SIGGRAPH ASIA 2023 SYDNEY

Robust Skin Weights Transfer via Weight Inpainting

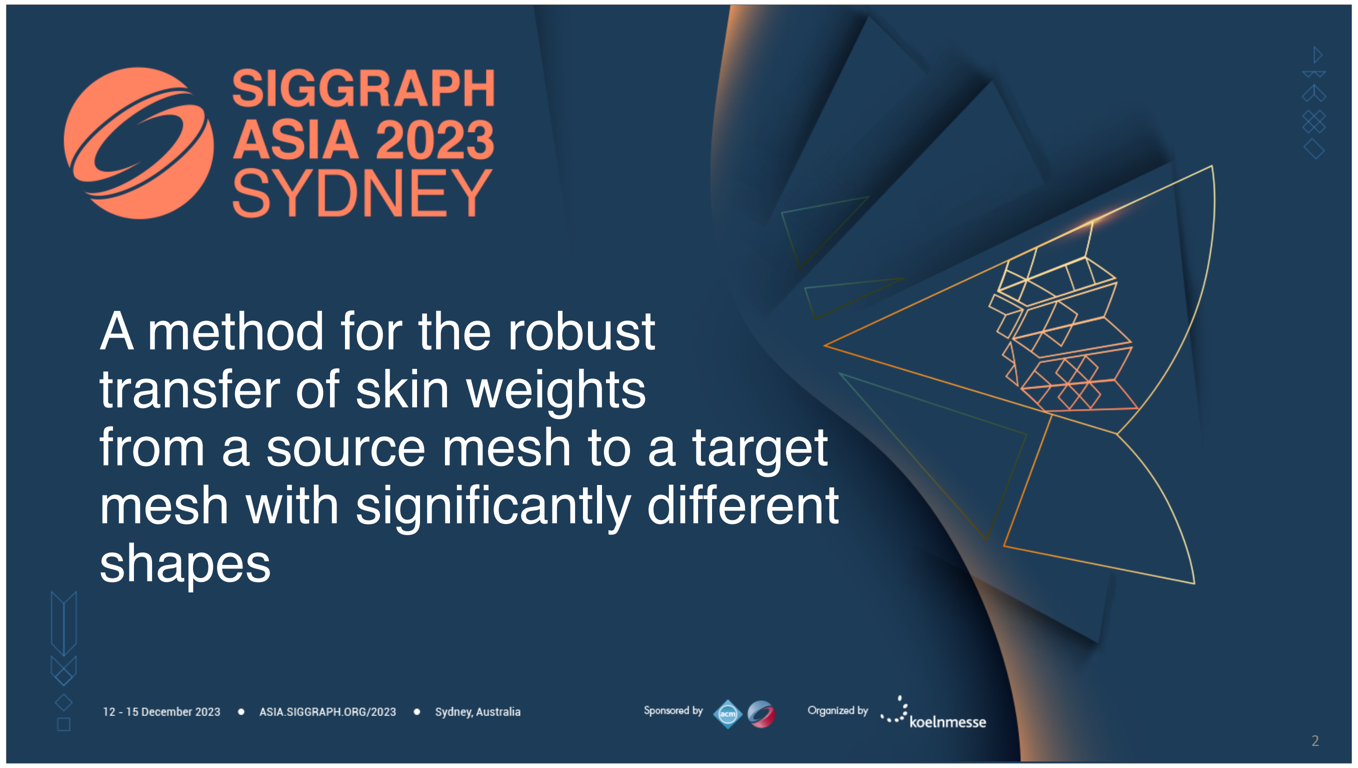
Rinat Abdrashitov	Epic Games
Kim Raichstat	Epic Games
Jared Monsen	Epic Games
David Hill	Epic Games

Connecting STORIES

The 16th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia

CONFERENCE 12 - 15 December 2023
EXHIBITION 13 - 15 December 2023
ICC, Sydney, Australia
ASIA.SIGGRAPH.ORG/2023

Sponsored by acm
Organized by koelnmesse



SIGGRAPH ASIA 2023 SYDNEY

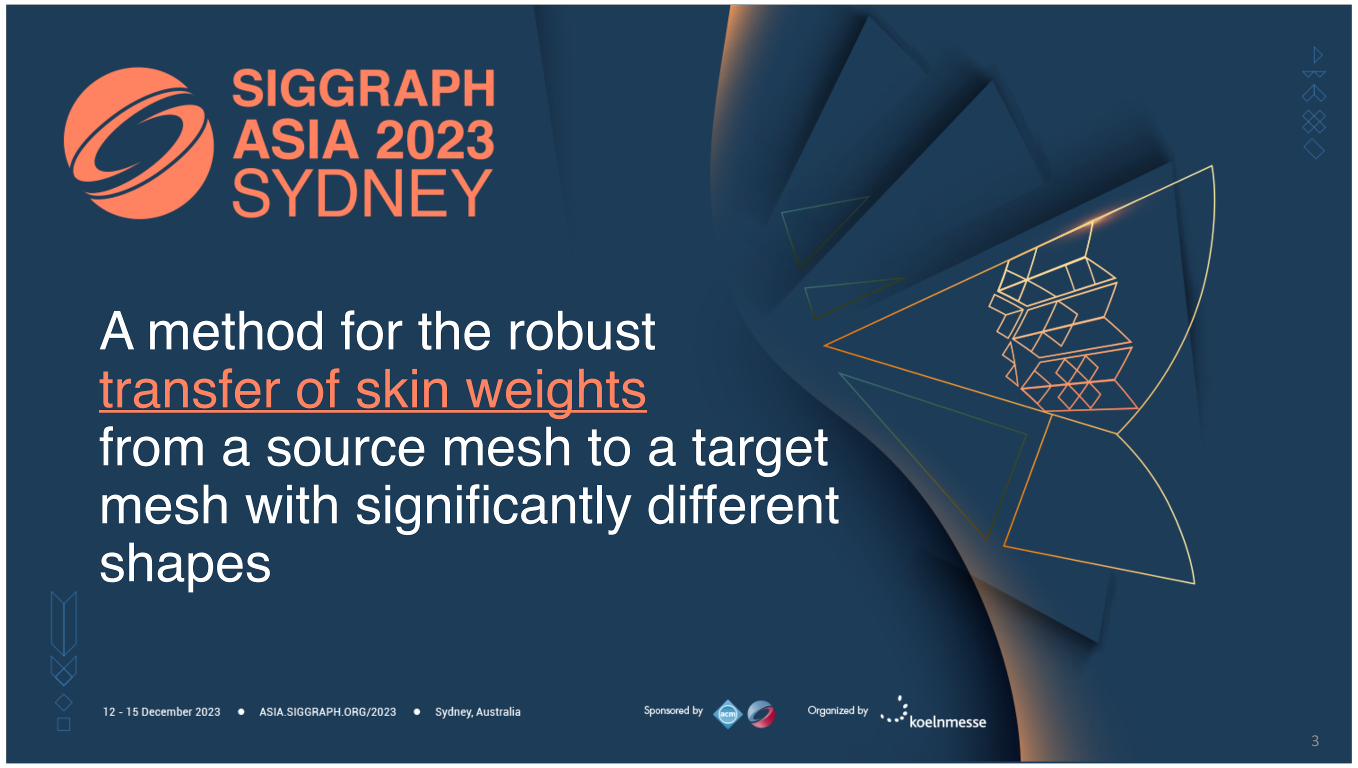
A method for the robust transfer of skin weights from a source mesh to a target mesh with significantly different shapes

12 - 15 December 2023 • ASIA.SIGGRAPH.ORG/2023 • Sydney, Australia

Sponsored by

Organized by koelnmesse

In this talk I will present a new method for the robust transfer of skin weights from a source mesh to a target mesh with significantly different shapes.



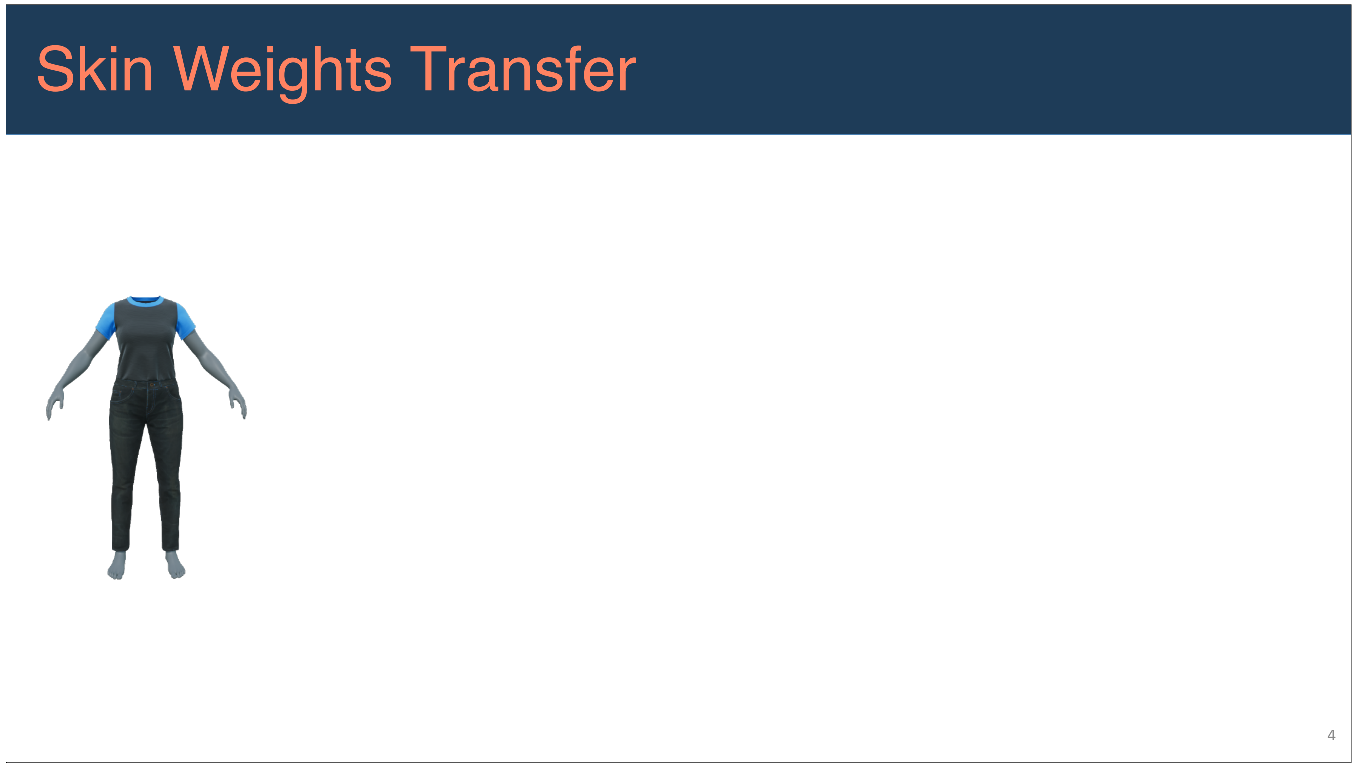
SIGGRAPH ASIA 2023 SYDNEY

A method for the robust transfer of skin weights from a source mesh to a target mesh with significantly different shapes

12 - 15 December 2023 • ASIA.SIGGRAPH.ORG/2023 • Sydney, Australia

Sponsored by Organized by koelnmesse

Lets first understand what is a skin weights transfer problem and what are the issues with prior work that attempted to solve it

