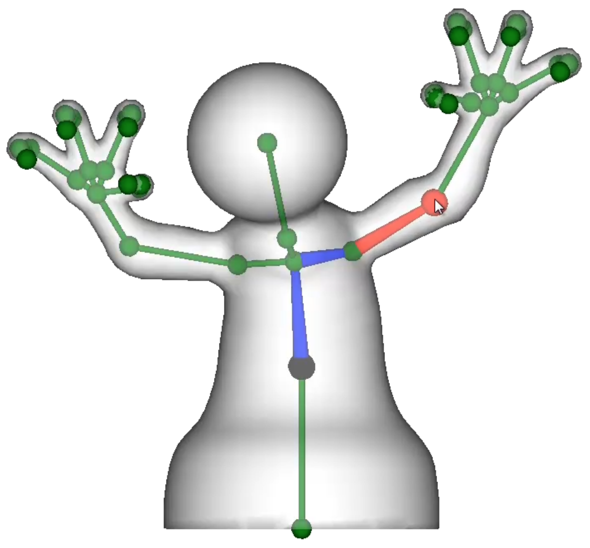
RigMesh: Automatic Rigging for Part-Based Shape Modeling and Deformation

Péter Borosán
Ming Jin
Doug DeCarlo
Yotam Gingold
Andrew Nealen

RUTGERS THE STATE UNIVERSITY OF NEW JERSEY
COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK
NYU-poly POLYTECHNIC INSTITUTE OF NYU
GEORGE MASON UNIVERSITY



Sponsored by ACM SIGGRAPH

SIGGRAPH ASIA2012

Thank you for the introduction.

I am Ming Jin, and this is Peter Borosan.

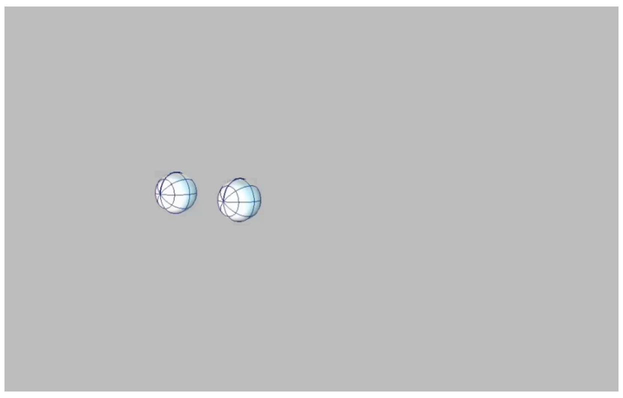
Today we are going to show you RigMesh, a system that unifies the modeling and rigging stages in state-of-the-art 3D modeling and animation pipeline.

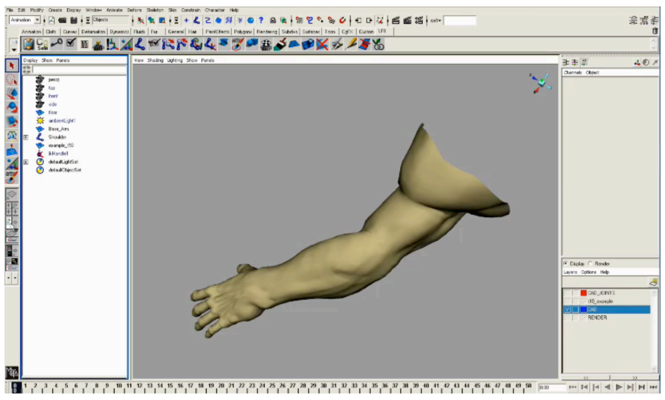
We can create complex shapes with just a few basic operations of sketching, cutting and merging

And better yet, the shapes can be deformed at any step of the modeling process.

[click]

Problem





Sponsored by ACM SIGGRAPH

Let's start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it's been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

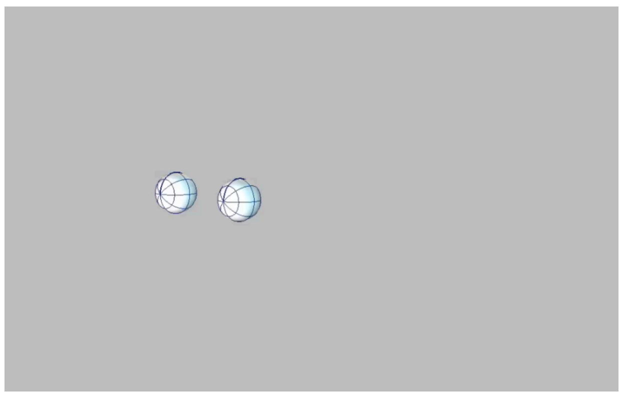
[click]

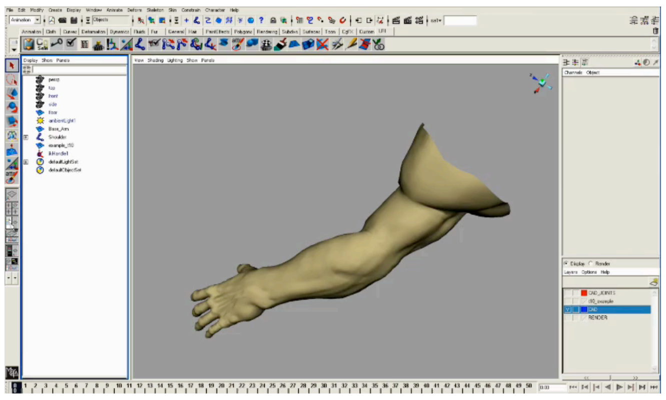
The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

