of geometric data structures (point clouds, polygon meshes, *etc.*). We also study the numerical behavior of our method, showing empirically that it converges under refinement, and augment the construction of intrinsic Delaunay triangulations (iDT) so that they can be used in the context of tangent vector field processing.

CCS: • **Mathematics of computing → Discretization**; **Partial differential equations**; • **Computing methodologies → Shape analysis**;

Additional Key Words and Phrases: discrete differential geometry, parallel transport, velocity extrapolation, logarithmic map, exponential map, Karcher mean, geometric median

ACM Reference Format:
Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. The Vector Heat Method. *ACM Trans. Graph.* 38, 3, Article 00 (June 2019), 19 pages. https://doi.org/00.0000/0000000.0000000

1 INTRODUCTION

Given a vector at a point of a curved domain, how do we find the most parallel vector at all other points (as shown in Fig. 1)? This "most parallel" vector field—not typically considered in numerical algorithms—provides a surprisingly valuable starting point for a wide variety of tasks across geometric and scientific computing, from extrapolating level set velocity to computing centers of distributions. To compute this field, one idea is to transport the vector along explicit paths from the source x to all other points y, but even just constructing these paths is already quite expensive (Sec. 2). We instead leverage a little-used relationship between parallel transport and the *vector heat equation*, which describes the diffusion of a given vector field over a time t. As t goes to zero, the diffused field is related to the original one via parallel transport along minimal geodesics, *i.e.*, shortest paths along the curved domain (Sec. 3.4).

Authors' address: Nicholas Sharp; Yousuf Soliman; Keenan Crane, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213.

0730-0301/2019/6-ART00
https://doi.org/00.0000/0000000.0000000

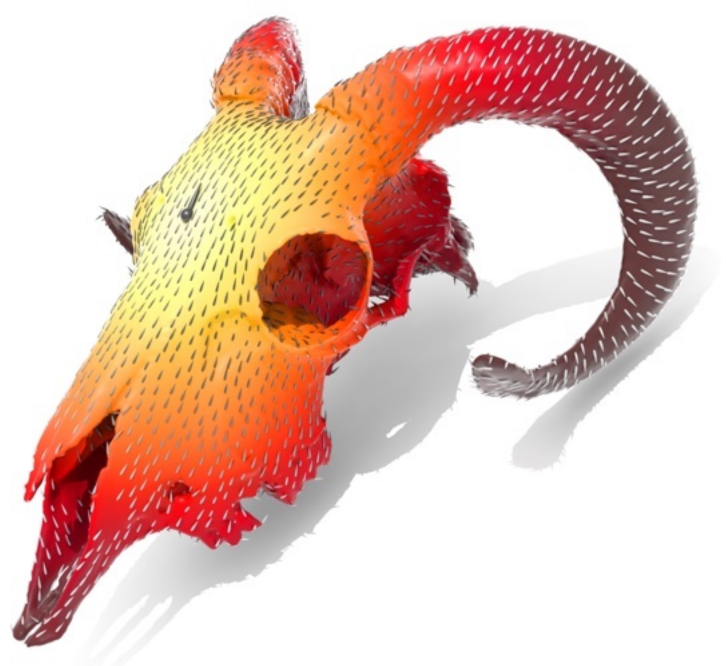


Fig. 1. Given a vector at a point, the vector heat method computes the most parallel vector at every other point. The method easily generalizes to other data (such as a velocity field along a curve), providing a novel and efficient way to implement fundamental algorithms across geometry and simulation.

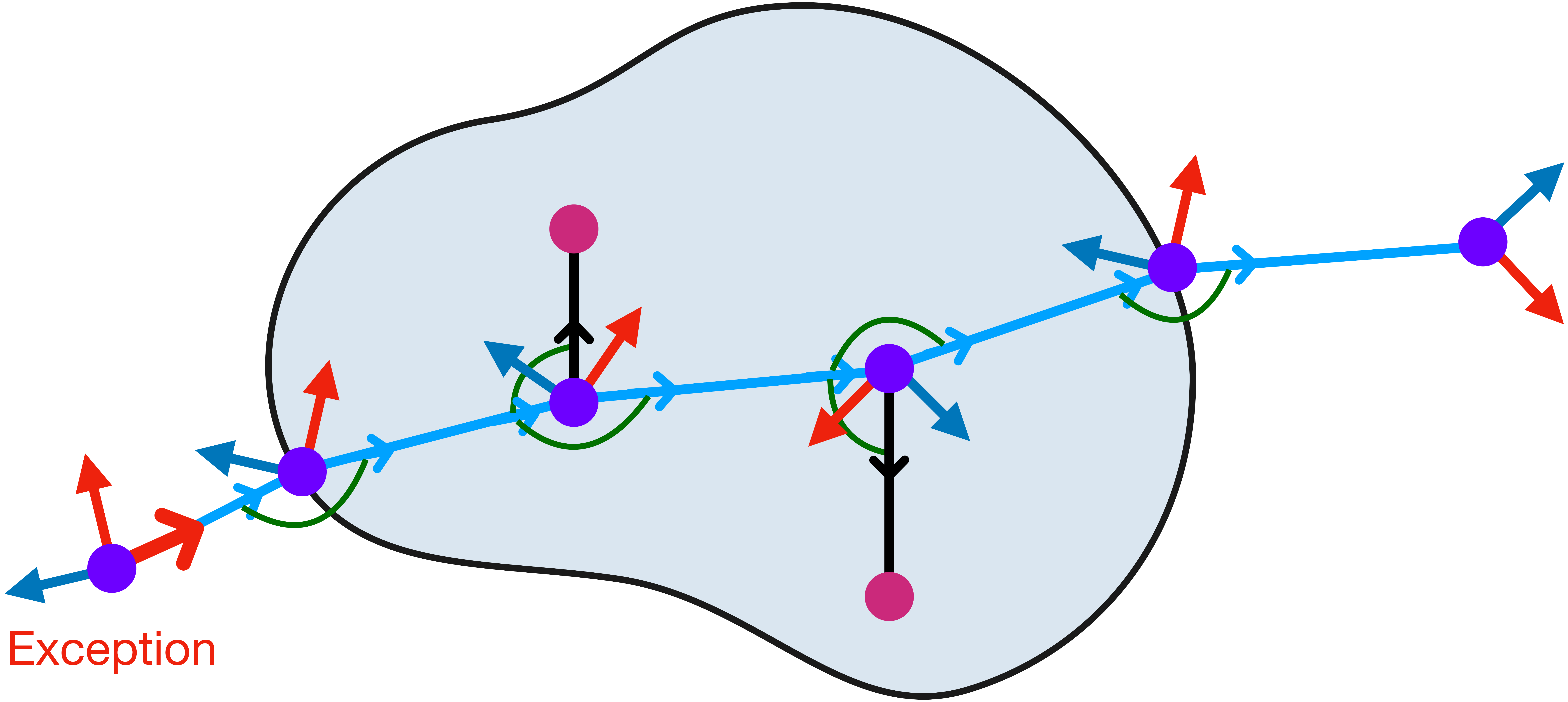
The same principle applies not only to point sources, but also to vector fields over curves or other subsets of the domain. Since diffusion equations are expressed in terms of standard Laplace-like operators, we effectively reduce parallel transport tasks to sparse linear systems that are extremely well-studied in scientific computing—and can hence immediately benefit from mature, high-performance solvers. Moreover, since discrete Laplacians are available for a wide variety of shape representations (polygon meshes, point clouds, *etc.*), and generalize to many kinds of vector data (symmetric direction fields, differential forms, *etc.*), we can apply this same strategy to numerous applications. In particular, this paper introduces

- a fast method for computing parallel transport from a given source set (Sec. 4)
- an augmented intrinsic Delaunay algorithm for vector field processing (Sec. 5.4)
- the first method for computing a logarithmic map over the entire surface, rather than in a local patch (Sec. 8.2), and
- the first method for computing true Karcher/Fréchet means and geometric medians on general surfaces (Sec. 8.3).