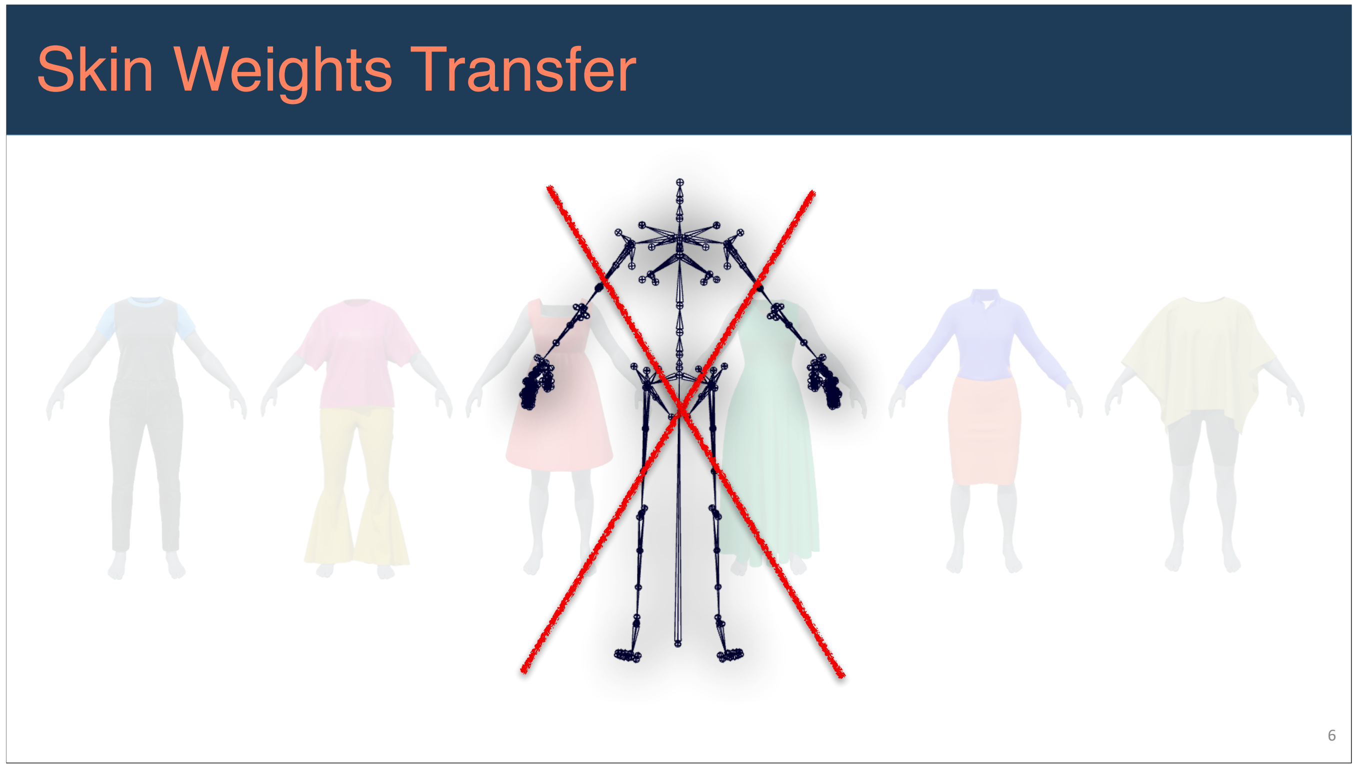


Skin Weights Transfer

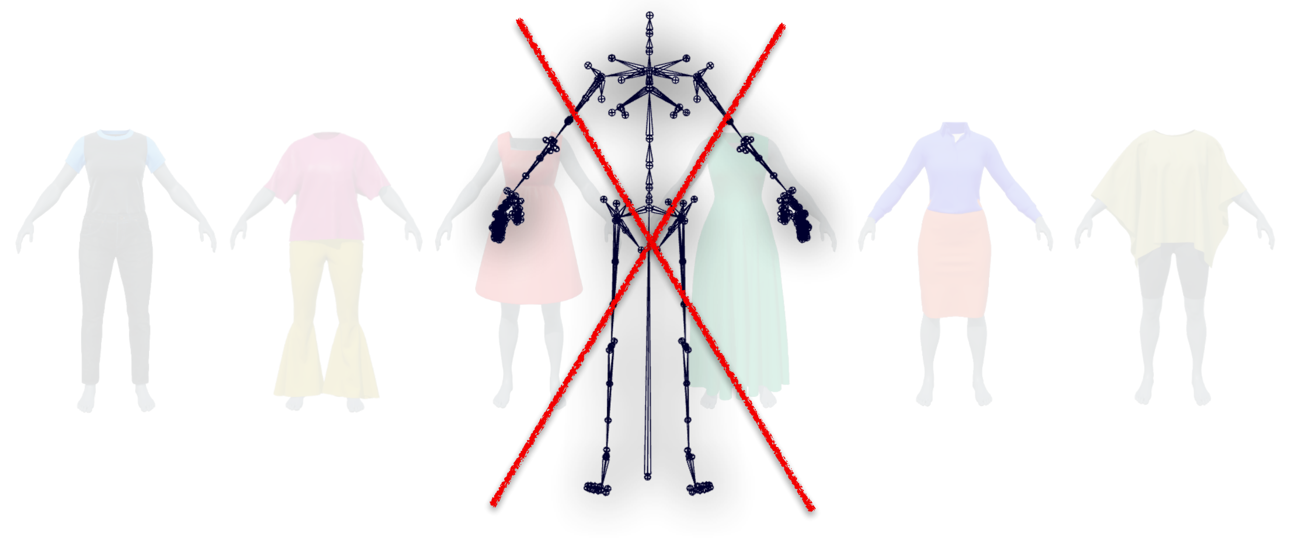
We can focus on the most common application of this algorithm where we are given a body mesh whose skin weights are **manually** and carefully painted by an artist and we want to **automatically** compute the skin weights of clothing by transferring the weights from the body.

Skin Weights Transfer

The issue is that clothing designs can vary a lot and if we want to go beyond simple tshirts and jeans we need this transfer process to be very robust to handle all of these designs.



Skin Weights Transfer



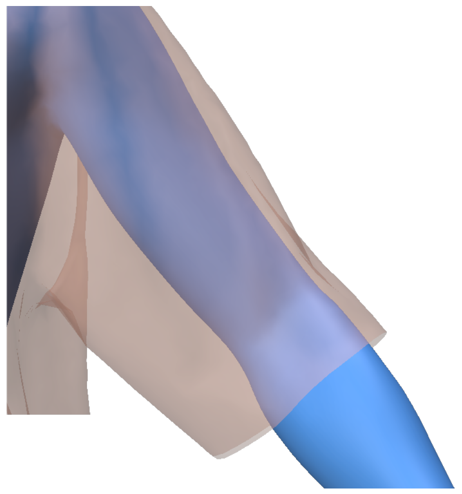
Important to note here is that this is a different problem from binding meshes to skeleton which has received a lot of attention in the research literature.

The skeleton is **not** an input to our transfer algorithm.

We already have skin weights on one mesh and we want to transfer them to another mesh.

Closest Point Transfer

Some variation of the closest point algorithm:

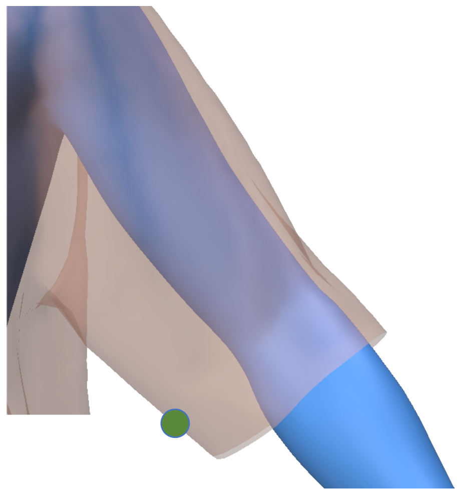


And this problem has received little attention.
Current solutions in both commercially available tools and prior academic work boil down to some variation of the closest point algorithm.

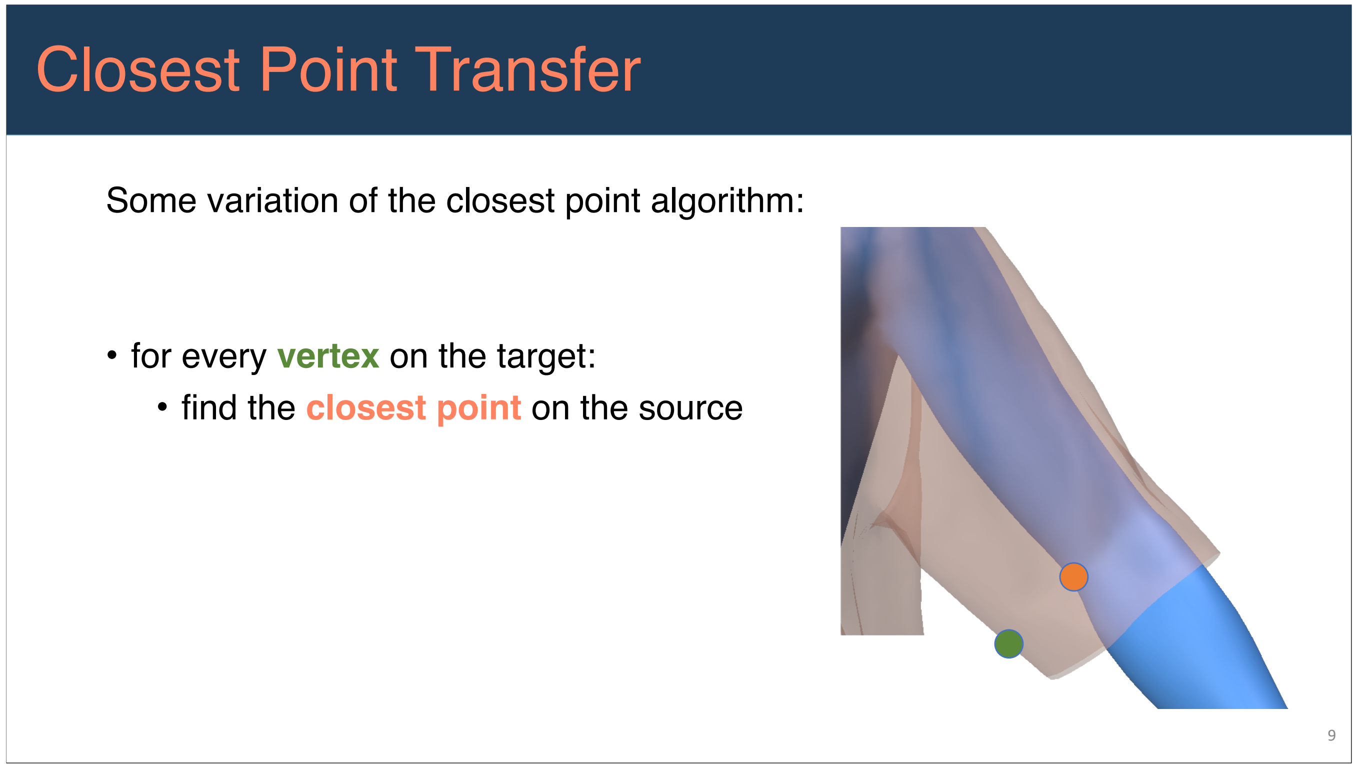
Closest Point Transfer

Some variation of the closest point algorithm:

- for every **vertex** on the target:



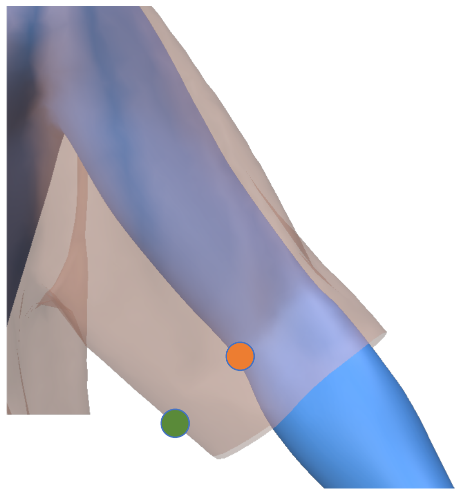
Meaning that, for every vertex on the target mesh…



Closest Point Transfer

Some variation of the closest point algorithm:

- for every **vertex** on the target:
    - find the **closest point** on the source

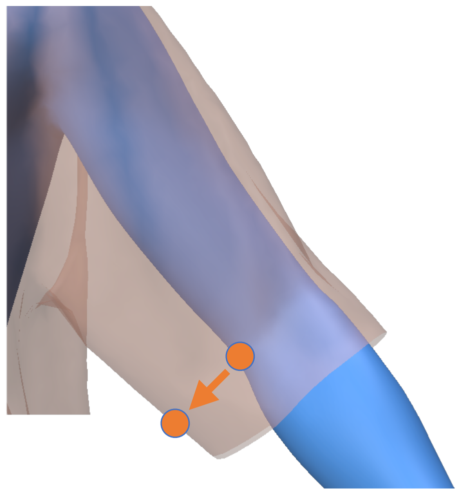


…you find the closest point on the source mesh…

Closest Point Transfer

Some variation of the closest point algorithm:

- for every **vertex** on the target:
  - find the **closest point** on the source
  - copy the weights

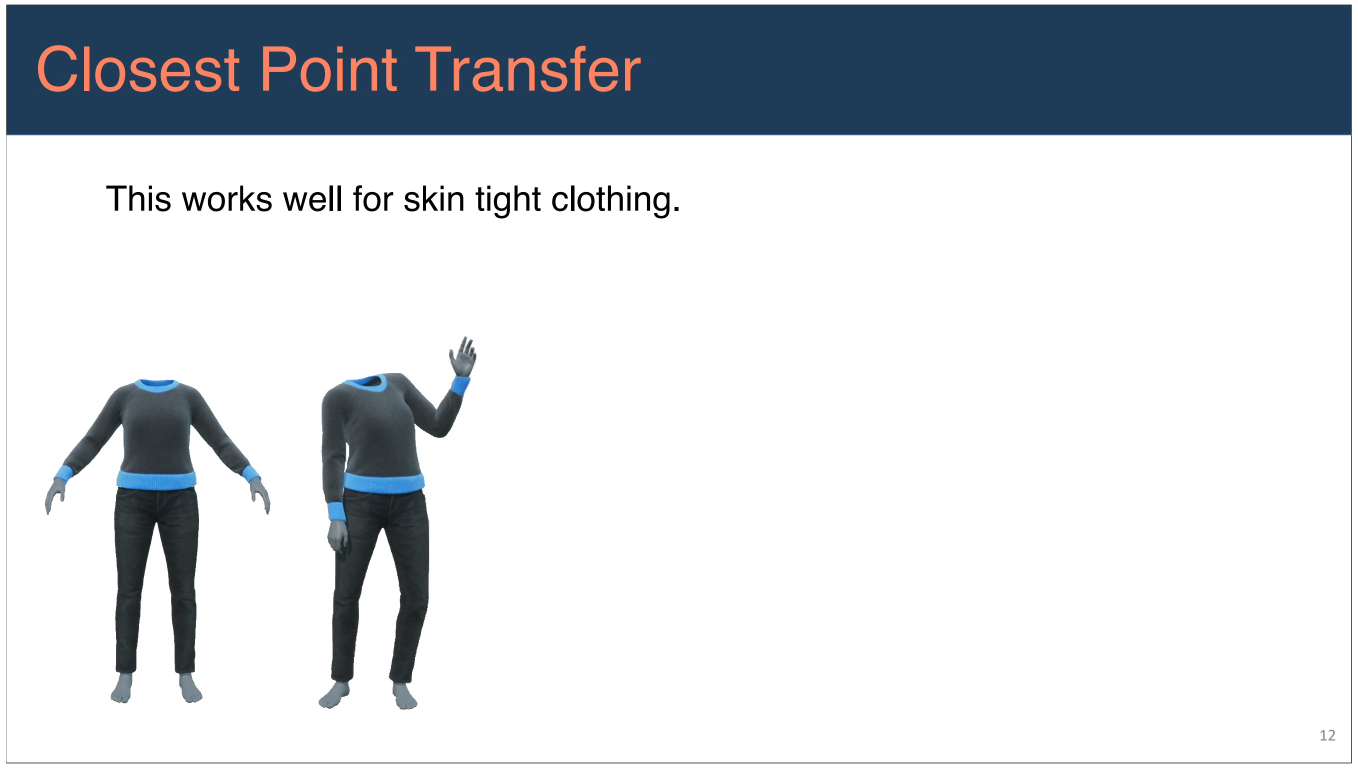


…and then copy weights. Very straightforward.

Closest Point Transfer

This works well for skin tight clothing.

To be fair, this almost works perfectly well for somewhat skin tight clothing, so if you take a sweater and jeans and transfer weights to them….



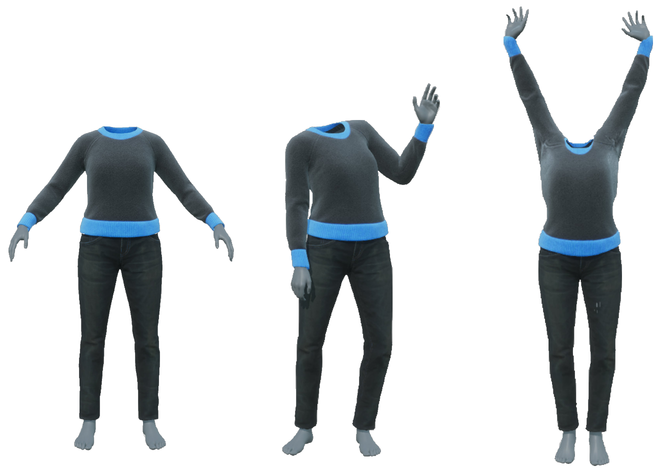
Closest Point Transfer

This works well for skin tight clothing.