Problem

- Creating ready-to-animate 3D models is **hard**





Sponsored by ACM SIGGRAPH

Let's start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it's been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

[click]

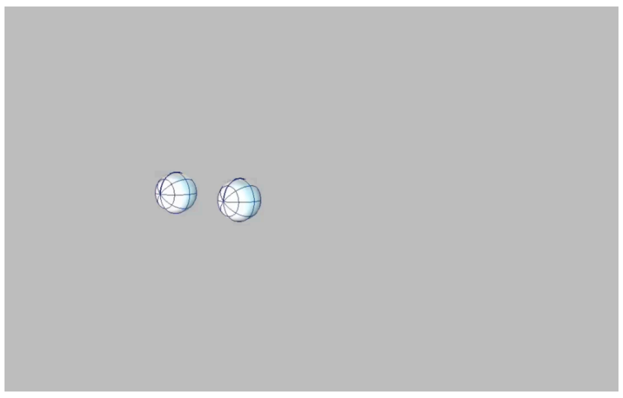
The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

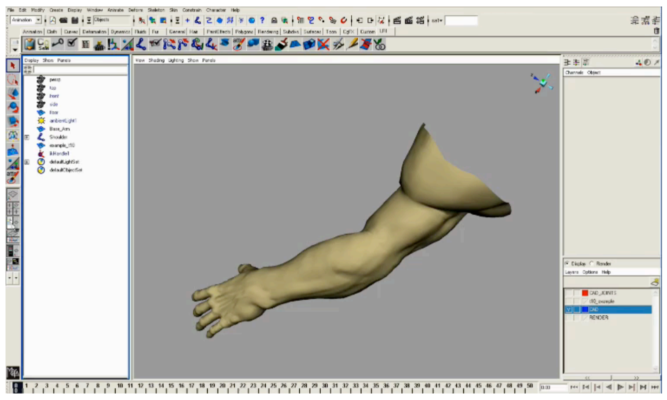
Problem

- Creating ready-to-animate 3D models is **hard**

Modeling



[youtube user: MasahiroUshiyama]



Sponsored by ACM SIGGRAPH

Let’s start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it’s been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

[click]

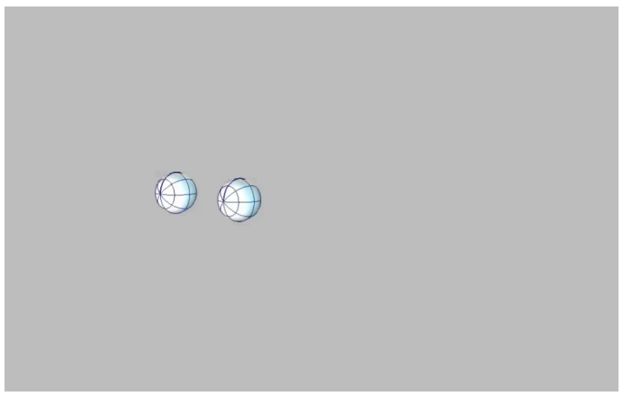
The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

Problem

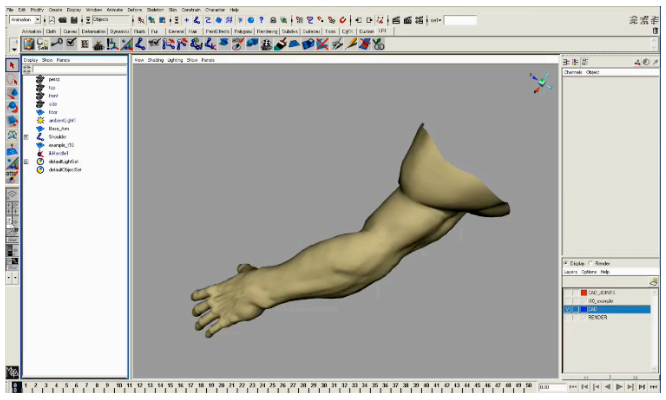
- Creating ready-to-animate 3D models is **hard**

Modeling



[youtube user: MasahiroUshiyama]

Rigging



[Weber et al. 07]

Let’s start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it’s been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

[click]

The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

Problem

- Creating ready-to-animate 3D models is **hard**

Let’s start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it’s been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

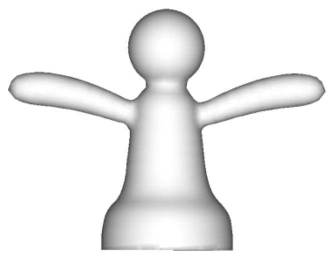
[click]

The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

Problem

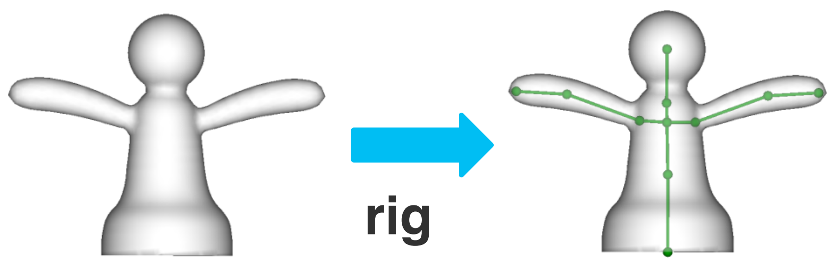
- Creating ready-to-animate 3D models is **hard**