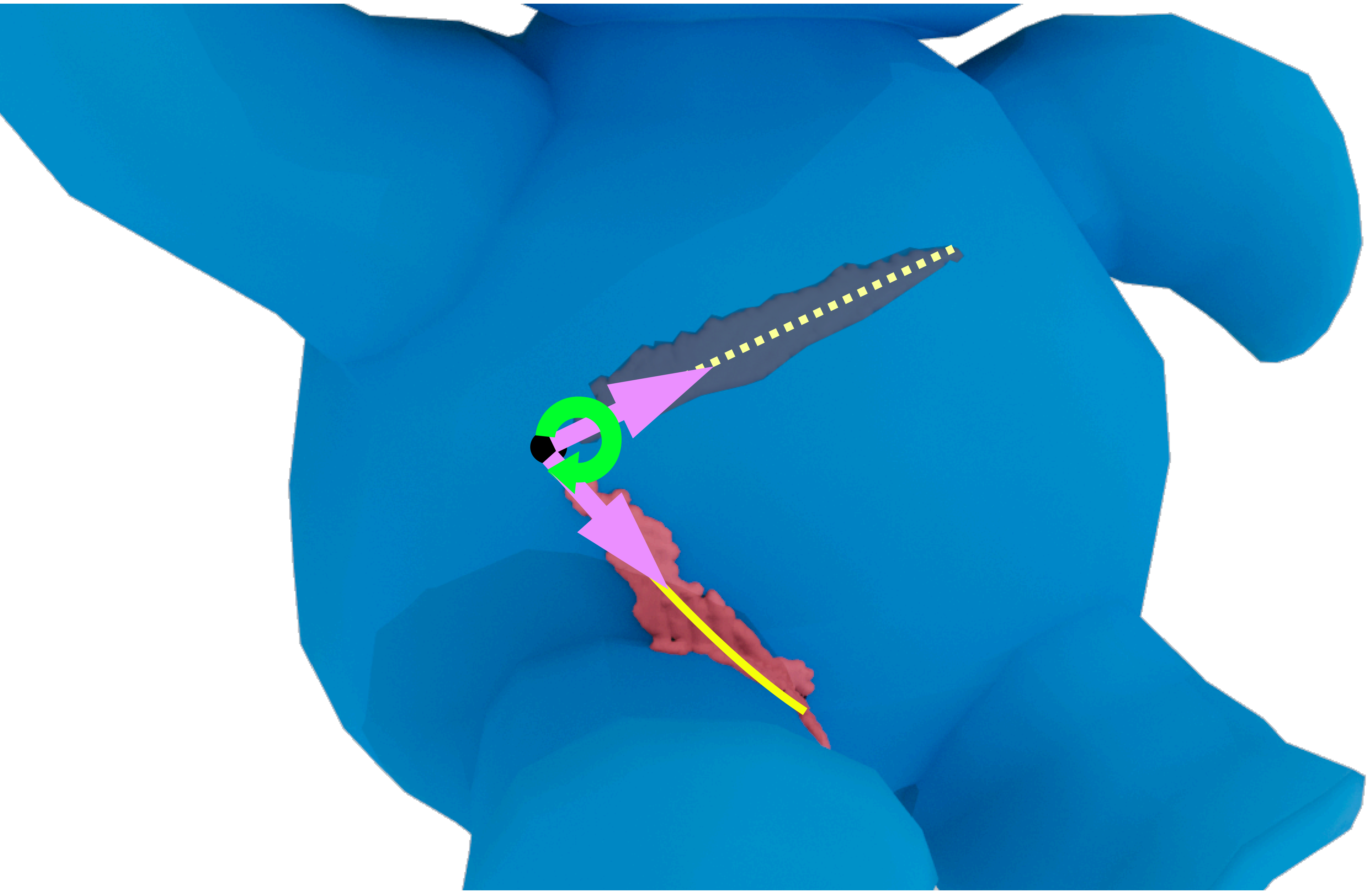
We also describe how to discretize the connection Laplacian on several different geometric data structures and types of vector data (Sec. 6), and consider a variety of other applications including distance-preserving velocity extrapolation for level set methods, computing geodesic centroidal Voronoi tessellations (GCVT), and finding consistently ordered intrinsic landmarks (Sec. 8).

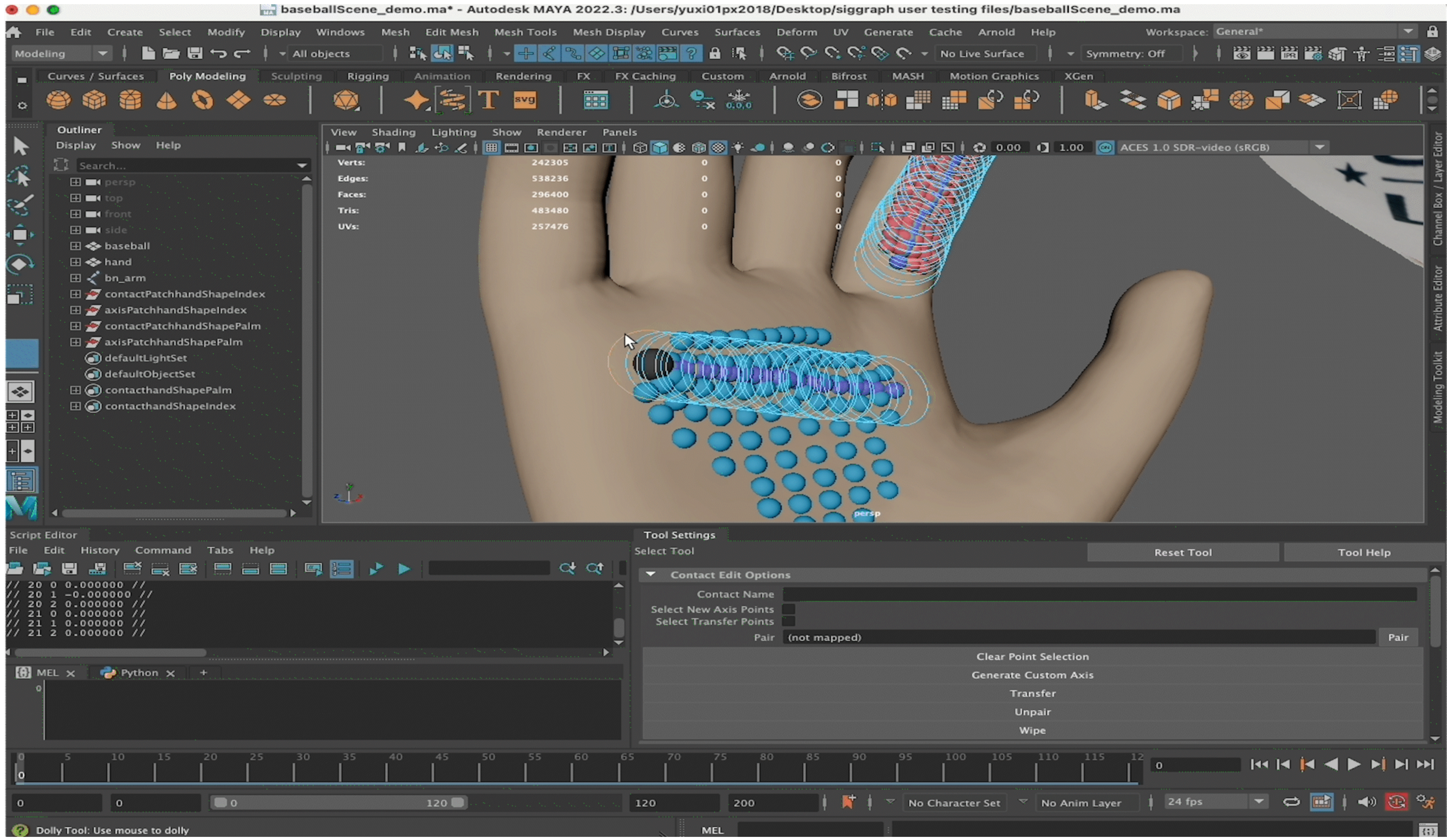
[Sharp & Crane, 2019]