… and the pose the body, the clothing nicely deforms with the body and you get good results, but…

Closest Point Transfer

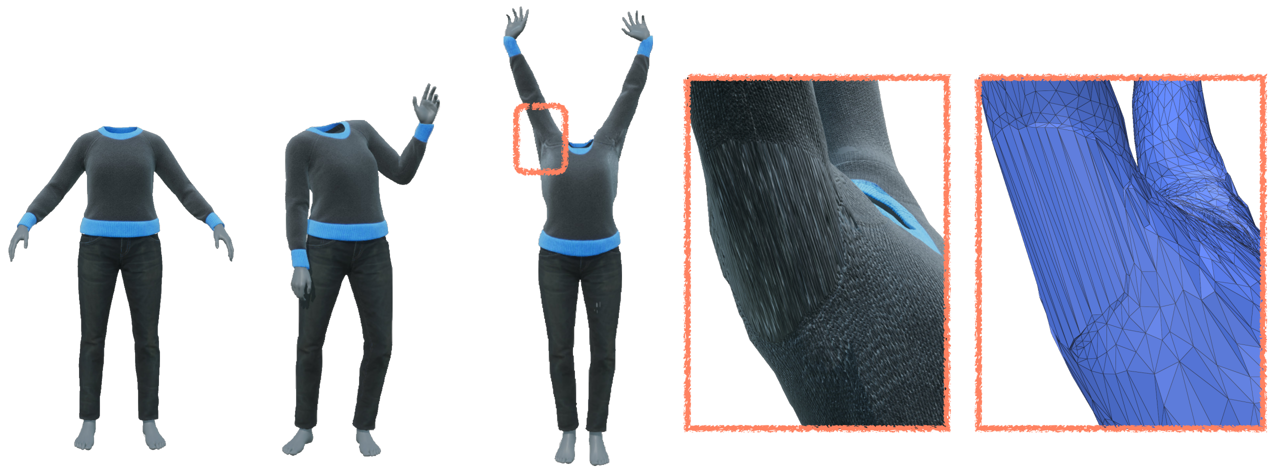
This works well for skin tight clothing.



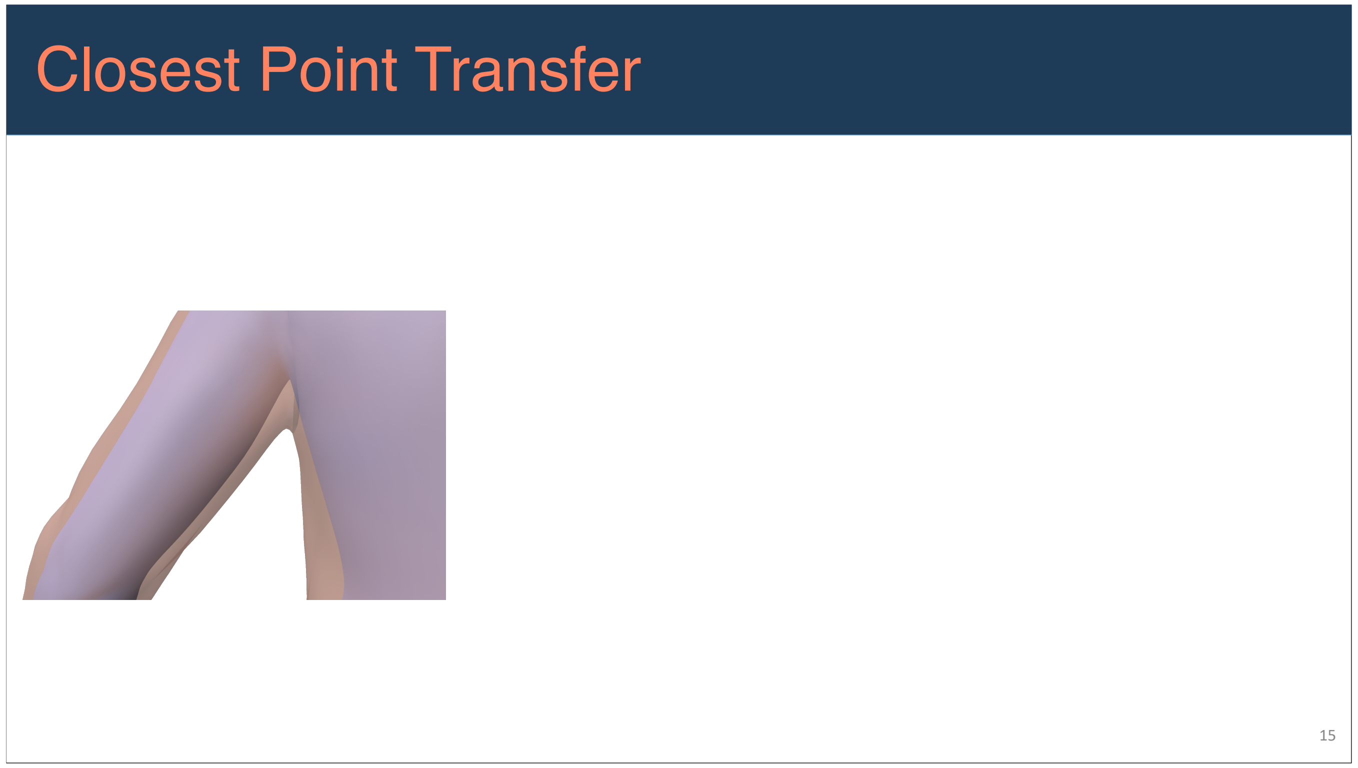
…if we try another pose where we raise the arms you will see …

Closest Point Transfer

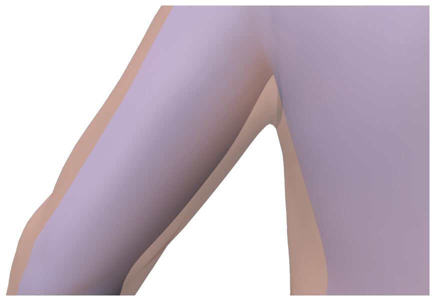
This works well for skin tight clothing.



…that the area around the armpits gets stretched….

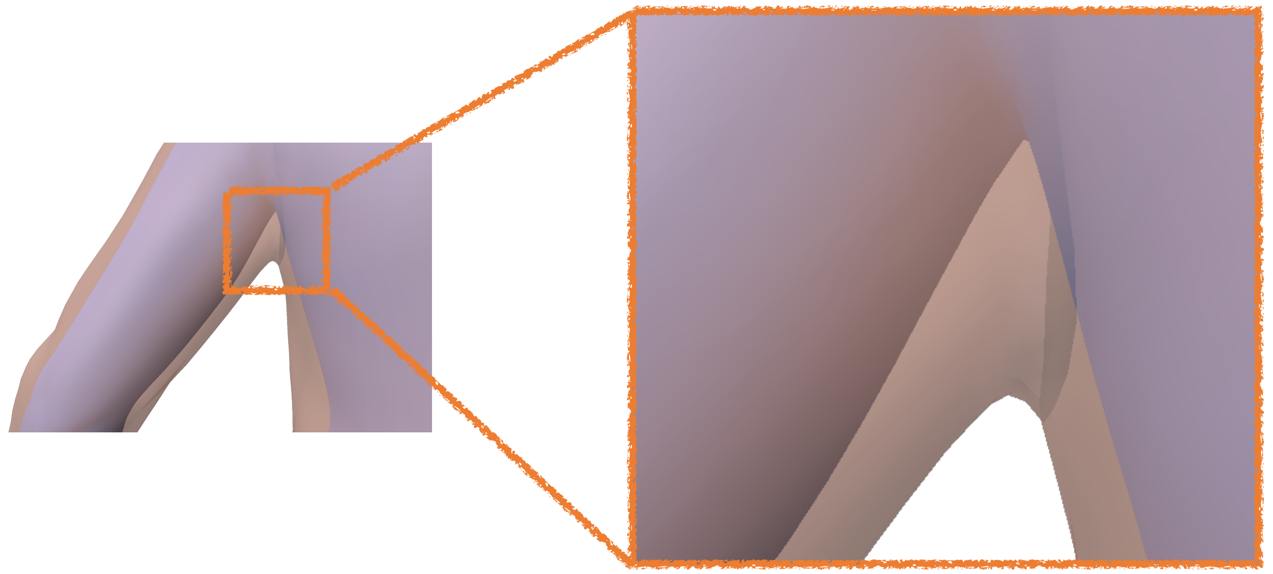


Closest Point Transfer

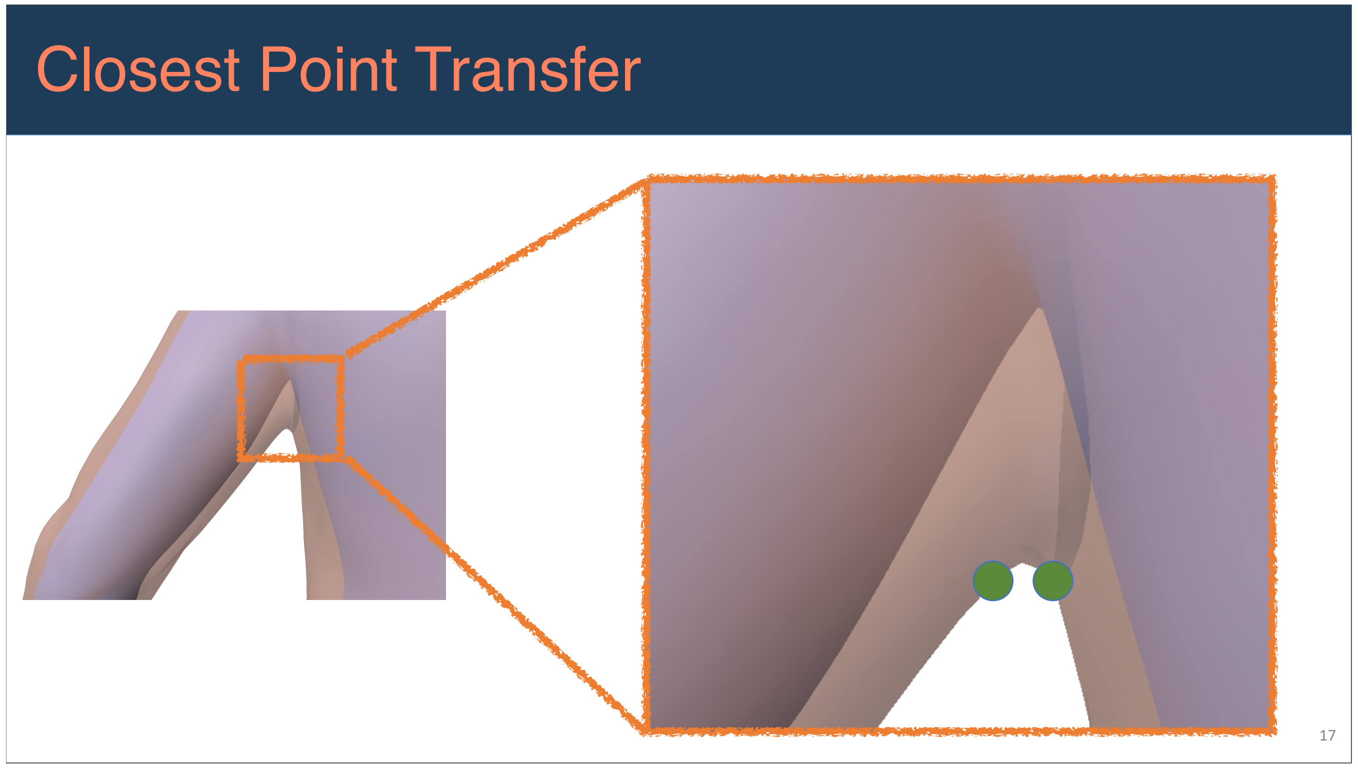


Lets quickly try to understand why this is happening.