Let's start by looking at the problem in the state-of-the-art and why we think this work is important.
[click]
First of all, creating ready-to-animate 3D models is fundamentally hard.
This whole process is broken down into modeling and rigging.
[click]
In professional modeling tools, modeling the surfaces is usually performed manually.
The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.
[click]
Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.
It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.
[click]
Here is an overview of the pipeline in a nutshell.
[click]
The artist creates a 3D model
[click]
She rigs the model by designing the skeleton and painting the skin weights
[click]
The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.
But what if the animator wants to change the model after it's been rigged?
[click]
For example, she wants to change the puffy hand into a claw?
[click]
Any changes like this will invalidate the existing rig
[click]
Requiring that the model be fed back into the pipeline again
[click]
The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially
[click]

Problem

- Creating ready-to-animate 3D models is **hard**



Let’s start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it’s been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

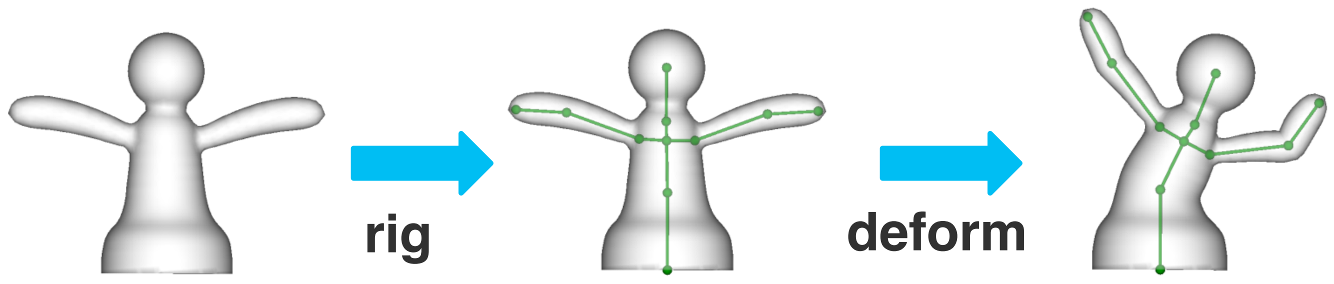
[click]

The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

Problem

- Creating ready-to-animate 3D models is **hard**



Let’s start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it’s been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

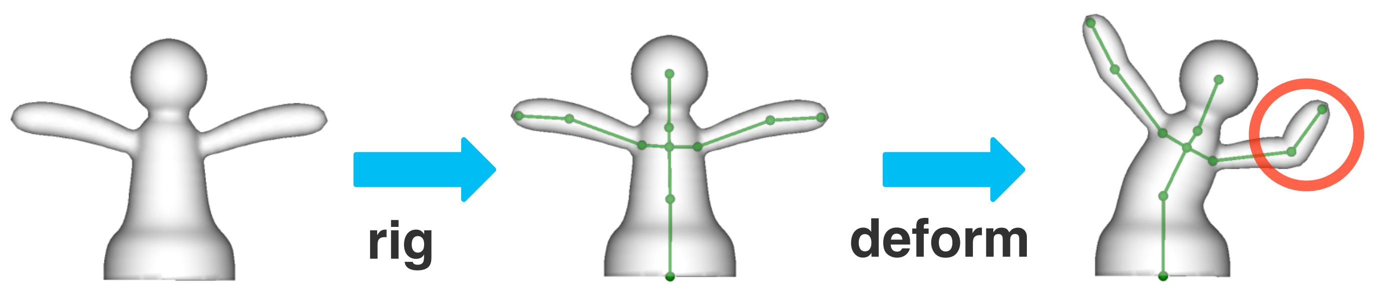
[click]

The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

Problem

- Creating ready-to-animate 3D models is **hard**



Let's start by looking at the problem in the state-of-the-art and why we think this work is important.
[click]
First of all, creating ready-to-animate 3D models is fundamentally hard.
This whole process is broken down into modeling and rigging.
[click]
In professional modeling tools, modeling the surfaces is usually performed manually.
The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.
[click]
Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.
It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.
[click]
Here is an overview of the pipeline in a nutshell.
[click]
The artist creates a 3D model
[click]
She rigs the model by designing the skeleton and painting the skin weights
[click]
The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.
But what if the animator wants to change the model after it's been rigged?
[click]
For example, she wants to change the puffy hand into a claw?
[click]
Any changes like this will invalidate the existing rig
[click]
Requiring that the model be fed back into the pipeline again
[click]
The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially
[click]

Problem

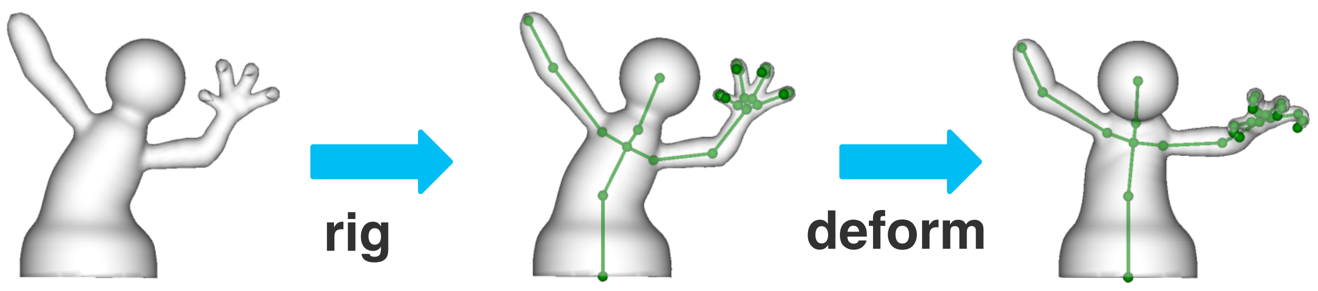
- Creating ready-to-animate 3D models is **hard**