Axis Model Revisited

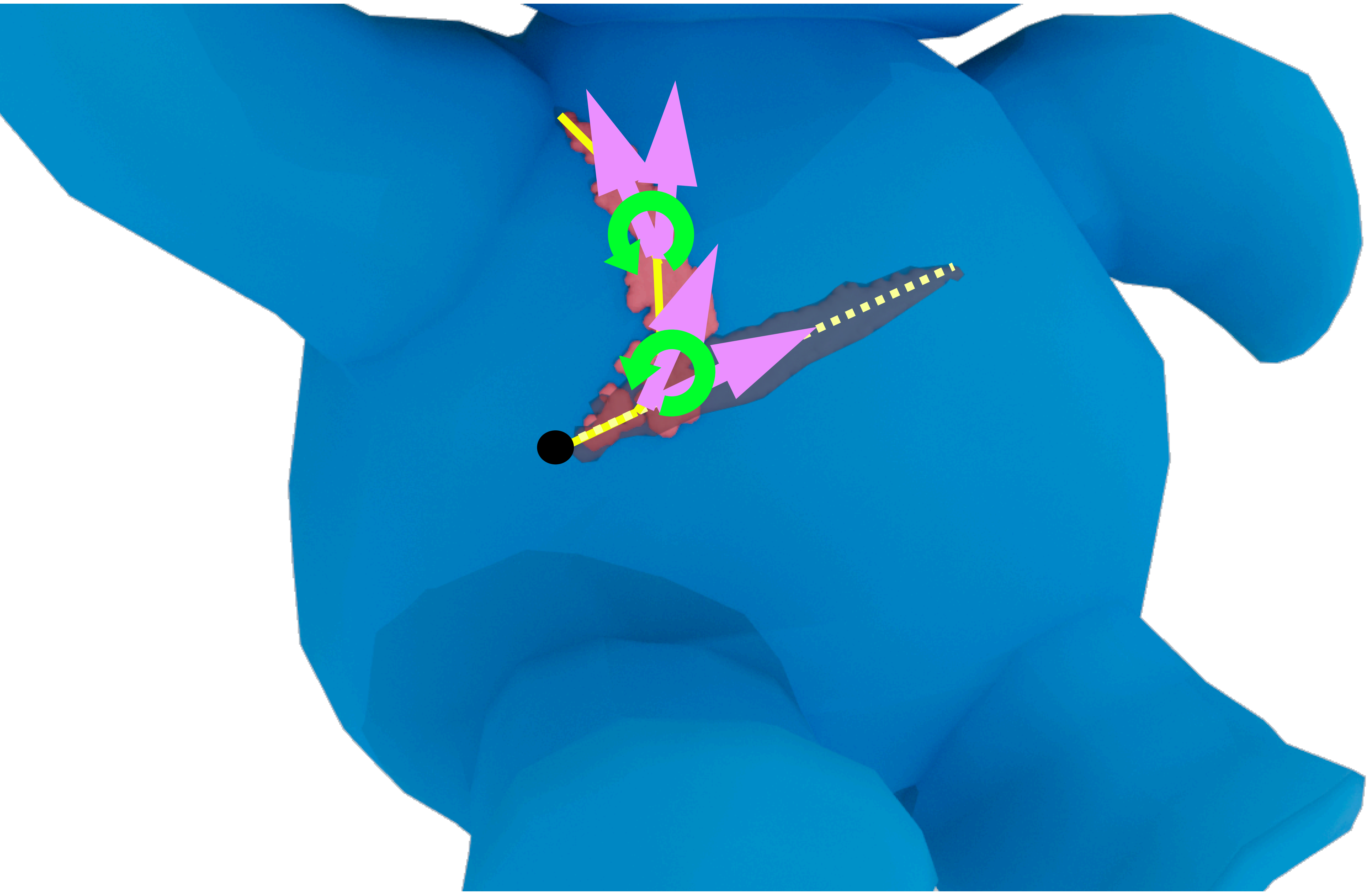


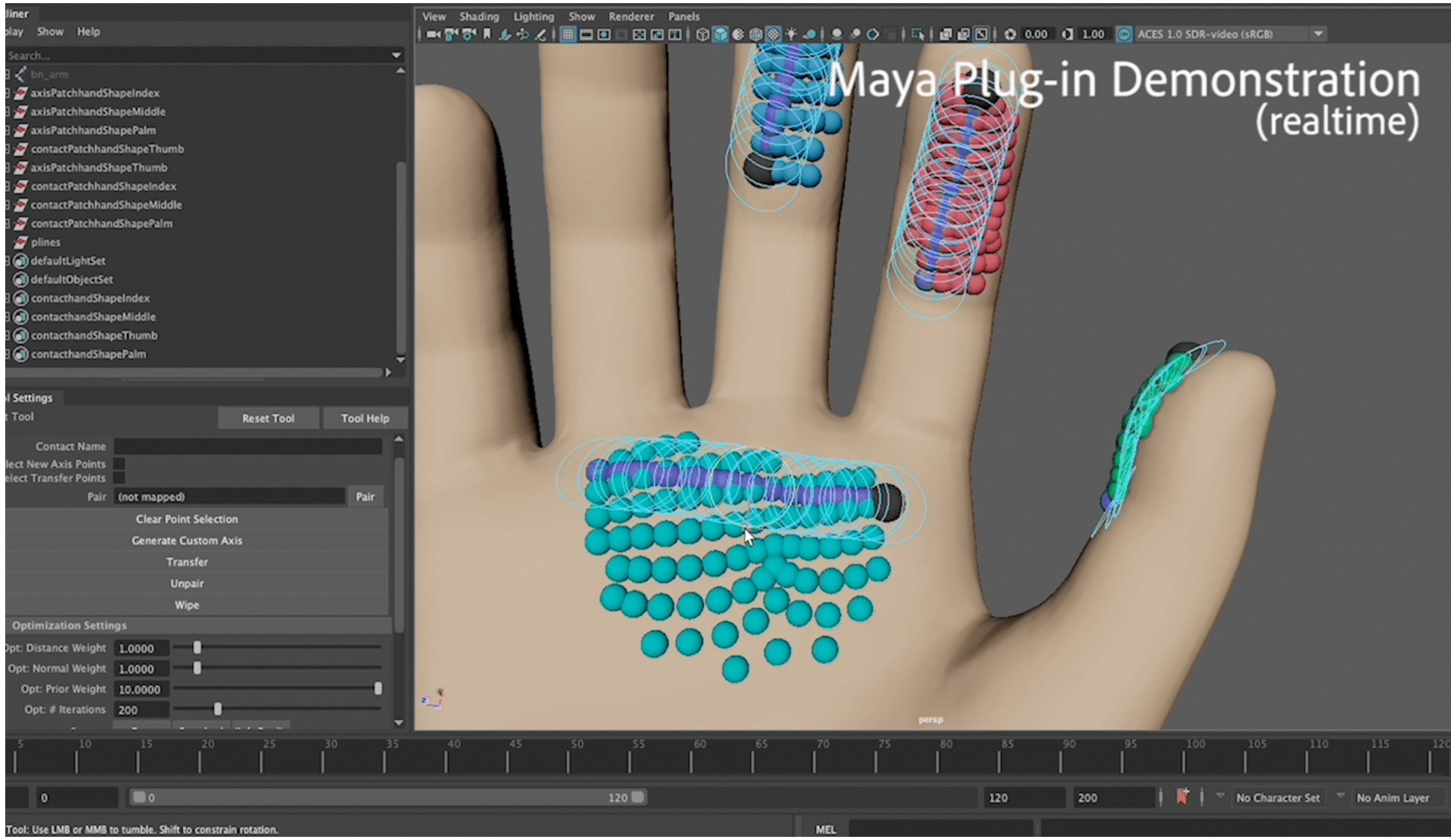
Contact Area Rotation



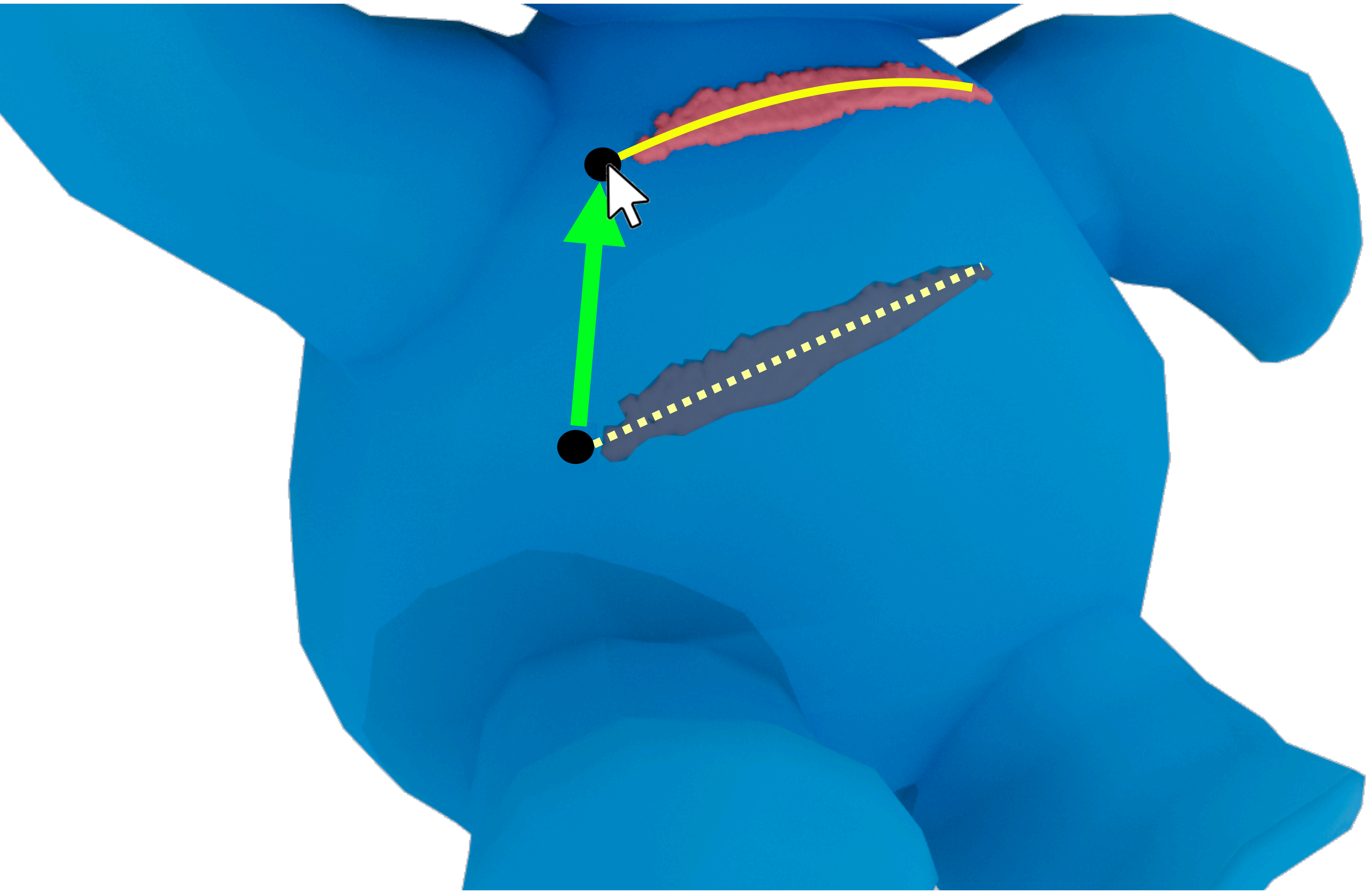


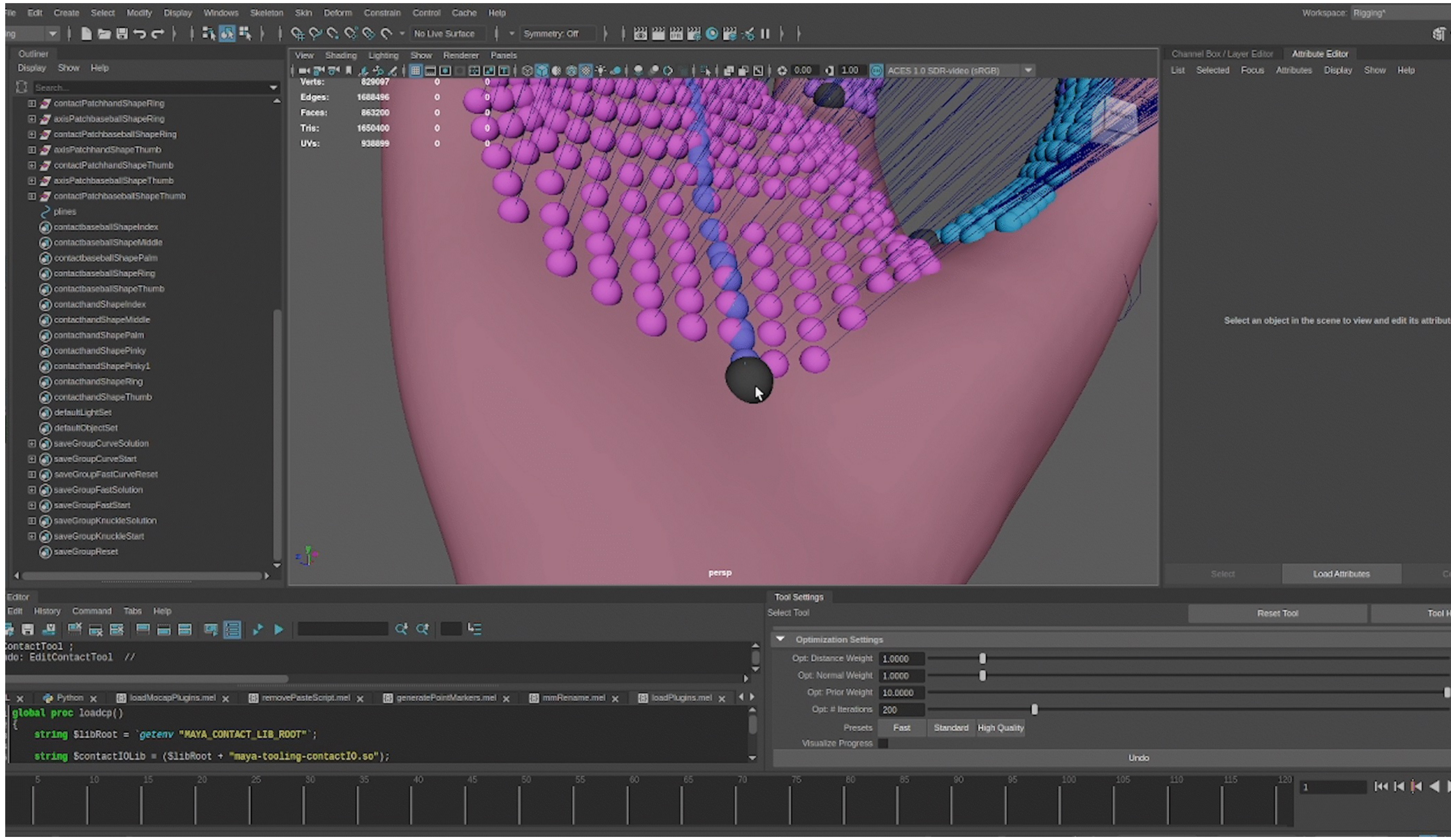
Contact Area Deformation



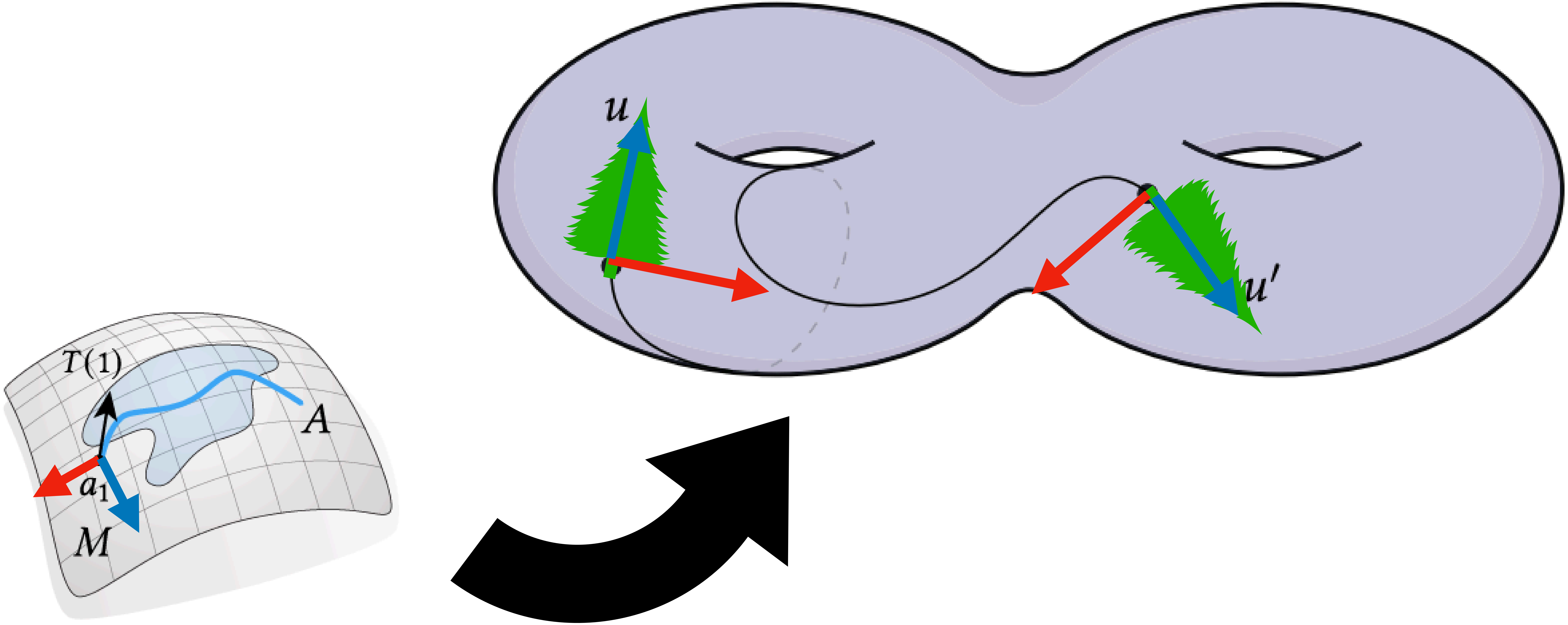


Contact Area Translation

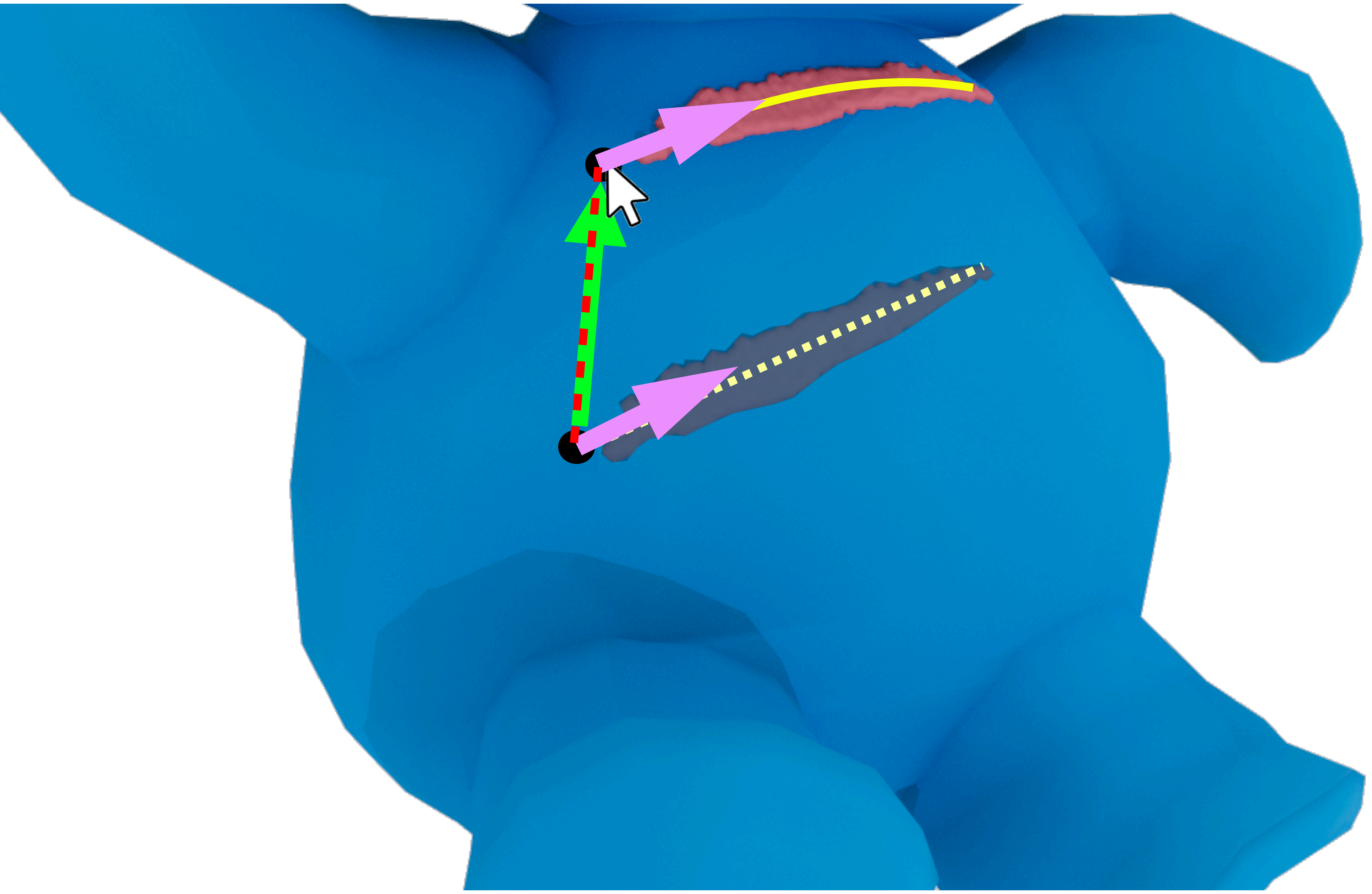