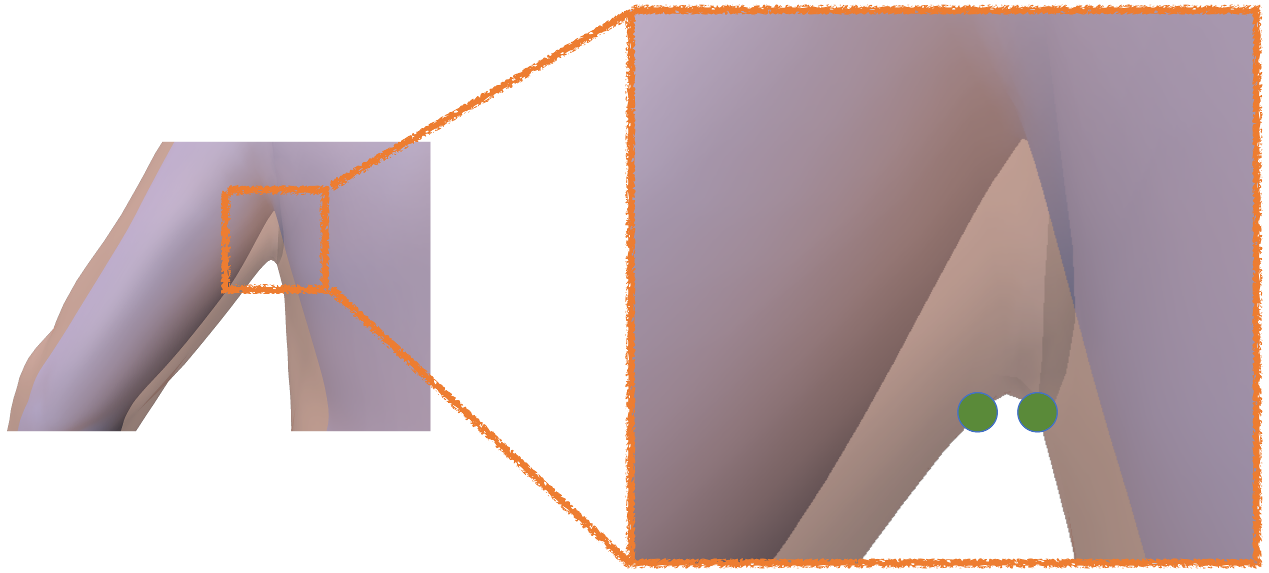
Closest Point Transfer



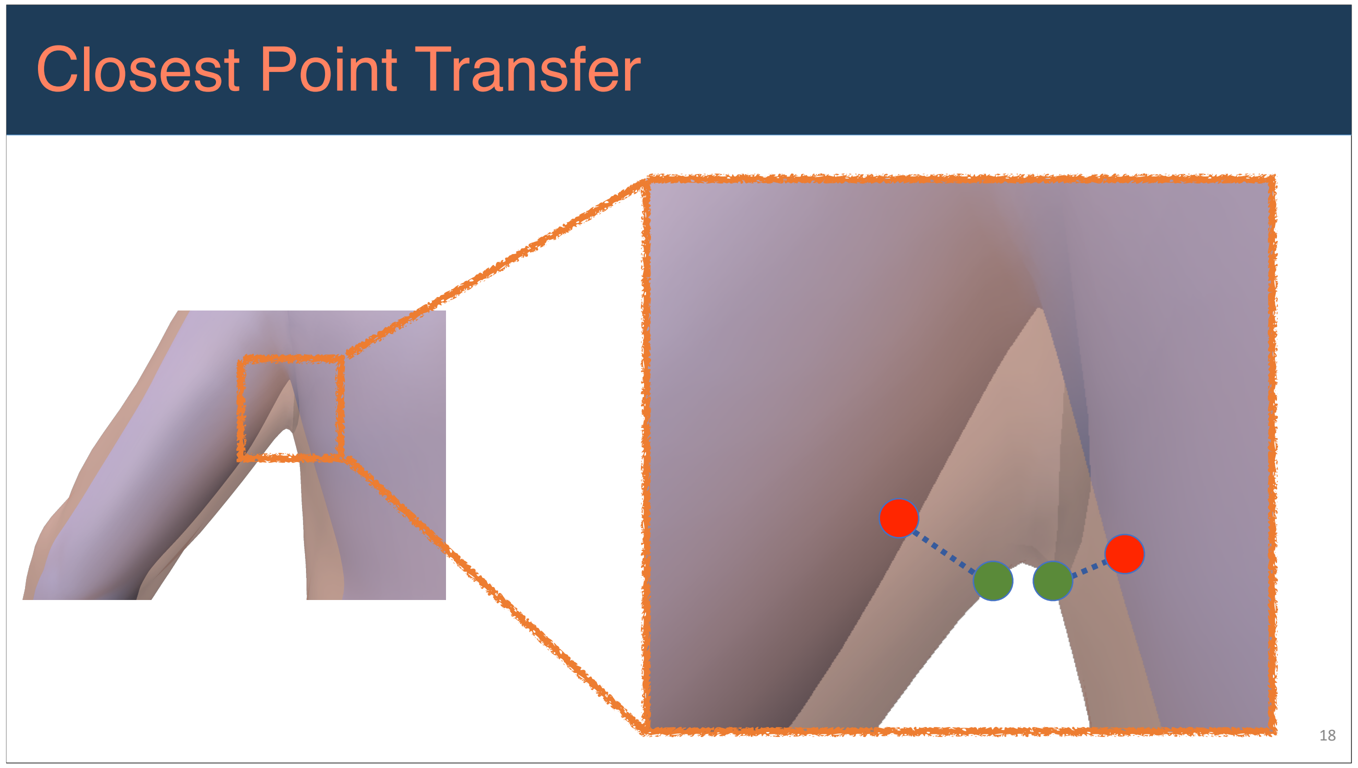
Lets zoom on our armpit area.



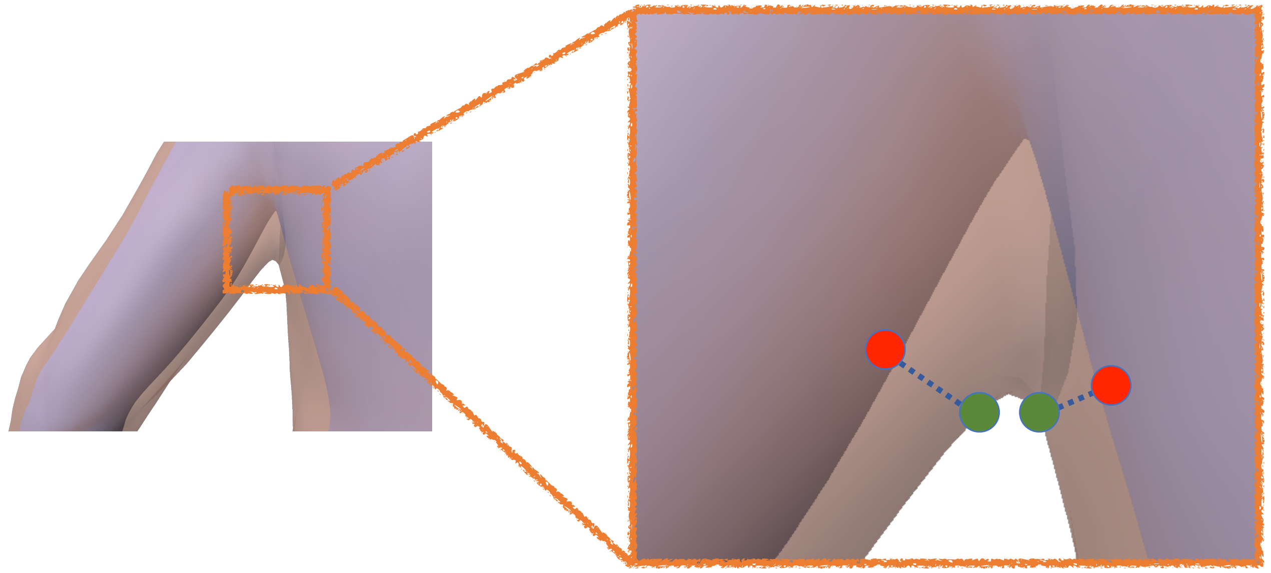
Closest Point Transfer



And lets pick two vertices on the sweater.



Closest Point Transfer



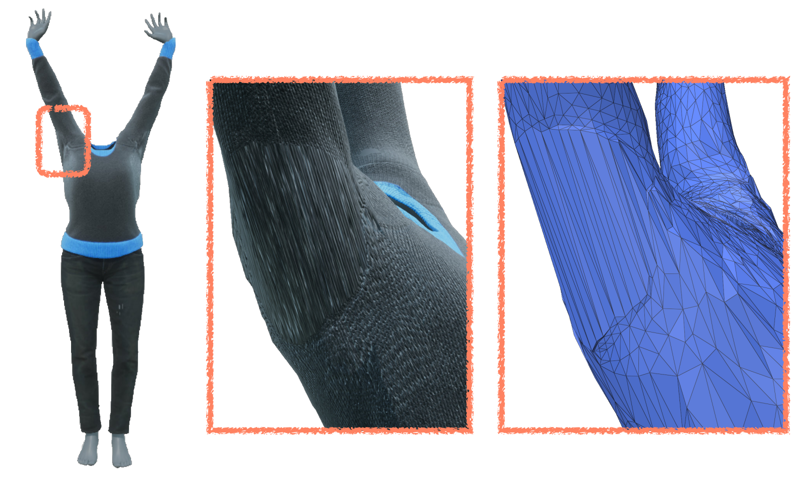
Then find their closest points on the body mesh.

The problem here, is that the points on the clothing are very close to each other in shape space of the sweater but they inherit skinning weights which are largely influenced by different bones.

One of them get the skinning weights from the torso area and another gets the weights from the arm.

And so when play the animation these two vertices end up moving largerly in different directions even though they are close to each other.

Closest Point Transfer



And that’s why we get this stretching

Closest Point Transfer

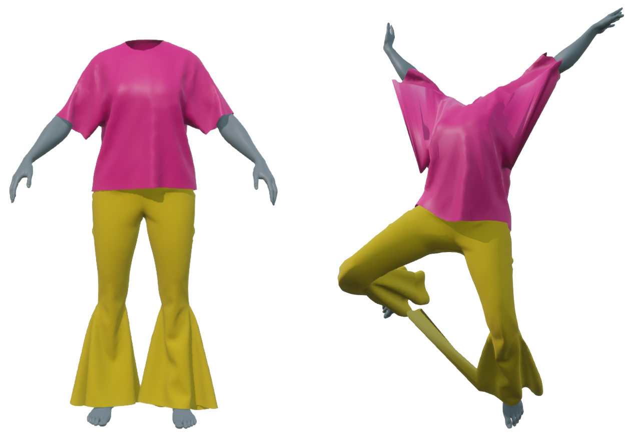
Things get worse for loose clothing.

And you can probably guess that if we already having problems with these simple types of clothings things are only gonna get worse when we are working with more complex/loose clothing.

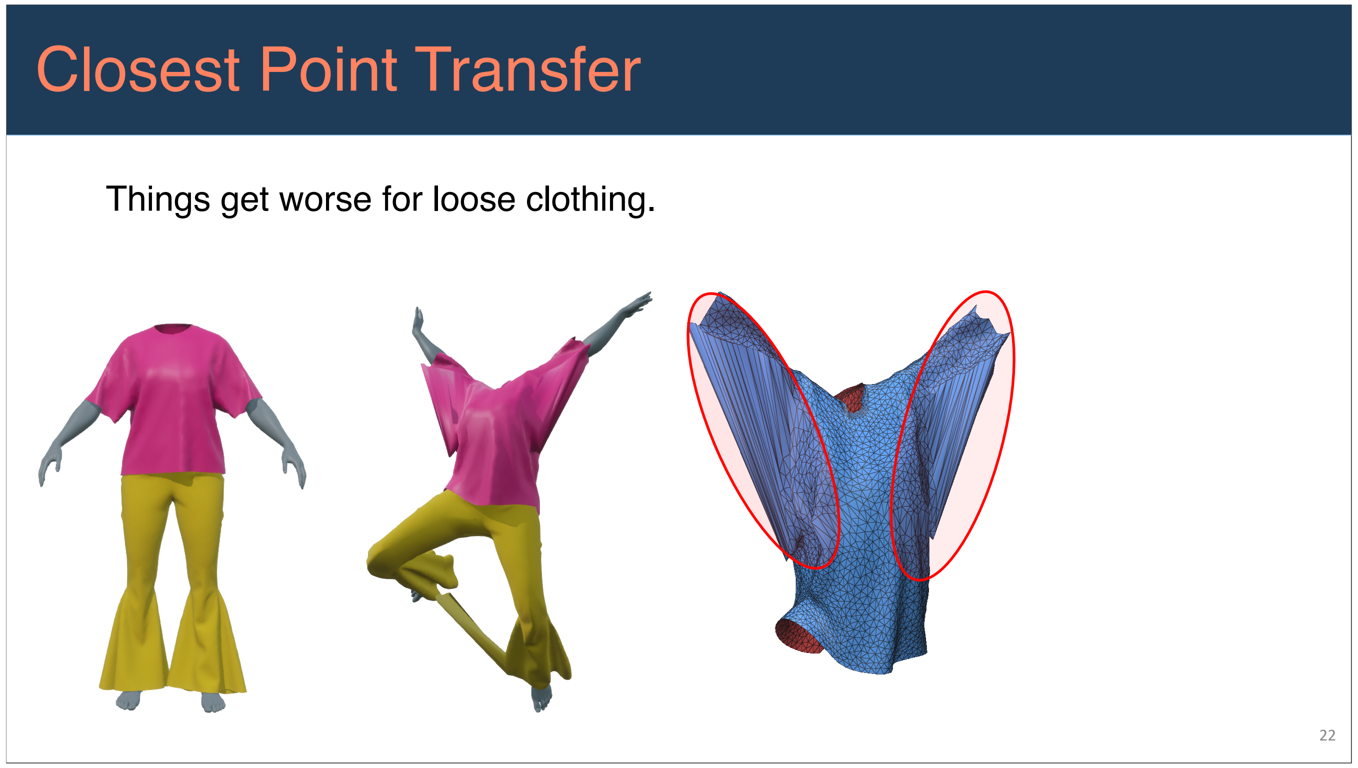
So here we slightly increase the complexity of the garments by having a baggy t-shirt and flared pants.

Closest Point Transfer

Things get worse for loose clothing.

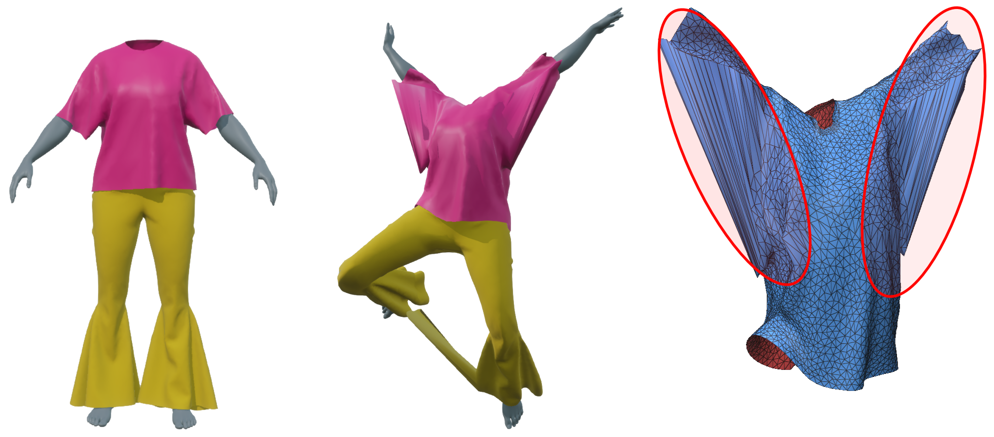


Lets pose it and we see a lot of problems….



Closest Point Transfer

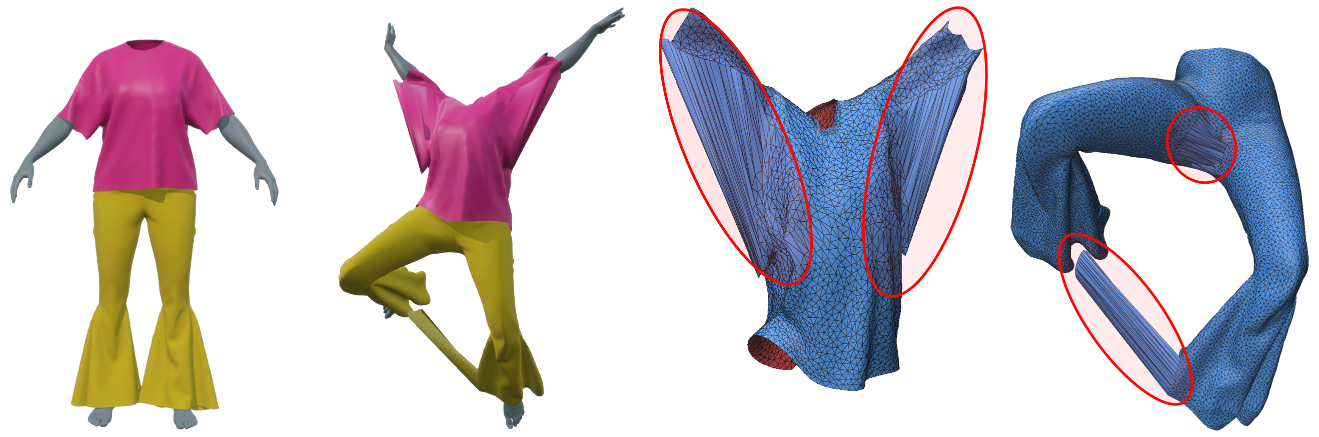
Things get worse for loose clothing.