→ **rig** → **deform**

Let’s start by looking at the problem in the state-of-the-art and why we think this work is important.
[click]
First of all, creating ready-to-animate 3D models is fundamentally hard.
This whole process is broken down into modeling and rigging.
[click]
In professional modeling tools, modeling the surfaces is usually performed manually.
The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.
[click]
Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.
It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.
[click]
Here is an overview of the pipeline in a nutshell.
[click]
The artist creates a 3D model
[click]
She rigs the model by designing the skeleton and painting the skin weights
[click]
The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.
But what if the animator wants to change the model after it’s been rigged?
[click]
For example, she wants to change the puffy hand into a claw?
[click]
Any changes like this will invalidate the existing rig
[click]
Requiring that the model be fed back into the pipeline again
[click]
The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially
[click]

Problem

- Creating ready-to-animate 3D models is **hard**



Let's start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it's been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

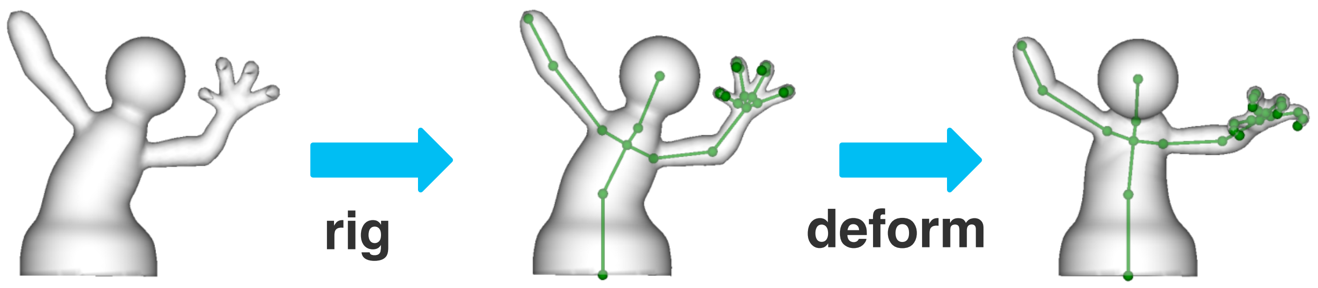
[click]

The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

Problem

- Creating ready-to-animate 3D models is **hard**



- Fundamental problem: static sequential pipeline

Let's start by looking at the problem in the state-of-the-art and why we think this work is important.

[click]

First of all, creating ready-to-animate 3D models is fundamentally hard.

This whole process is broken down into modeling and rigging.

[click]

In professional modeling tools, modeling the surfaces is usually performed manually.

The artist starts with coarse subdivision surfaces or parametric patches and works out all the details.

[click]

Rigging is another process in which the artist has to design the skeleton and specify its mappings to the model.

It is a laborious process which takes a lot of tries and fails until finally a satisfactory design is achieved.

[click]

Here is an overview of the pipeline in a nutshell.

[click]

The artist creates a 3D model

[click]

She rigs the model by designing the skeleton and painting the skin weights

[click]

The resulting shape is fully rigged and ready for skin deformations by skeletons. Everything seems fine.

But what if the animator wants to change the model after it's been rigged?

[click]

For example, she wants to change the puffy hand into a claw?

[click]

Any changes like this will invalidate the existing rig

[click]

Requiring that the model be fed back into the pipeline again

[click]

The fundamental problem here is the static pipeline where modeling and rigging have to be performed sequentially

[click]

Sponsored by ACM SIGGRAPH

Recent advancement in the literature has provided various ways to simplify both modeling and rigging.

Noticeably, sketch-based 3D modeling has been a popular research field. It provides an easy-to-use interface and enables rapid iterative prototyping.

[click]

For example, the seminal paper on Teddy describes the creation of freeform 3D models from scratch.

[click]

As extensions to such freeform modelers, other researchers have proposed to use variational implicit functions for the underlying shape representations, such as the BlobTree used in ShapeShop.

[click]

Later, FiberMesh demonstrated how to integrate 3D control curves into sketch-based modeling tools. The curves on the surfaces serve as handles for controlling the geometry.

[click]

Because a single sketch usually does not suffice for creating a complex shape, there has also been interesting work on part based modeling, such as layered surface composition using Surface Trees.

[click]

Sketch-based Modeling

Sponsored by ACM SIGGRAPH

Recent advancement in the literature has provided various ways to simplify both modeling and rigging.

Noticeably, sketch-based 3D modeling has been a popular research field. It provides an easy-to-use interface and enables rapid iterative prototyping.

[click]

For example, the seminal paper on Teddy describes the creation of freeform 3D models from scratch.

[click]

As extensions to such freeform modelers, other researchers have proposed to use variational implicit functions for the underlying shape representations, such as the BlobTree used in ShapeShop.

[click]

Later, FiberMesh demonstrated how to integrate 3D control curves into sketch-based modeling tools. The curves on the surfaces serve as handles for controlling the geometry.

[click]

Because a single sketch usually does not suffice for creating a complex shape, there has also been interesting work on part based modeling, such as layered surface composition using Surface Trees.

[click]

Sketch-based Modeling

- Teddy [Igarashi et al. 99]

Recent advancement in the literature has provided various ways to simplify both modeling and rigging.