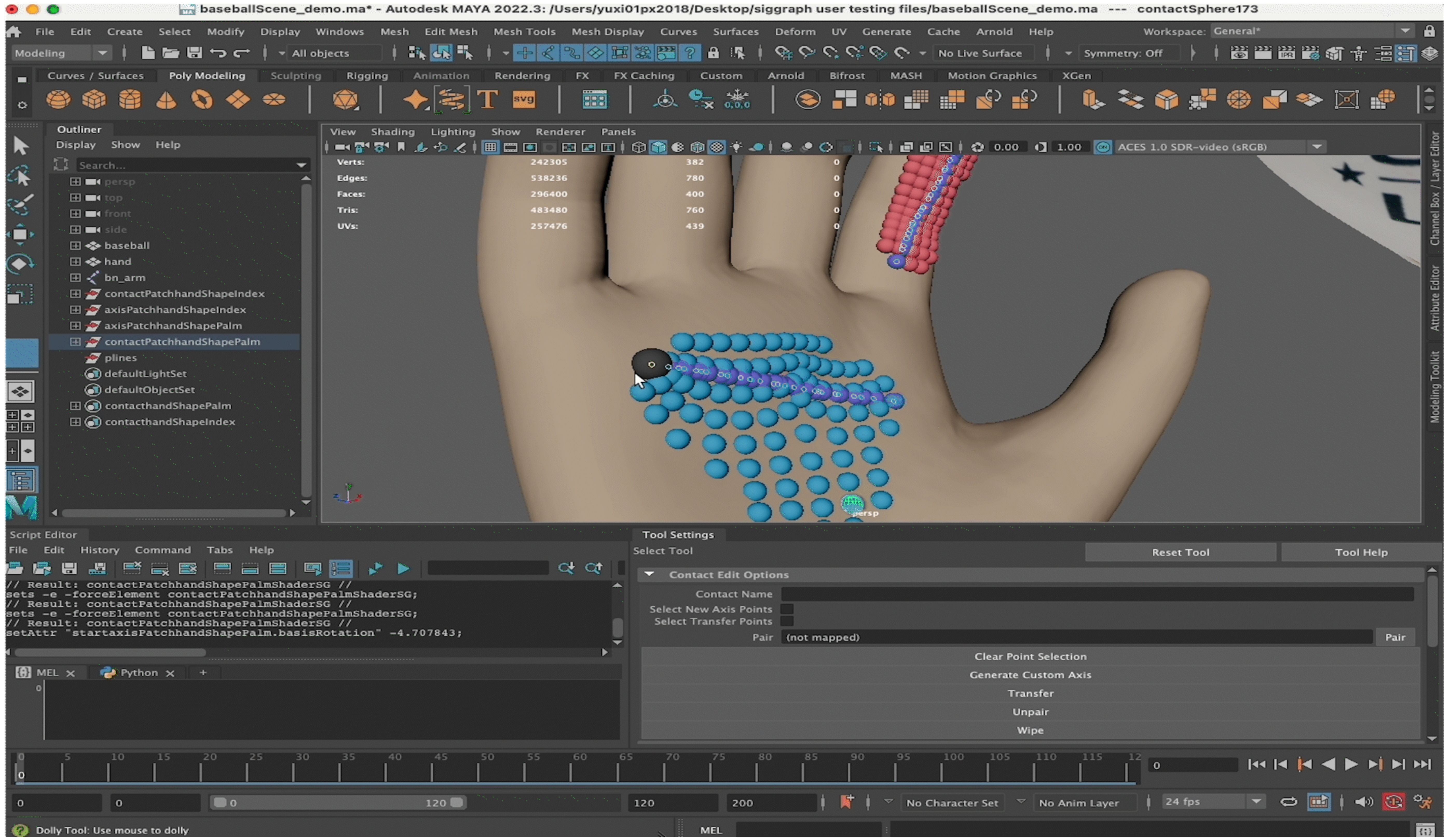


An Important Caveat with Translation

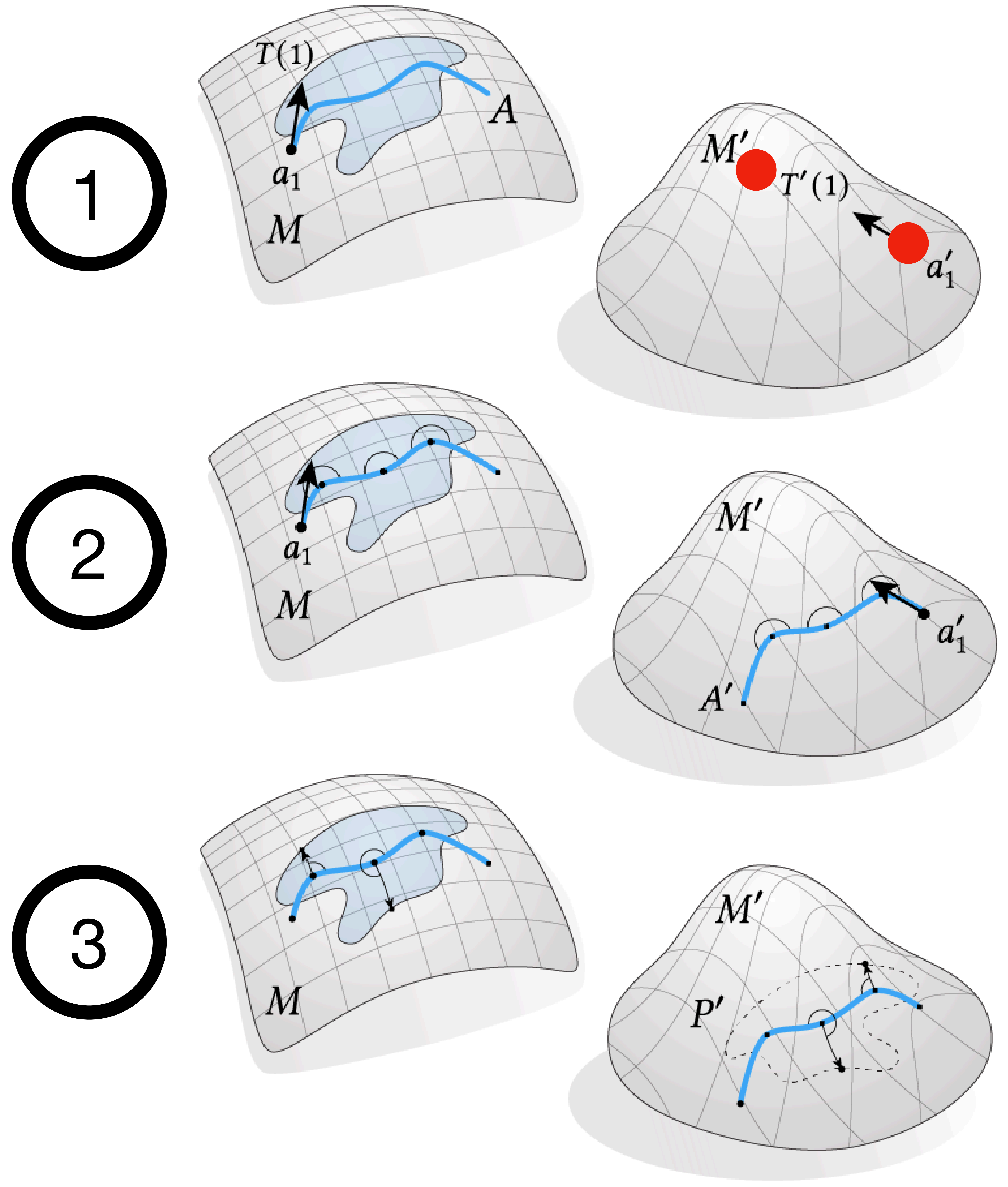


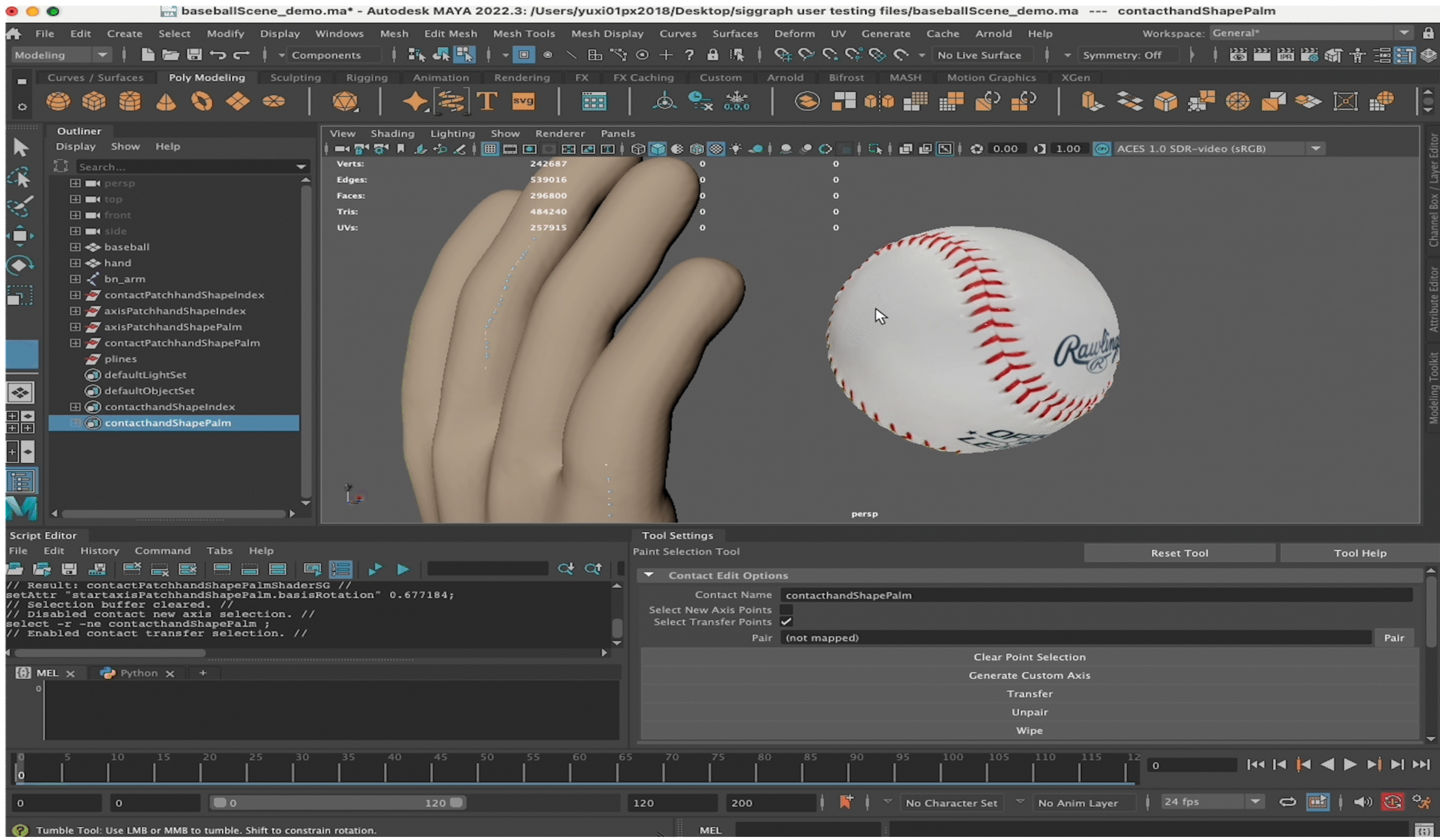
Actual Contact Area Translation



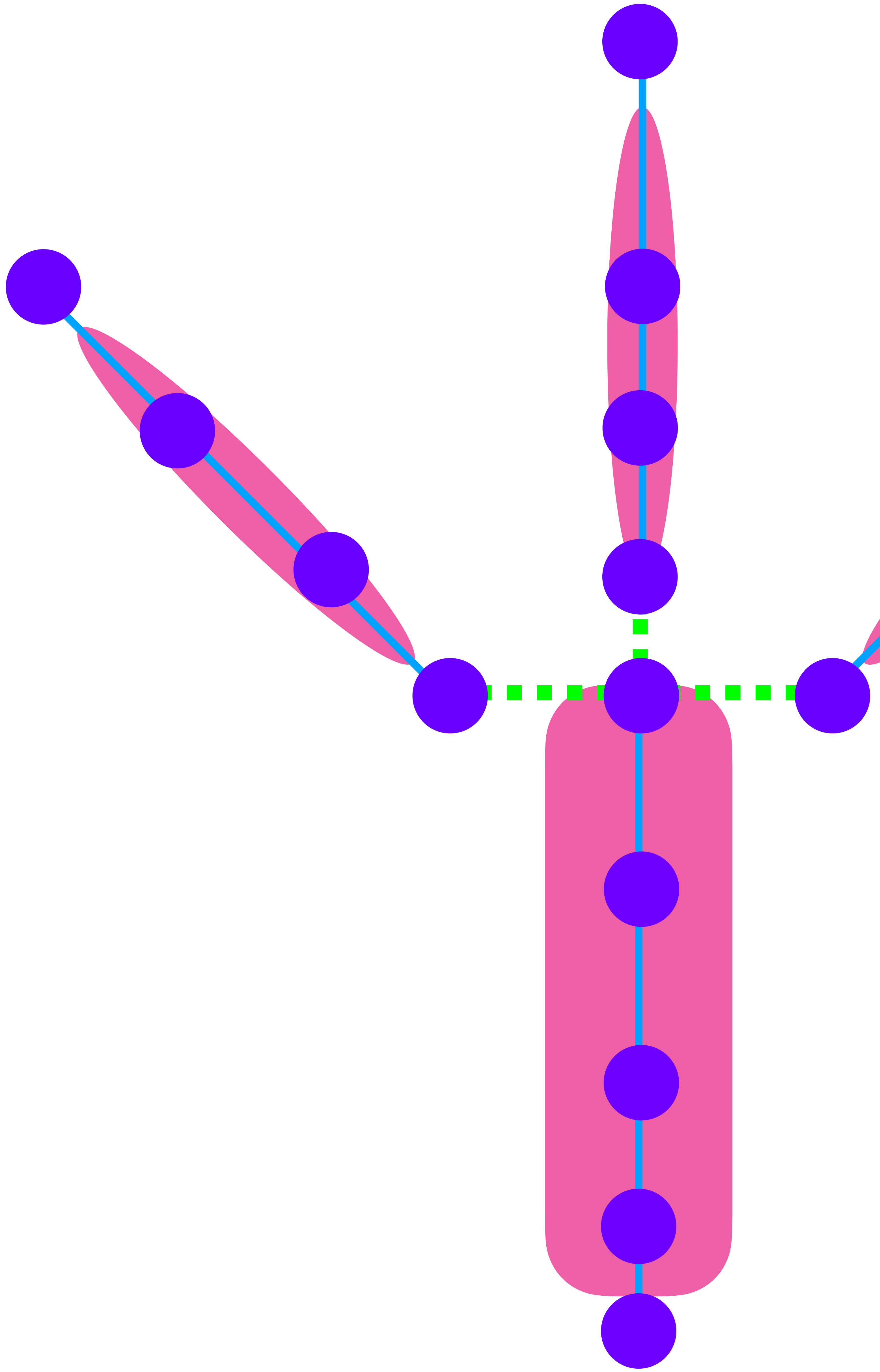


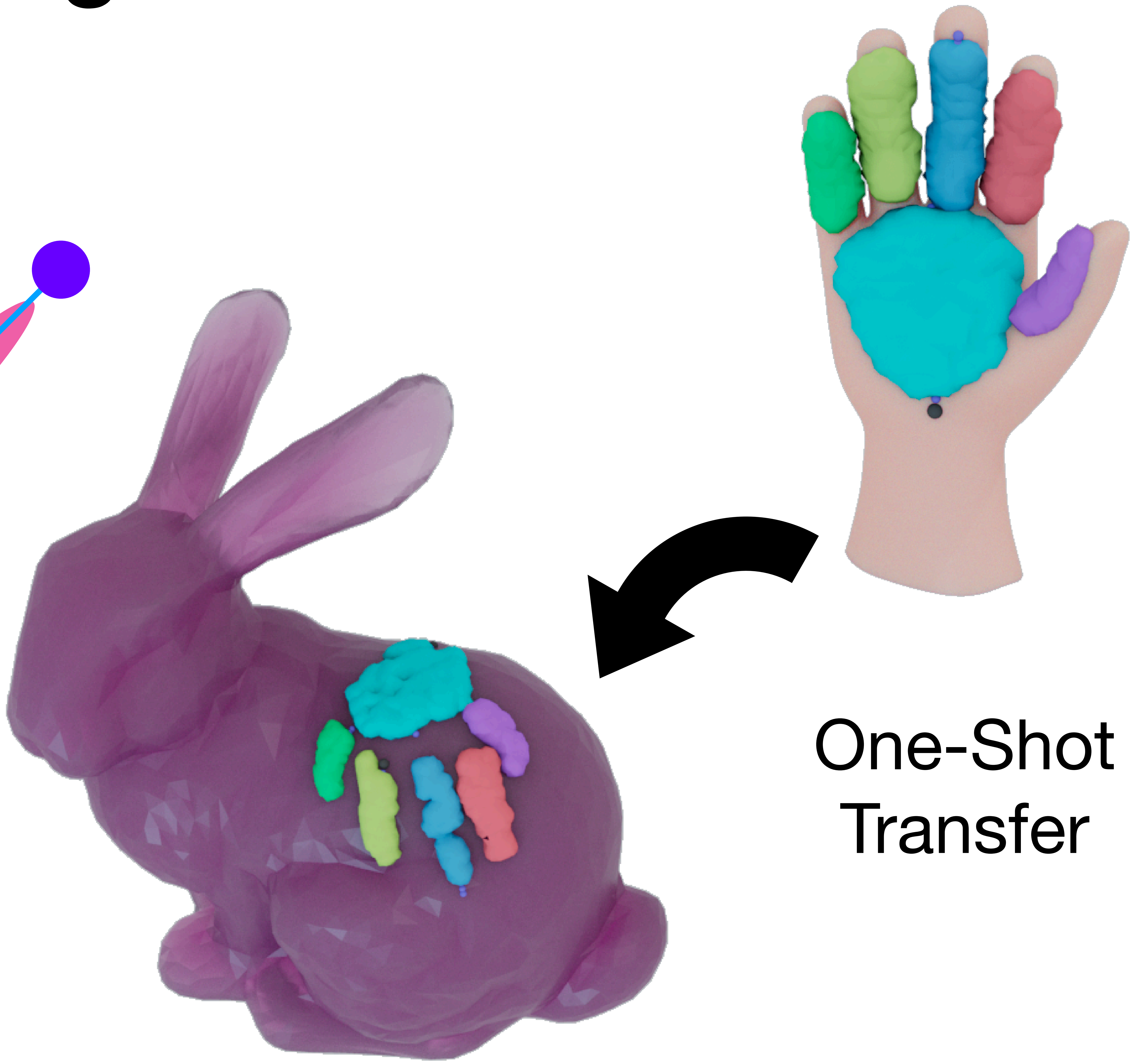
Contact Area Transfer