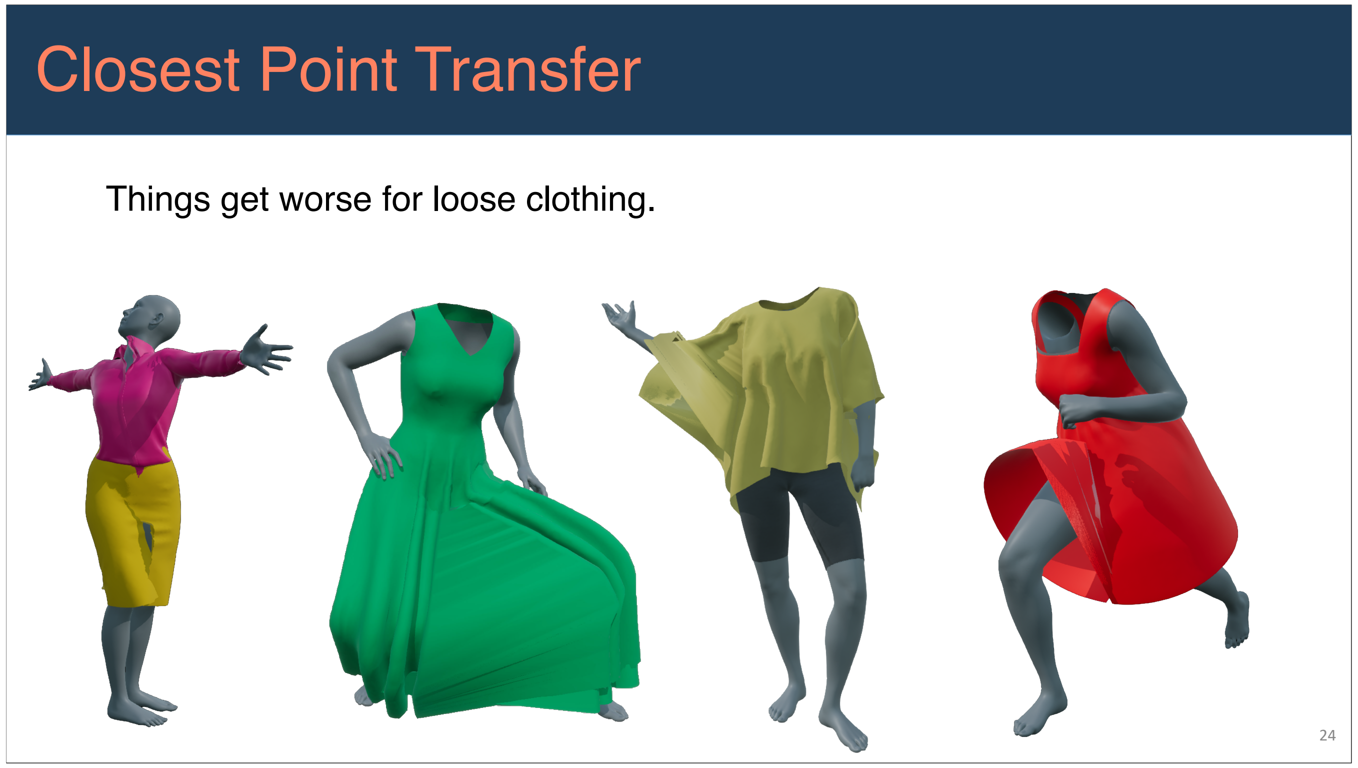
In particular the sleeves and the areas around the armpits have a lot of stretching.

Closest Point Transfer

Things get worse for loose clothing.

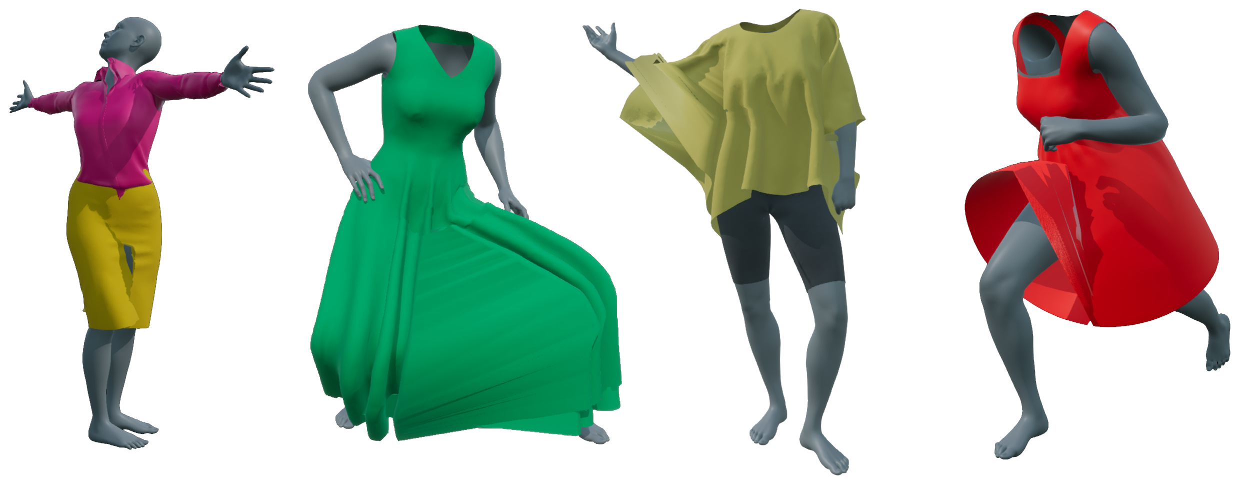


With the pants we have stretching in the crotch area and at the bottom of the pants.



Closest Point Transfer

Things get worse for loose clothing.



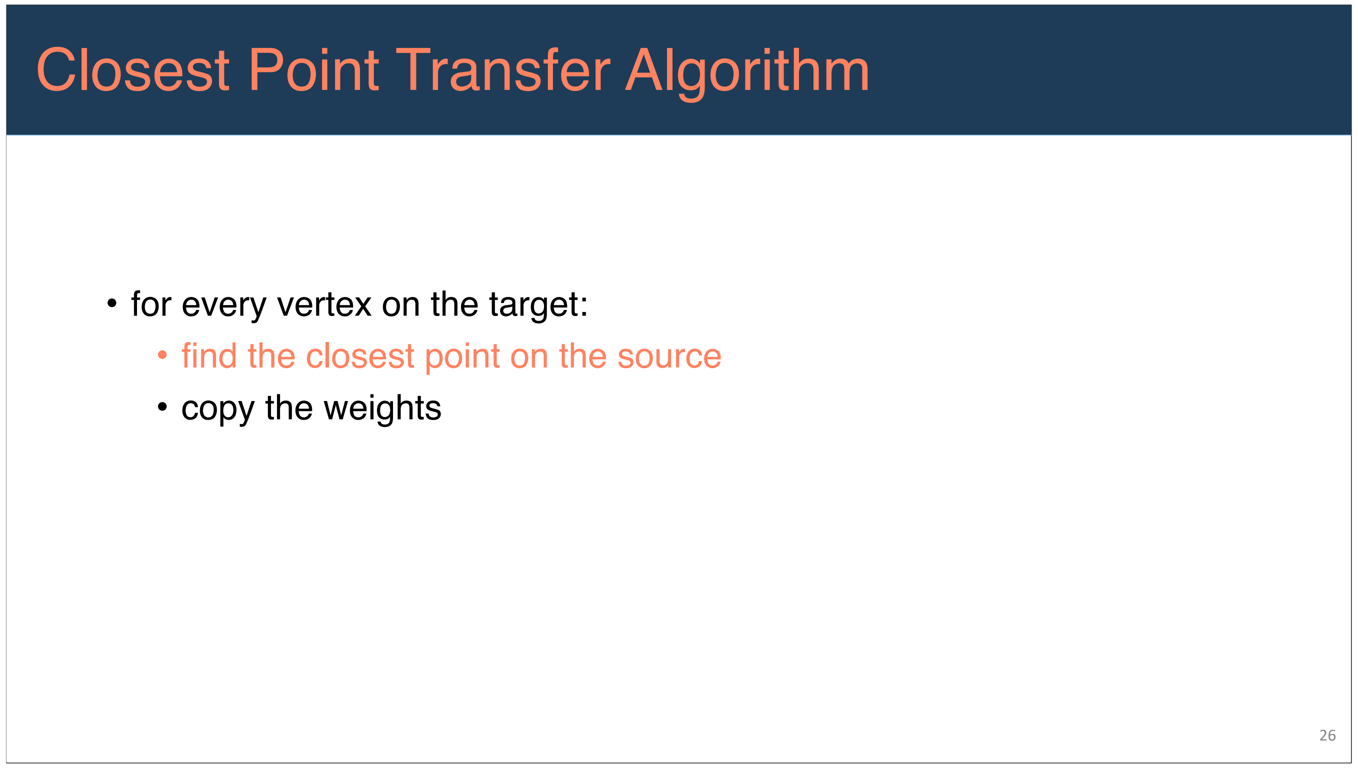
And same story for other garments. Dresses are especially hard to handle.

Closest Point Transfer Algorithm

- for every vertex on the target:
  - find the closest point on the source
  - copy the weights

So hopefully now you understand what are the common issues we encounter with the skin weights transfer and why current solutions do not solve those issues.

Lets go back to our original simple pseudocode, the line that causes our problems is this line….



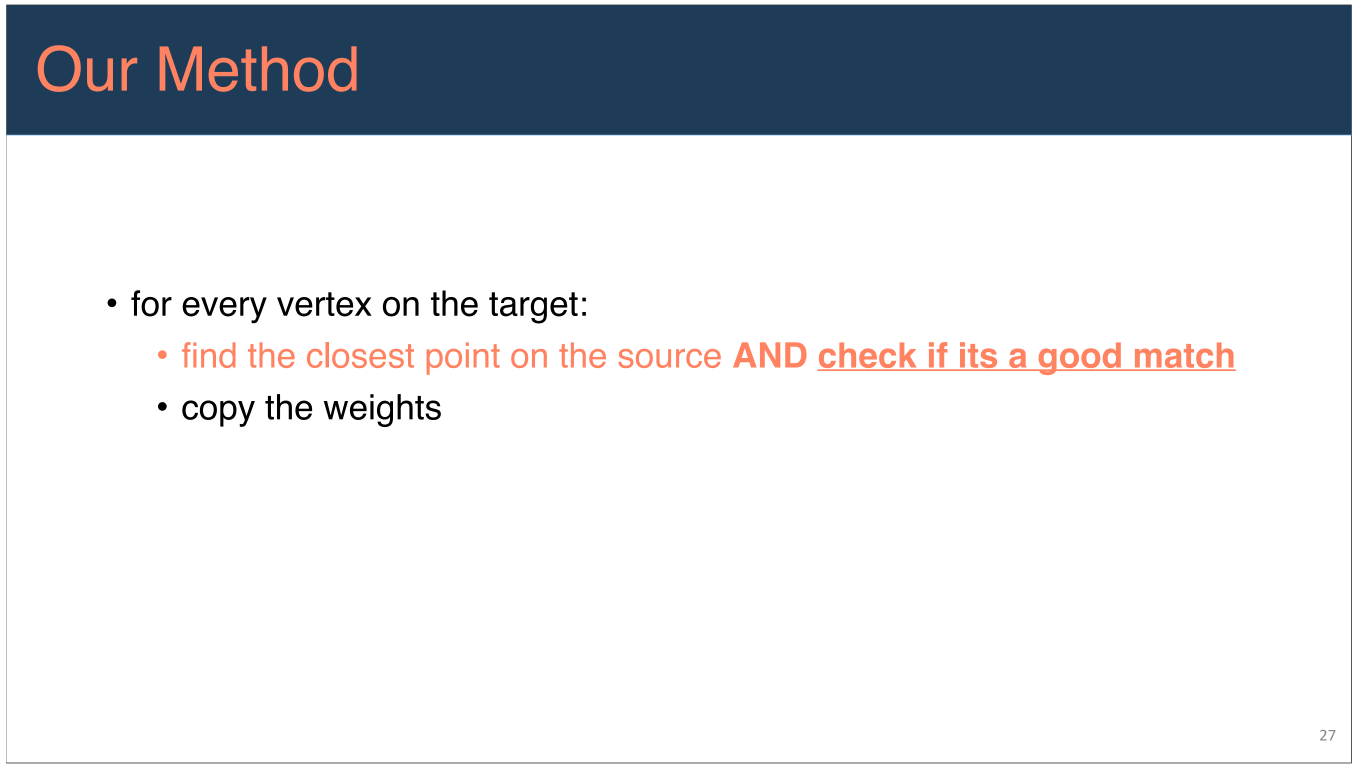
Closest Point Transfer Algorithm

- for every vertex on the target:
  - find the closest point on the source
  - copy the weights

“We find the closest point on the source…”

This implies that we are always forced to find a match on the source because we have to get weights from somewhere.

But as we saw from my previous examples not all matches are good.



Our Method

- for every vertex on the target:
  - find the closest point on the source **AND <u>check if its a good match</u>**
  - copy the weights

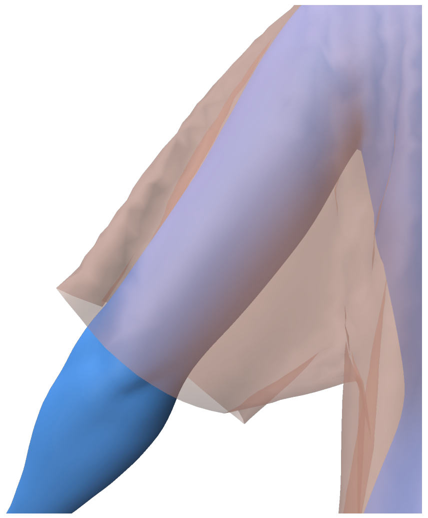
This is where our contribution starts comes in

We ask ourselves a question: What if we were more selective about finding the matches? Meaning, not every point on the target mesh will find a match on the source mesh. And thats ok! We will deal with them later.

Then the question is, what do we consider a good match?

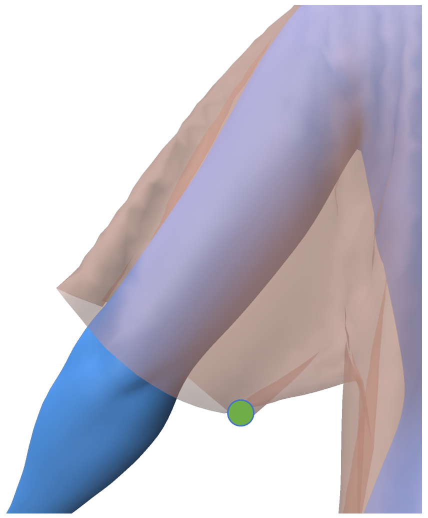
What is a good match?

Distance



What is a good match?