Noticeably, sketch-based 3D modeling has been a popular research field. It provides an easy-to-use interface and enables rapid iterative prototyping.

[click]

For example, the seminal paper on Teddy describes the creation of freeform 3D models from scratch.

[click]

As extensions to such freeform modelers, other researchers have proposed to use variational implicit functions for the underlying shape representations, such as the BlobTree used in ShapeShop.

[click]

Later, FiberMesh demonstrated how to integrate 3D control curves into sketch-based modeling tools. The curves on the surfaces serve as handles for controlling the geometry.

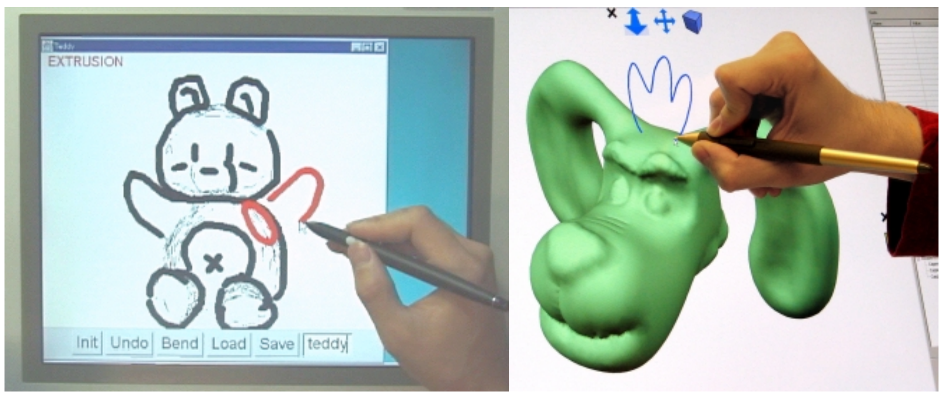
[click]

Because a single sketch usually does not suffice for creating a complex shape, there has also been interesting work on part based modeling, such as layered surface composition using Surface Trees.

[click]

Sketch-based Modeling

- Teddy [Igarashi et al. 99]
- ShapeShop [Schmidt et al. 05]



Recent advancement in the literature has provided various ways to simplify both modeling and rigging.

Noticeably, sketch-based 3D modeling has been a popular research field. It provides an easy-to-use interface and enables rapid iterative prototyping.

[click]

For example, the seminal paper on Teddy describes the creation of freeform 3D models from scratch.

[click]

As extensions to such freeform modelers, other researchers have proposed to use variational implicit functions for the underlying shape representations, such as the BlobTree used in ShapeShop.

[click]

Later, FiberMesh demonstrated how to integrate 3D control curves into sketch-based modeling tools. The curves on the surfaces serve as handles for controlling the geometry.

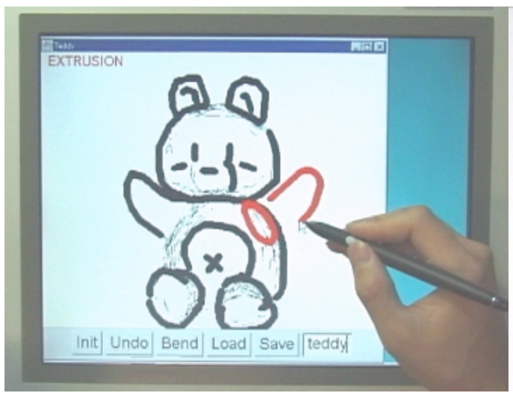
[click]

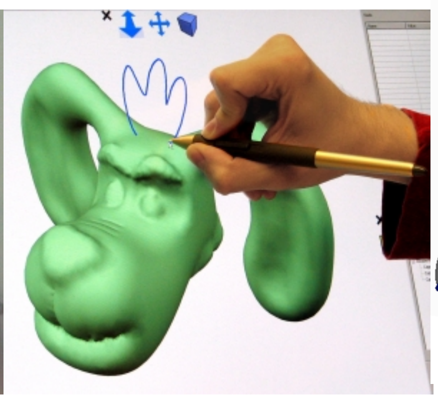
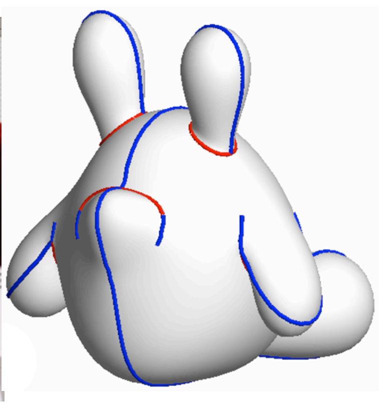
Because a single sketch usually does not suffice for creating a complex shape, there has also been interesting work on part based modeling, such as layered surface composition using Surface Trees.

[click]

Sketch-based Modeling

- Teddy [Igarashi et al. 99]
- ShapeShop [Schmidt et al. 05]
- FiberMesh [Nealen et al. 07]



Recent advancement in the literature has provided various ways to simplify both modeling and rigging.

Noticeably, sketch-based 3D modeling has been a popular research field. It provides an easy-to-use interface and enables rapid iterative prototyping.

[click]

For example, the seminal paper on Teddy describes the creation of freeform 3D models from scratch.

[click]

As extensions to such freeform modelers, other researchers have proposed to use variational implicit functions for the underlying shape representations, such as the BlobTree used in ShapeShop.

[click]

Later, FiberMesh demonstrated how to integrate 3D control curves into sketch-based modeling tools. The curves on the surfaces serve as handles for controlling the geometry.