“Chaining” Areas Together





One-Shot
Transfer

Hand Pose Computation

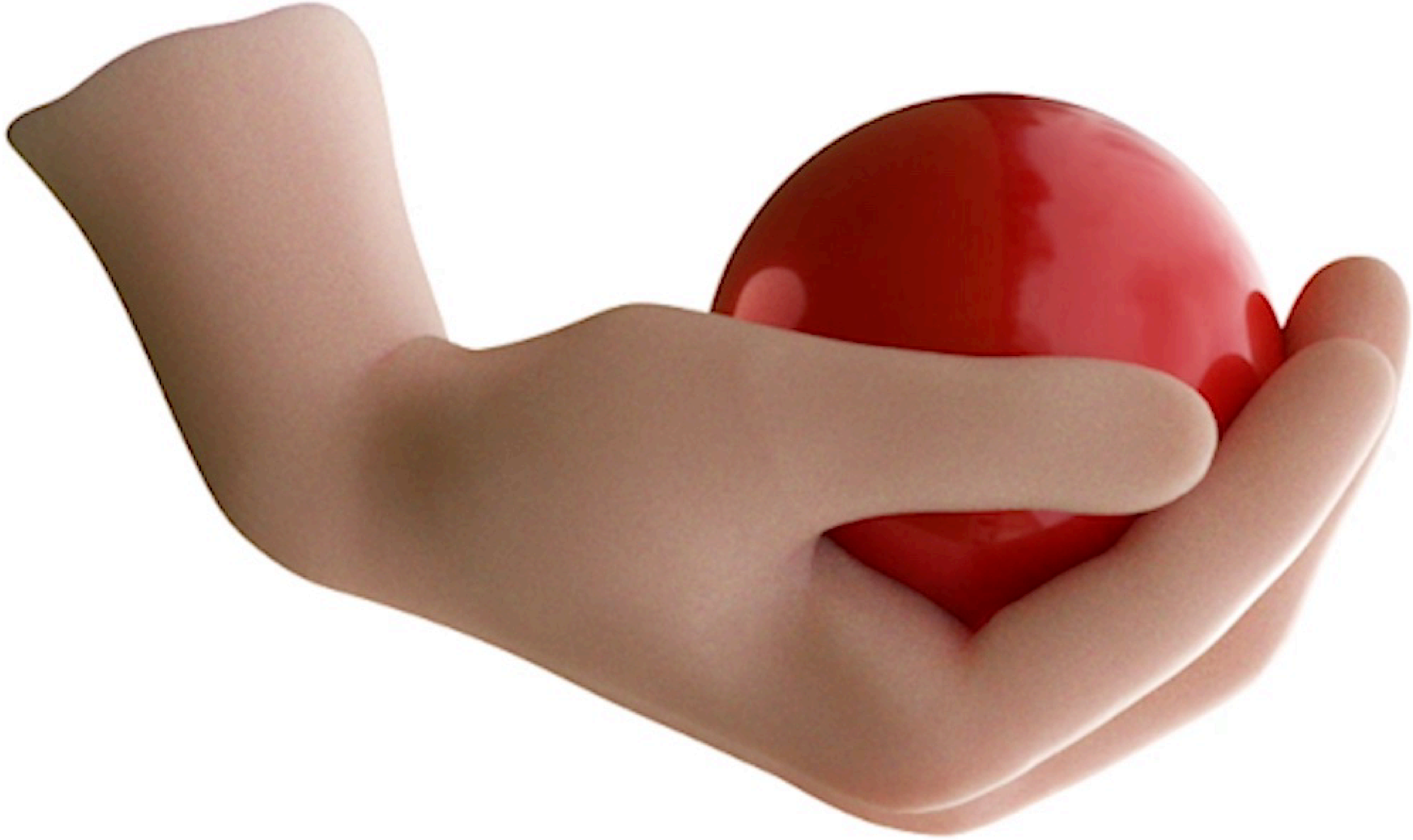
$$\underset{\boldsymbol{\theta}}{\arg\max} \sum_{i=1}^{N} \left(\lambda_d \boldsymbol{\Gamma}(\boldsymbol{\theta})_{D,i} + \lambda_n \boldsymbol{\Gamma}(\boldsymbol{\theta})_{N,i} \right) + \sum_{j=7}^{J} \lambda_p \boldsymbol{\Gamma}(\boldsymbol{\theta})_{P,j}$$