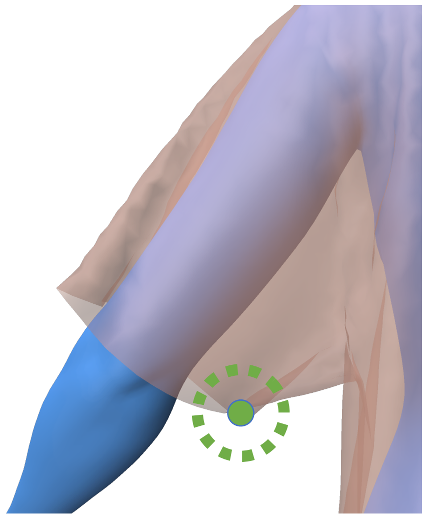
Distance



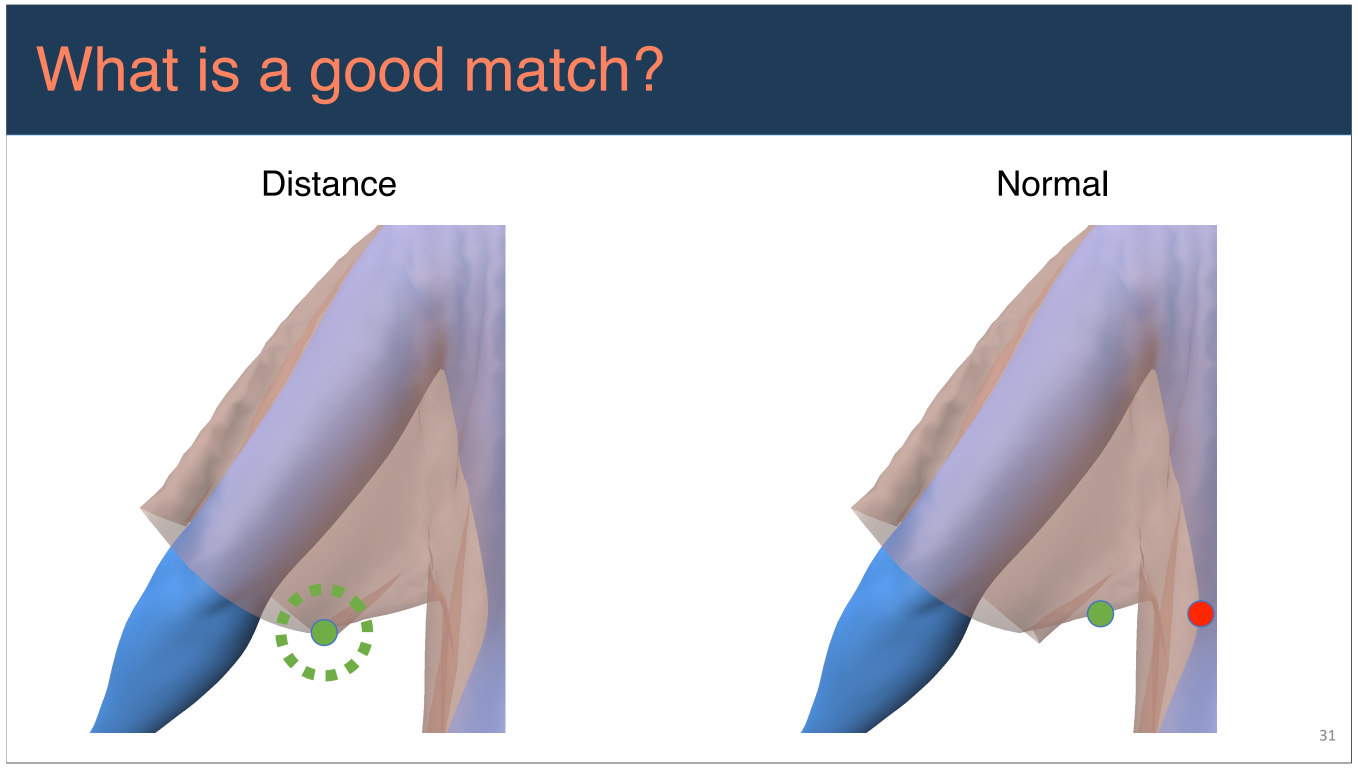
First, given some vertex on the target mesh…

What is a good match?

Distance



…we should limit the search radius for the closest point on the source mesh. Unless there is a point near by we can’t reliably assume that its a good match.



What is a good match?

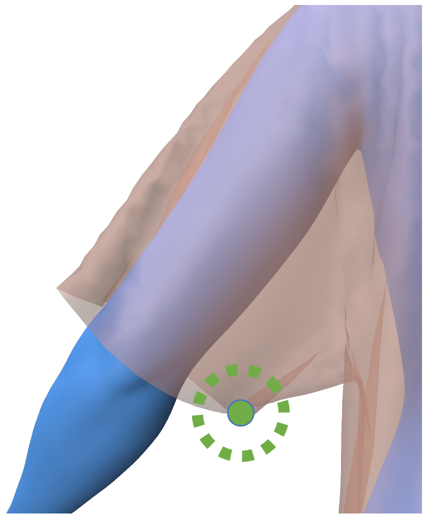
Distance

Normal

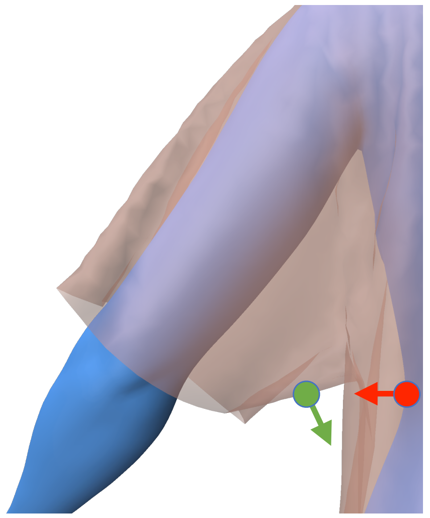
But if we do find a point near by, we should also check that the surfaces around our target vertex and that point are locally similar.

What is a good match?

Distance

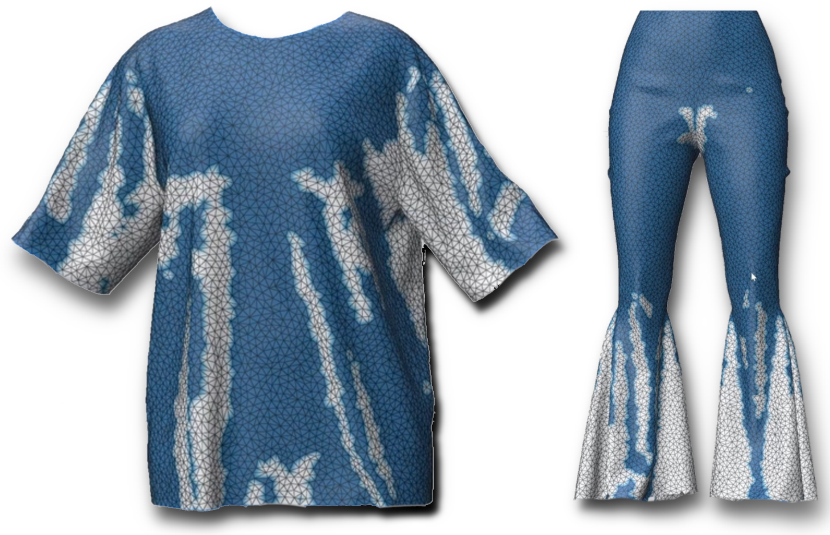


Normal



We measure that similarity, by simply checking if their normals do not deviate by more than some angle threshold.

What is a good match?

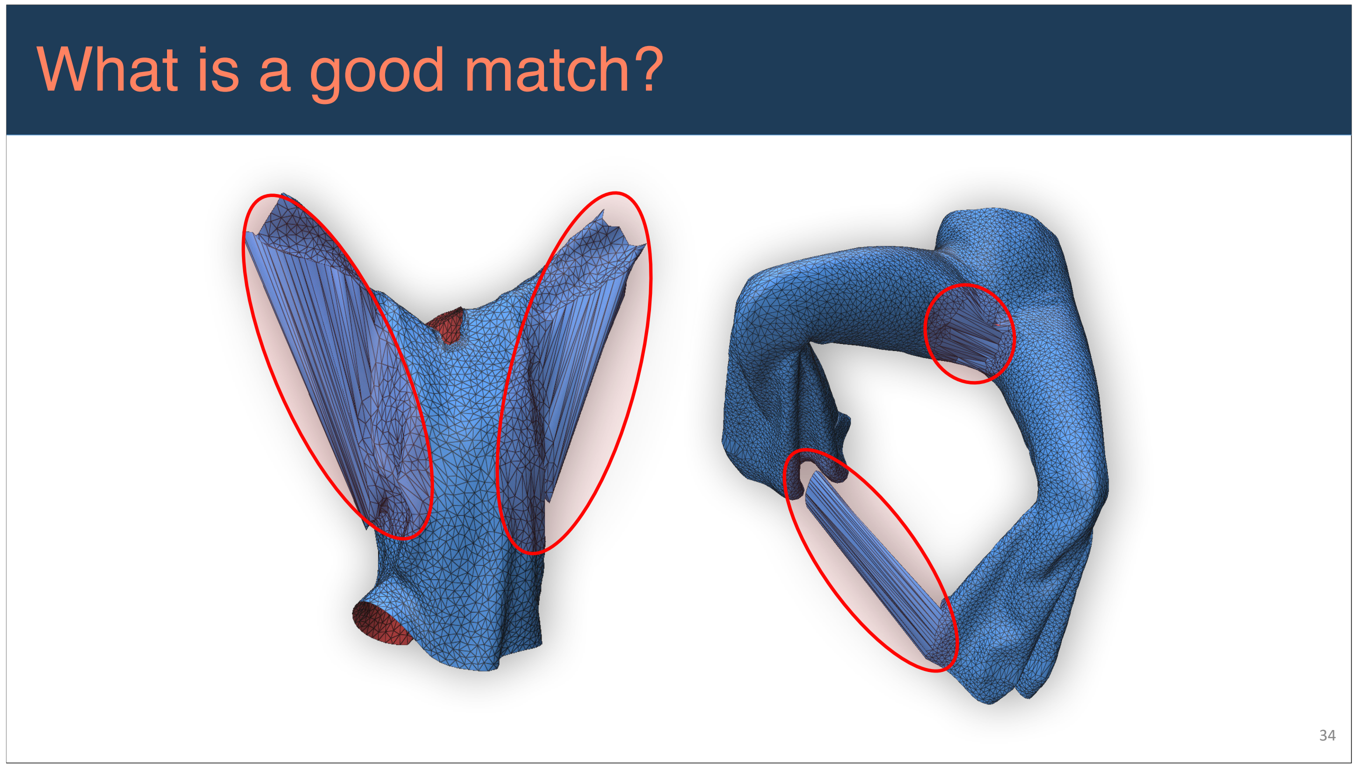


🟦 Match

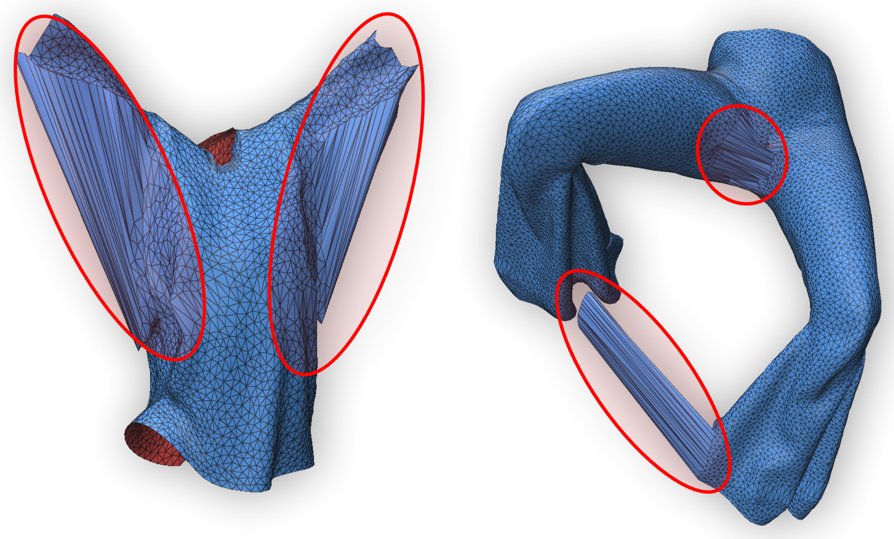
⬜ No Match

So lets run this algorithm and see what we get.

I am showing a binary mask where in blue are the vertices for which a good match was found on the source mesh and in white all the vertices without a match. Meaning either distance or normal check failed.



What is a good match?

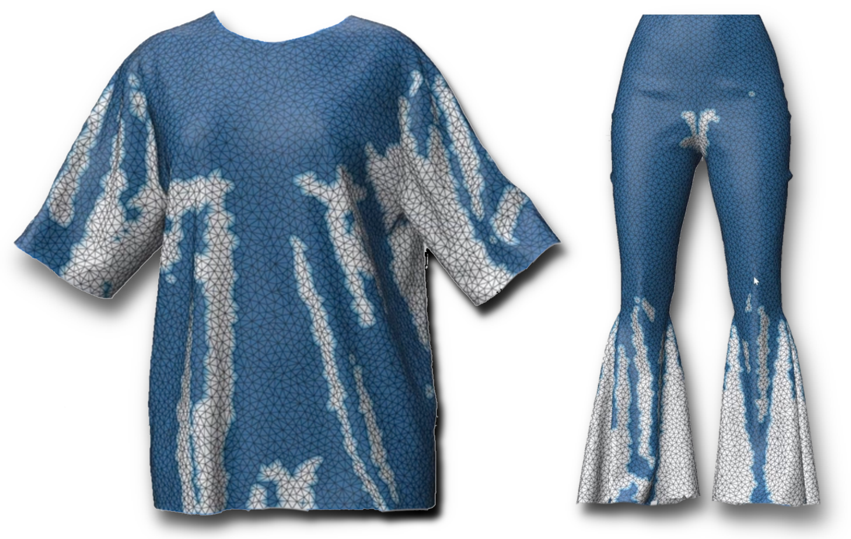


34

Lets quickly refresh our memory. Which areas gave us trouble?

The sleeves
The armpits
The crotch
The bottom of the pants

What is a good match?



■ Match

□ No Match

And for which areas are we rejecting the match?

The sleeves
The armpits
The crotch
The bottom of the pants

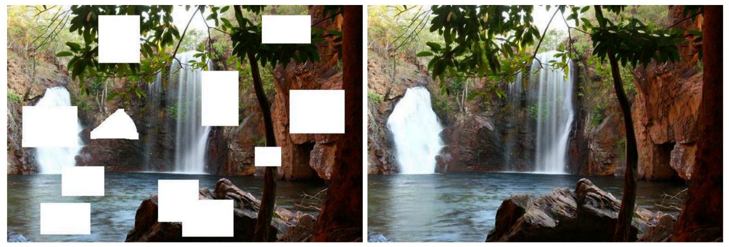
So although our criteria is not complex, we are just comparing distances and comparing normals, it gets the job done.

So for all the vertices where we found a good match we are just gonna copy weights from their closest points, same as before.

But now we are left with vertices without a match for which we do not know the weights. So then question is how do we compute their weights?