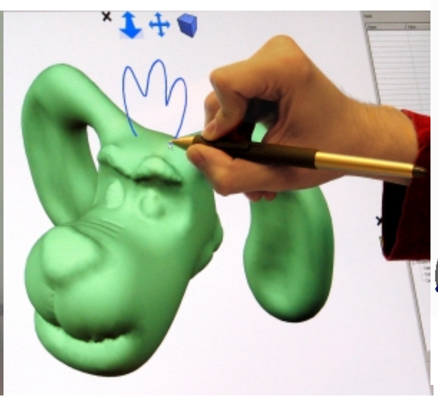
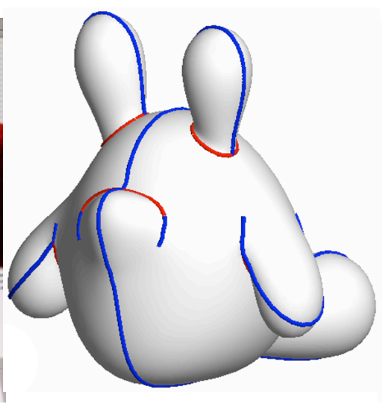
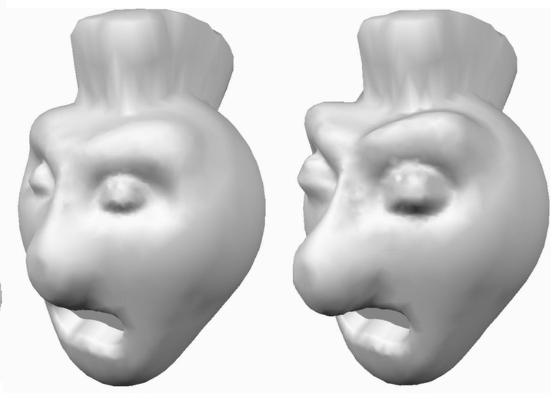
[click]

Because a single sketch usually does not suffice for creating a complex shape, there has also been interesting work on part based modeling, such as layered surface composition using Surface Trees.

[click]

Sketch-based Modeling

- Teddy [Igarashi et al. 99]
- ShapeShop [Schmidt et al. 05]
- FiberMesh [Nealen et al. 07]
- SurfaceTrees [Schmidt and Singh 08]

Recent advancement in the literature has provided various ways to simplify both modeling and rigging.

Noticeably, sketch-based 3D modeling has been a popular research field. It provides an easy-to-use interface and enables rapid iterative prototyping.

[click]

For example, the seminal paper on Teddy describes the creation of freeform 3D models from scratch.

[click]

As extensions to such freeform modelers, other researchers have proposed to use variational implicit functions for the underlying shape representations, such as the BlobTree used in ShapeShop.

[click]

Later, FiberMesh demonstrated how to integrate 3D control curves into sketch-based modeling tools. The curves on the surfaces serve as handles for controlling the geometry.

[click]

Because a single sketch usually does not suffice for creating a complex shape, there has also been interesting work on part based modeling, such as layered surface composition using Surface Trees.

[click]

Automatic Rigging

While sketch-based tools simplify the process of modeling, there also exists work in the literature that inspires automatic rigging on finished models.

[click]

Skeleton extraction is a well-studied problem in different communities such as vision, psychology and graphics.

[click]

For example, here's a recent survey on a lot of work from generating 2D medial-axes, 3D medial surfaces to 3D curve-skeletons.

[click]

On the other hand, if the skeleton topology is known, 3D models can be rigged using skeleton embedding and automatic skinning. For example, the Pinocchio system proposed an automatic skinning method using heat diffusion.

[click]

Automatic Rigging

- Skeleton extraction
  [Sharf et al. 07] [Pan et al. 09] [Ju et al. 11]

While sketch-based tools simplify the process of modeling, there also exists work in the literature that inspires automatic rigging on finished models.

[click]

Skeleton extraction is a well-studied problem in different communities such as vision, psychology and graphics.

[click]

For example, here's a recent survey on a lot of work from generating 2D medial-axes, 3D medial surfaces to 3D curve-skeletons.

[click]

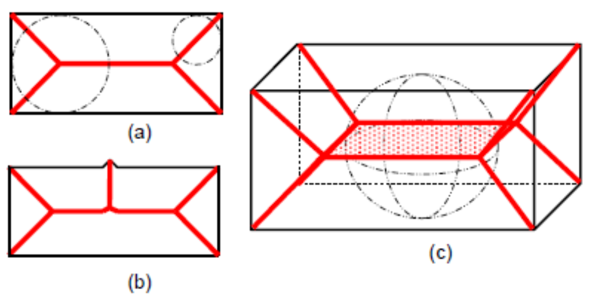
On the other hand, if the skeleton topology is known, 3D models can be rigged using skeleton embedding and automatic skinning. For example, the Pinocchio system proposed an automatic skinning method using heat diffusion.

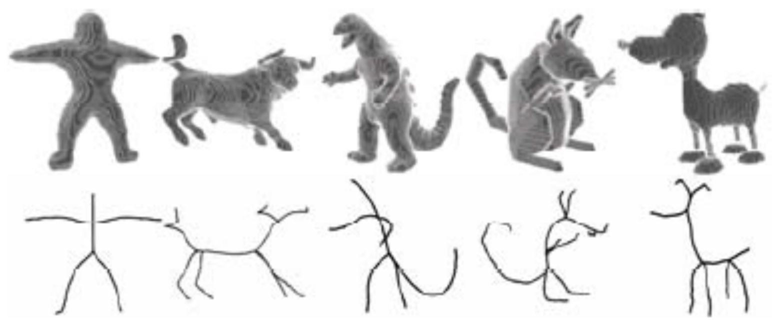
[click]

Automatic Rigging

- Skeleton extraction

[Sharf et al. 07] [Pan et al. 09] [Ju et al. 11]





[Cornea et al. 07]

Sponsored by ACM SIGGRAPH

While sketch-based tools simplify the process of modeling, there also exists work in the literature that inspires automatic rigging on finished models.