$$\text{s.t.} \quad 0 \leq \boldsymbol{\theta} < 2\pi$$

$\lambda_p >> \lambda_d, \lambda_n$ since $N >> J$

N: Total # of Discrete Contacts
J: Total # of DOFs
$\lambda_d, \lambda_n, \lambda_p$: Weighting Coefficients
$\boldsymbol{\theta}$: DOF Vector

$\boldsymbol{\Gamma}(\boldsymbol{\theta})_D$: Contact Distance Error
$\boldsymbol{\Gamma}(\boldsymbol{\theta})_N$: Contact Normal Error
$\boldsymbol{\Gamma}(\boldsymbol{\theta})_P$: Regularization Error



Does it Actually Work?


Drawbacks
Good!
Solve
...Fired?
Solve
Minor Degradation
Transfer
Transfer
Severe Degradation
Contact
Boundary
Axis

Summary

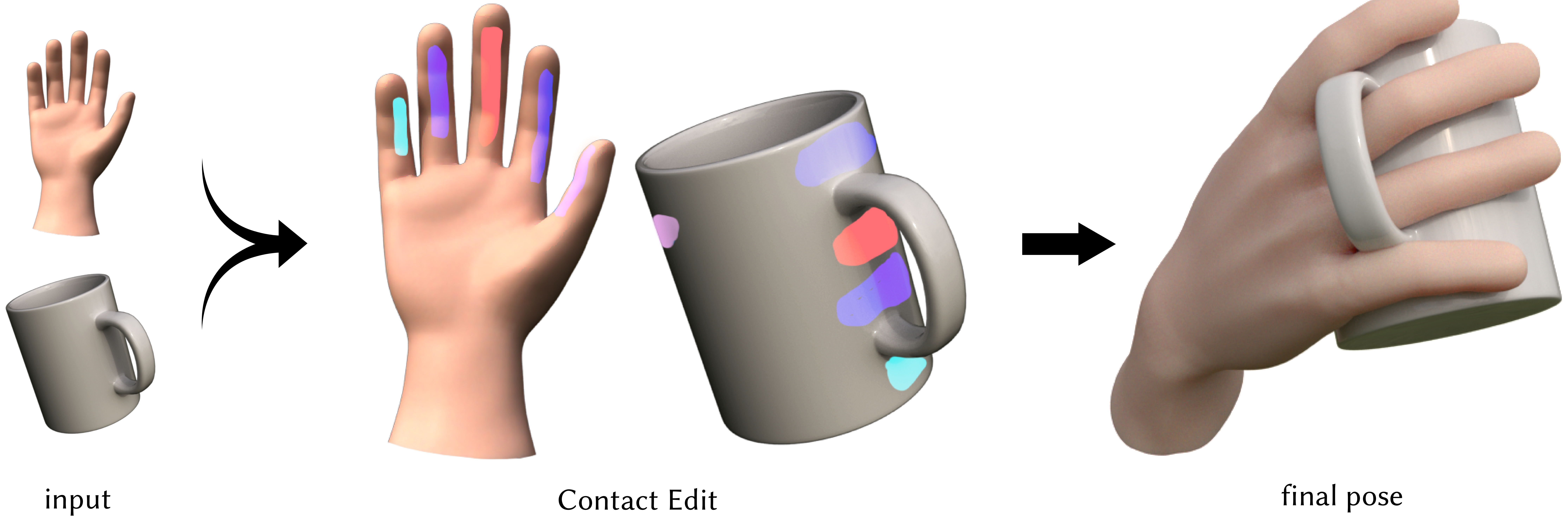