Inpainting

Image inpainting



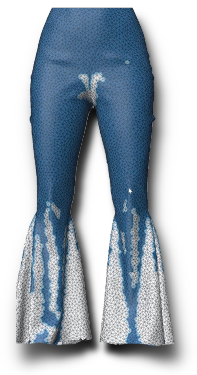
Generative Image Inpainting with Contextual Attention
Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, Thomas S. Huang
CVPR 2018

We take inspiration from inpainting techniques. For example, image inpainting tries to predict pixel colours in the missing regions based on the surrounding pixel data available.

Inpainting

Image inpainting

Skin weights inpainting



Generative Image Inpainting with Contextual Attention
Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, Thomas S. Huang
CVPR 2018

In our case, we know that for the vertices coloured in blue we have a high confidence of having ground truth weights because they passed our selection tests.

And we want to “inpaint” the skinning weights in the white regions, and hope we get as close as possible to artist painted weights.

Method

E(, W)

So we formulate this “inpainting” process as an optimization problem.

We have some energy function defined on the target mesh and the skinning weights matrix W.

Method

$E($ $, W)$

$W \in \mathbb{R}^{n \times b}$

skinning weights

which is an n by b matrix, where n is the number of vertices and b is the total number of bones.

Method

E(, W)

$\mathrm{W} \in \mathbb{R}^{n \times b}$

skinning weights

$\mathrm{W}(i, j)$

influence of bone **j** on vertex **i**

and the i,j entry is the influence of the bone j on the vertex i

Method

$\underset{W}{\text{minimize}}$ E(, W)

$W \in \mathbb{R}^{n \times b}$

skinning weights

$W(i, j)$

influence of bone **j** on vertex **i**

and we want find the values of the matrix W that minimize our energy function

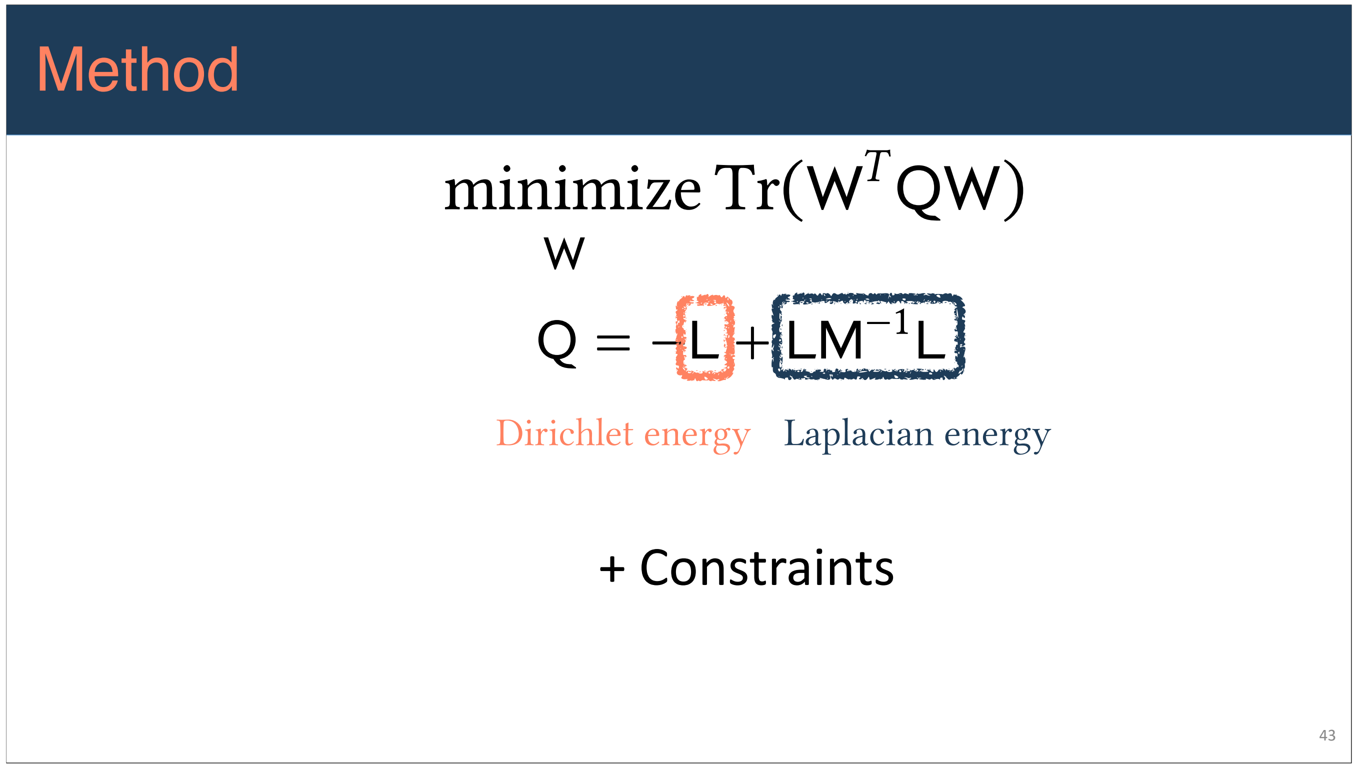
Method

$$\underset{W}{\text{minimize}}\ \mathrm{Tr}(W^T Q W)$$

$$Q = -\underset{\text{Dirichlet energy}}{\boxed{L}} + \underset{\text{Laplacian energy}}{\boxed{LM^{-1}L}}$$

The energy we minimizing is a combination of Dirchlet (−L) and squared-Laplacian (LM^{−1}L) energies where
L is the cotangent Laplacian, and M is the diagonal lumped mass matrix.

In our experiments, this combination gives the closest approximation to the artist-painted weights.



Method

$$\underset{W}{\text{minimize}}\ \mathrm{Tr}(W^T Q W)$$

$$Q = -\underset{\text{Dirichlet energy}}{\boxed{L}} + \underset{\text{Laplacian energy}}{\boxed{LM^{-1}L}}$$

+ Constraints

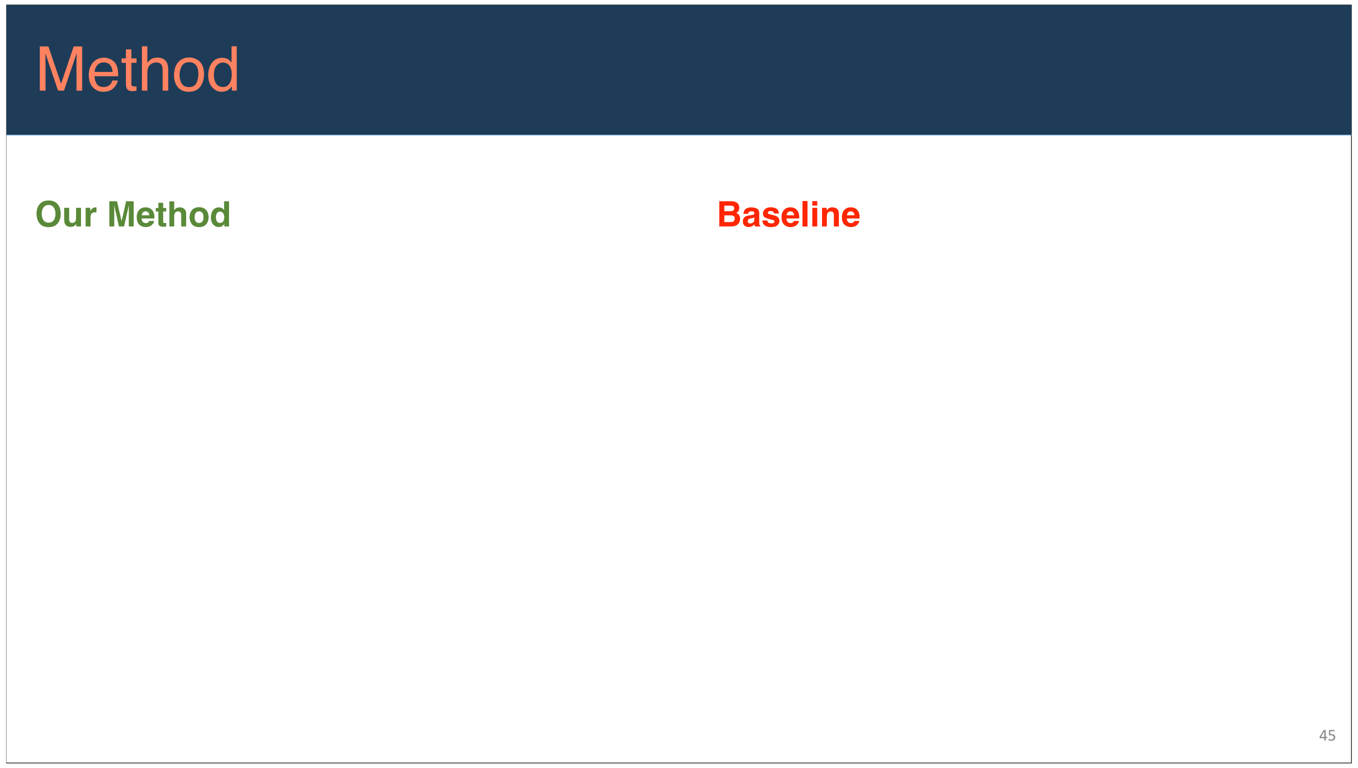
And we add some constraints to out optimization but for details please refer to the paper.

Method

$$Aw = b$$

Whats important is that in the end, after all the simplifications, what we actually solve is a simple linear system where our matrix will be positive semi definite, so we can solve this efficiently using the Cholesky decomposition.

And the vector w is our unknown weights for a **single** bone. And we would need to solve this for each bone that has a non-zero influence on the in-painted regions.

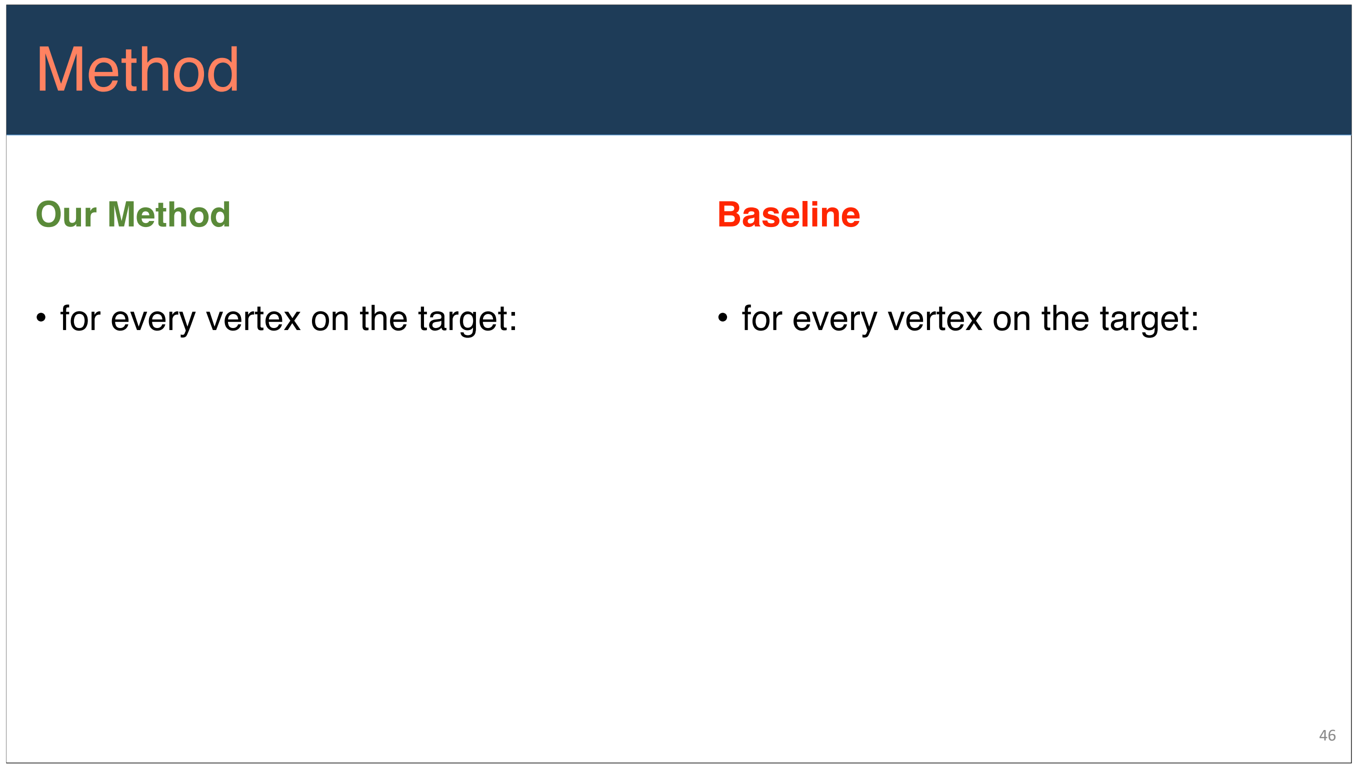


Method

Our Method

Baseline

Now that we have all the major pieces, lets write our core algorithm and compare it side by side to the baseline.