[click]

Skeleton extraction is a well-studied problem in different communities such as vision, psychology and graphics.

[click]

For example, here's a recent survey on a lot of work from generating 2D medial-axes, 3D medial surfaces to 3D curve-skeletons.

[click]

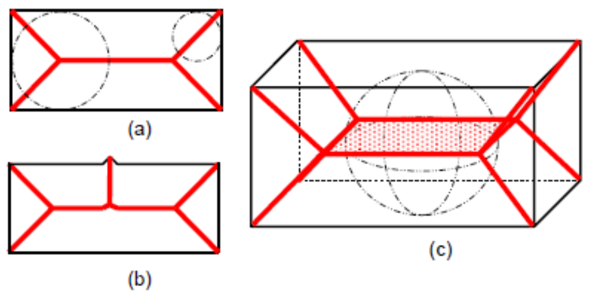
On the other hand, if the skeleton topology is known, 3D models can be rigged using skeleton embedding and automatic skinning. For example, the Pinocchio system proposed an automatic skinning method using heat diffusion.

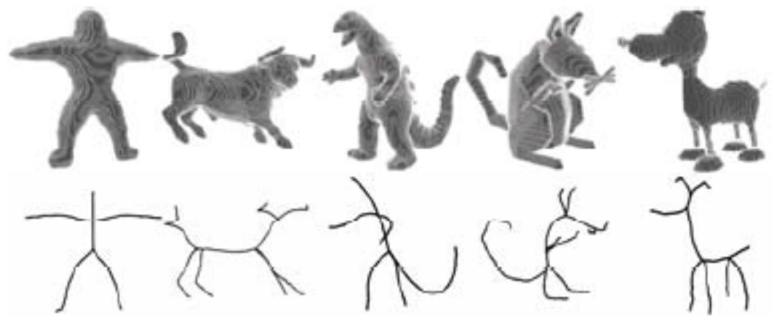
[click]

Automatic Rigging

- Skeleton extraction

  [Sharf et al. 07] [Pan et al. 09] [Ju et al. 11]

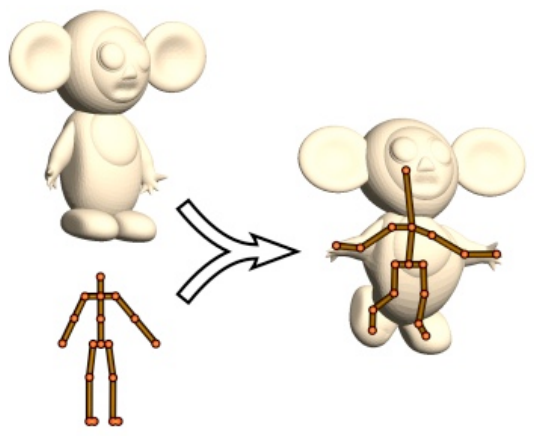




[Cornea et al. 07]

- Automatic skinning

  Pinocchio [Baran and Popović 07]



While sketch-based tools simplify the process of modeling, there also exists work in the literature that inspires automatic rigging on finished models.

[click]

Skeleton extraction is a well-studied problem in different communities such as vision, psychology and graphics.

[click]

For example, here's a recent survey on a lot of work from generating 2D medial-axes, 3D medial surfaces to 3D curve-skeletons.

[click]

On the other hand, if the skeleton topology is known, 3D models can be rigged using skeleton embedding and automatic skinning. For example, the Pinocchio system proposed an automatic skinning method using heat diffusion.

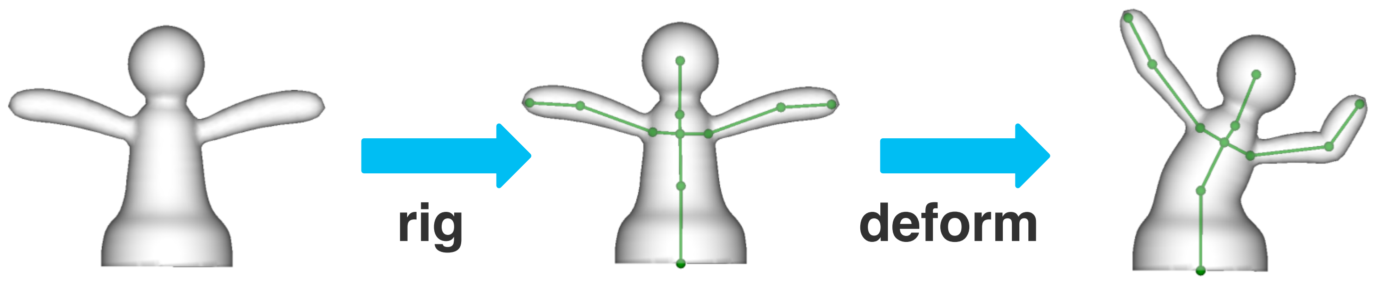
[click]

Solution?