New tools and algorithms for contact areas in traditional animation workflows

✓ Allows areas to serve as first class primitives

✓ Real time editing operations

✓ Simple IK solver

✓ Intuitive and useful to animators

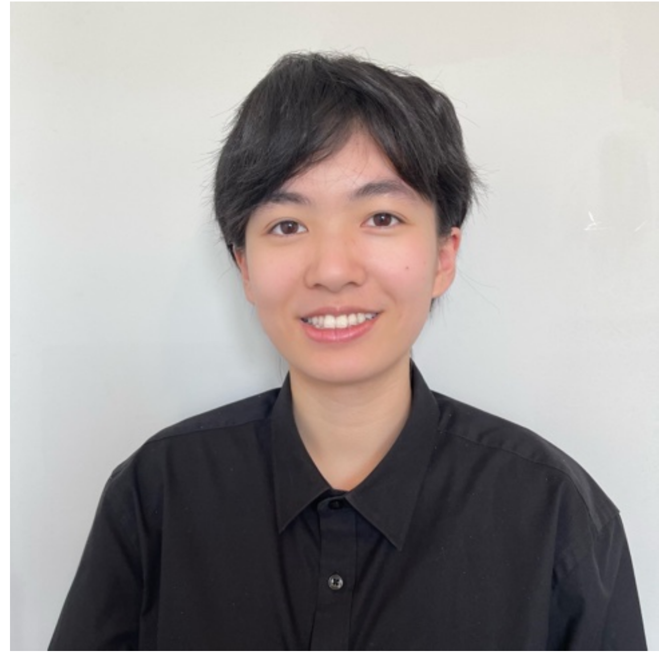


Thank You!










Interested in Trying it Out?

Email: aslakshm@andrew.cmu.edu