Method

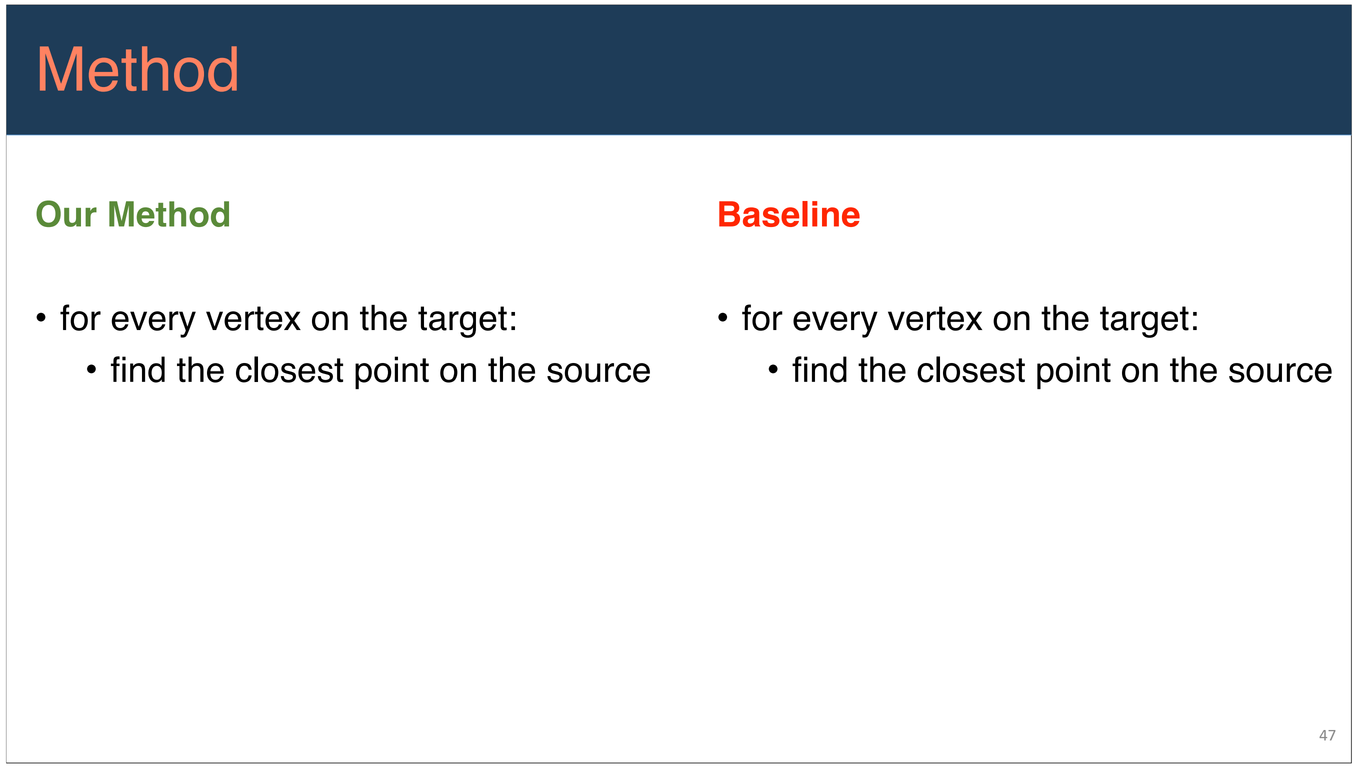
Our Method

- for every vertex on the target:

Baseline

- for every vertex on the target:

[just read the slides]



Method

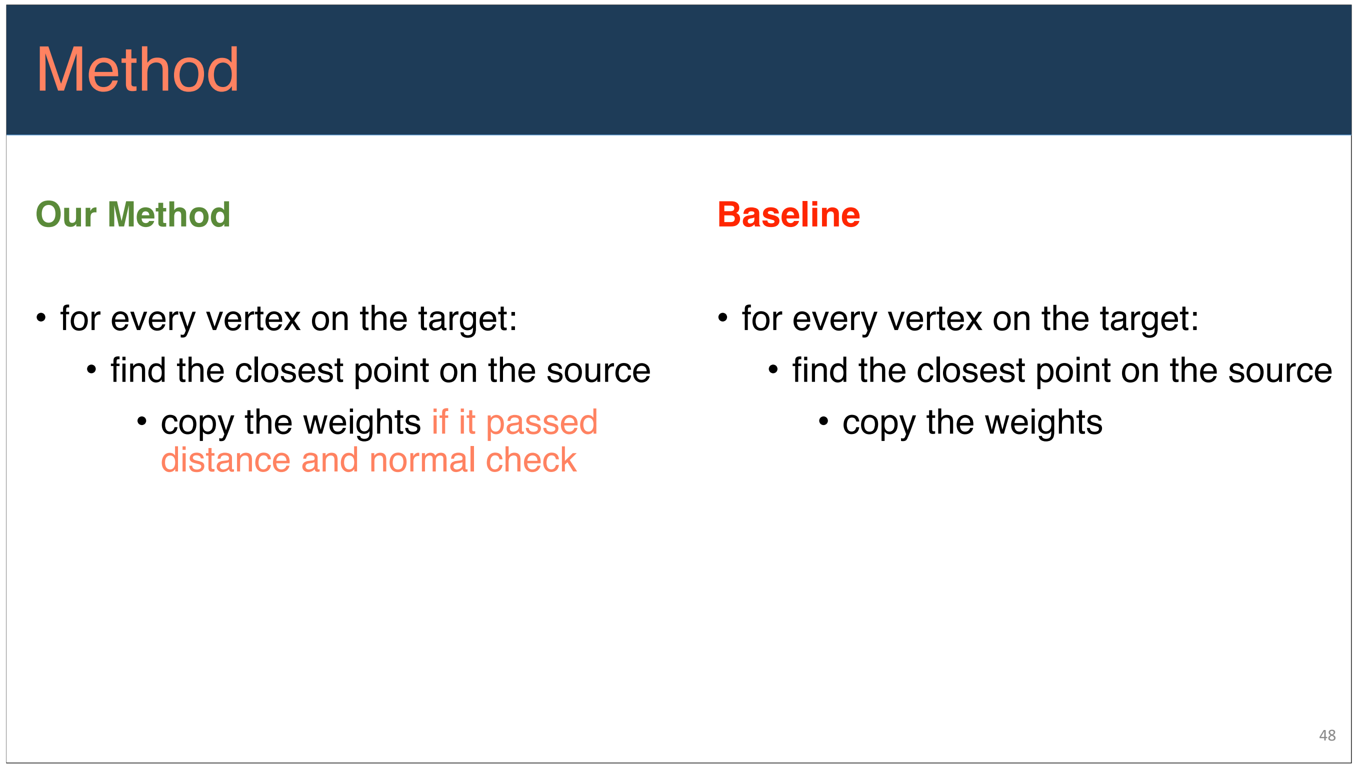
Our Method

- for every vertex on the target:
    - find the closest point on the source

Baseline

- for every vertex on the target:
    - find the closest point on the source

[just read the slides]



Method

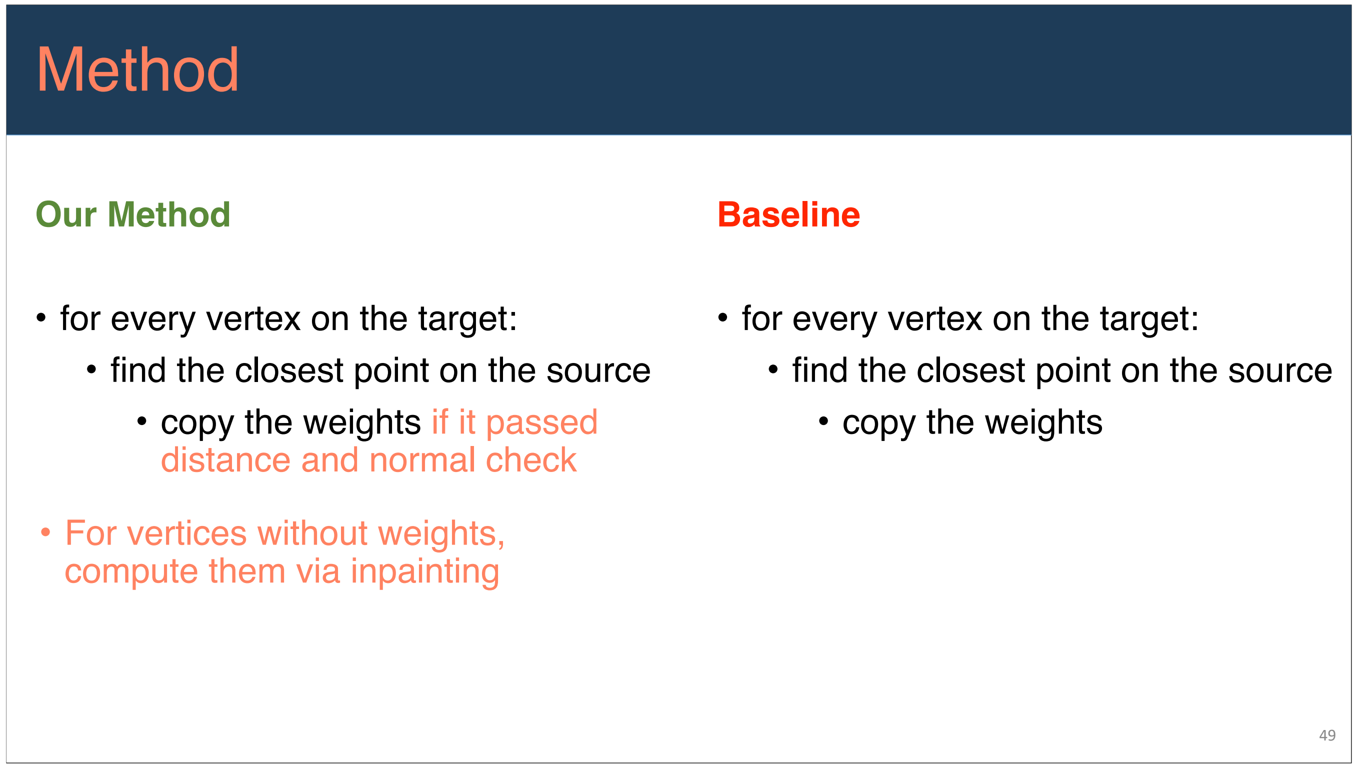
Our Method

- for every vertex on the target:
  - find the closest point on the source
    - copy the weights if it passed distance and normal check

Baseline

- for every vertex on the target:
  - find the closest point on the source
    - copy the weights

[just read the slides]



Method

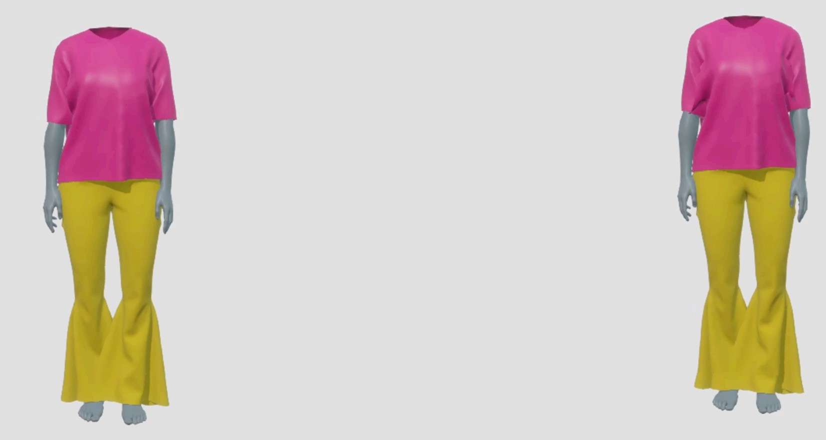
Our Method

- for every vertex on the target:
  - find the closest point on the source
    - copy the weights if it passed distance and normal check
- For vertices without weights, compute them via inpainting

Baseline

- for every vertex on the target:
  - find the closest point on the source
    - copy the weights

[just read the slides]

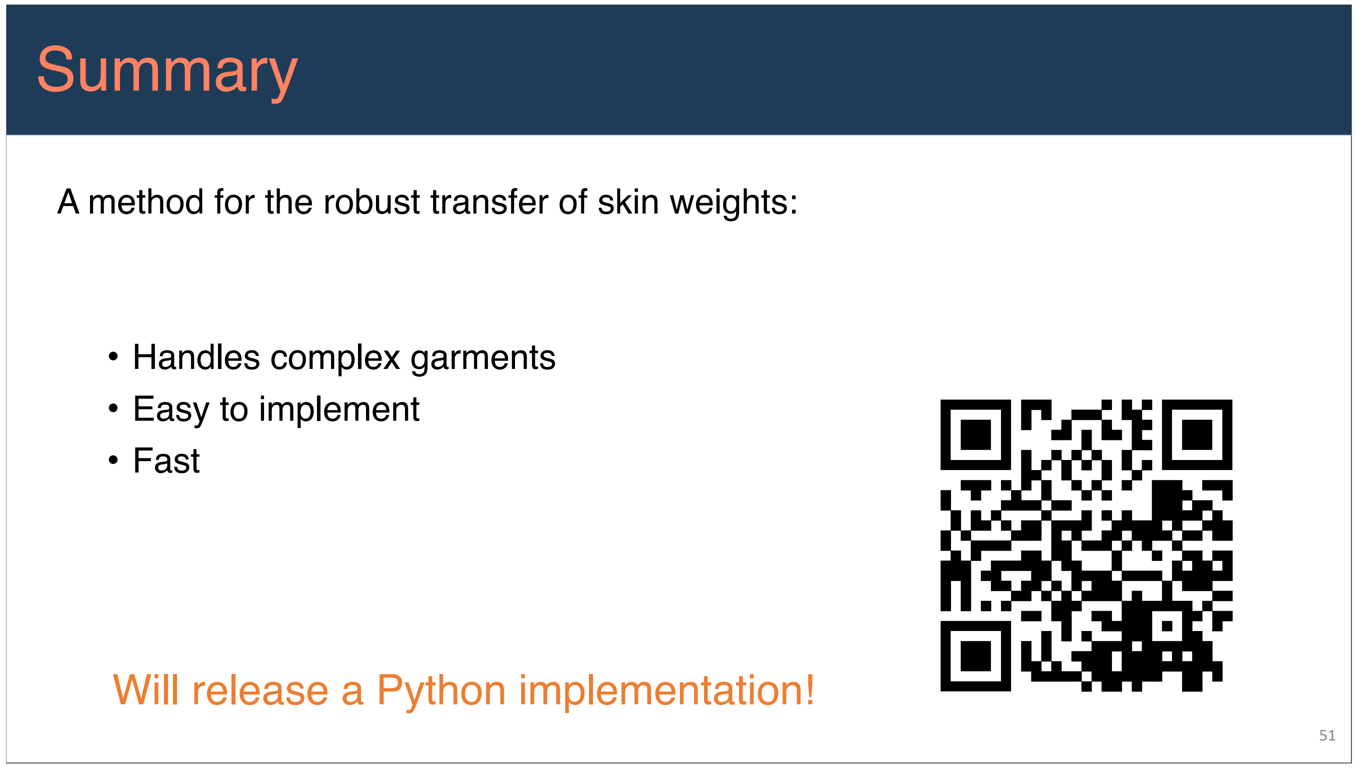


So the result our method that you see on your screens is generated automatically, there is no artist post-cleanup.

If you ever used Copy Weights Tool in Maya for example, the Baseline results would be very similar.

And we implemented this algorithm in the Unreal Engine as part of the experimental chaos cloth editor tool which is used for real-time cloth simulation. And since we don’t have any way to paint skin weight in that tool, we use this algorithm to initialize skinning weights for any garment you import into it.

Also if you want to find you more about how we handle collars which are a bit tricky and also multilayered clothing please refer to the paper.



Summary

A method for the robust transfer of skin weights:

- Handles complex garments
- Easy to implement
- Fast

Will release a Python implementation!

51

In summary… [read the slide]
We presented a method for robust skin weights transfer

SIGGRAPH ASIA 2023 SYDNEY

Robust Skin Weights Transfer via Weight Inpainting

Rinat Abdrashitov	Epic Games
Kim Raichstat	Epic Games
Jared Monsen	Epic Games
David Hill	Epic Games

Connecting STORIES

The 16th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia

CONFERENCE 12 - 15 December 2023
EXHIBITION 13 - 15 December 2023
ICC, Sydney, Australia

ASIA.SIGGRAPH.ORG/2023

Sponsored by

Organized by koelnmesse

And with that I conclude my talk. Thank you!