Sponsored by ACM SIGGRAPH

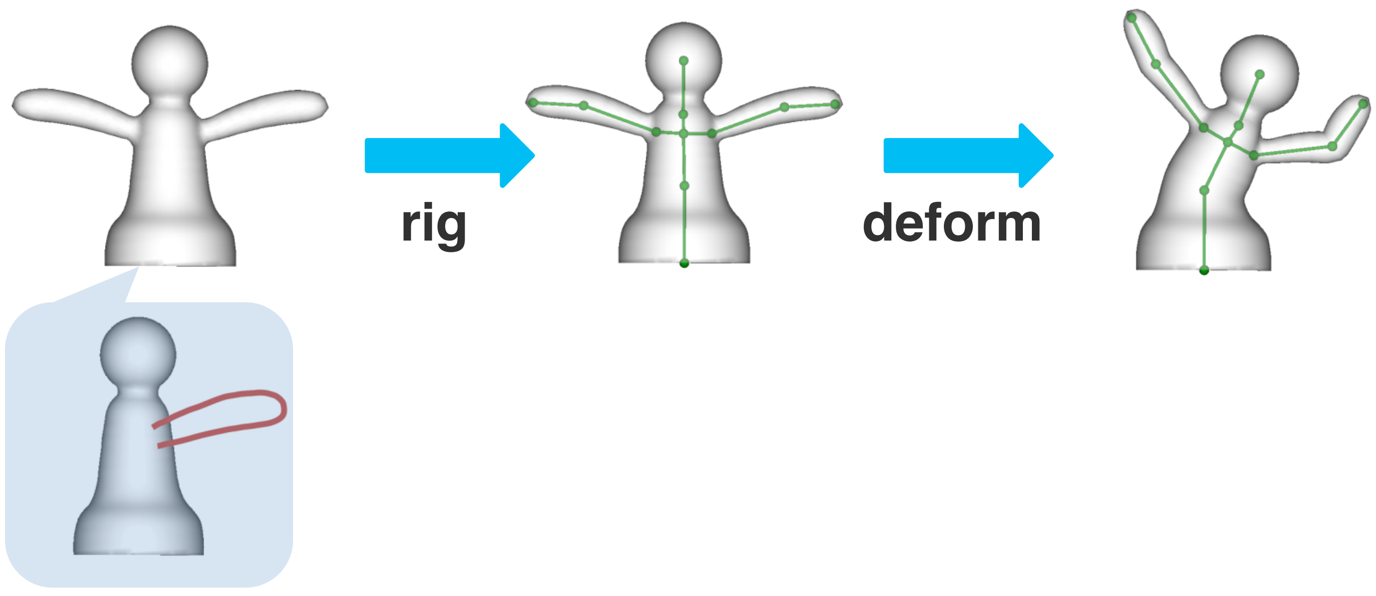
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution?



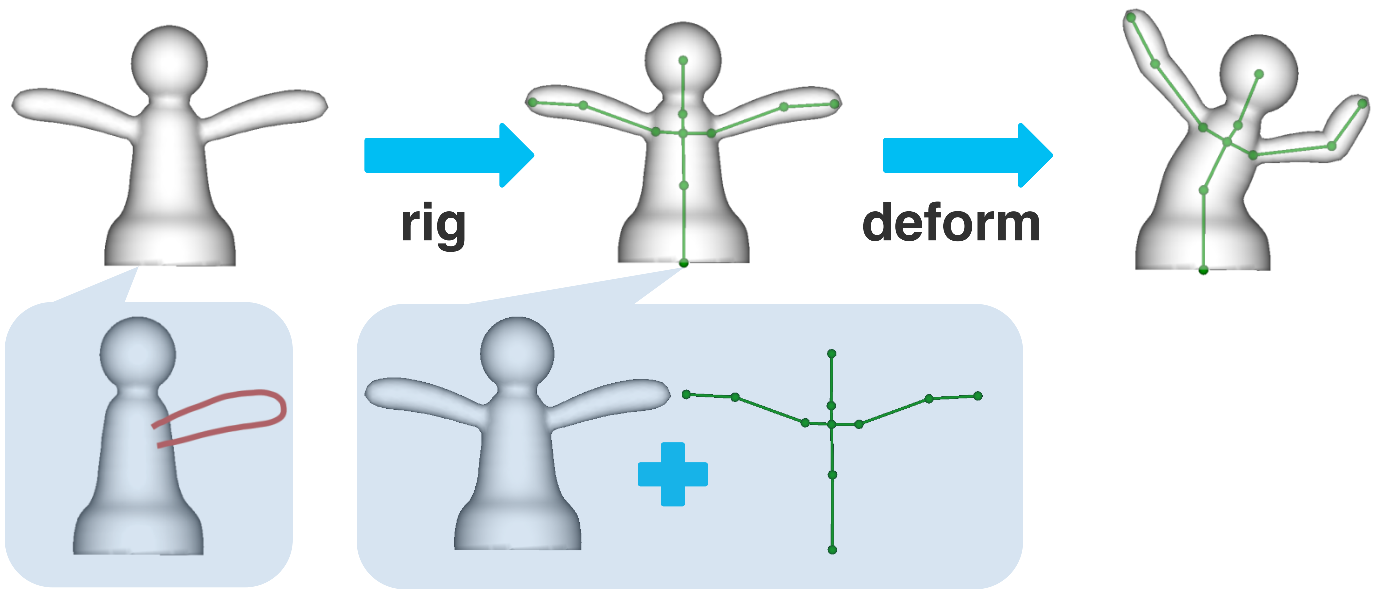
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution?



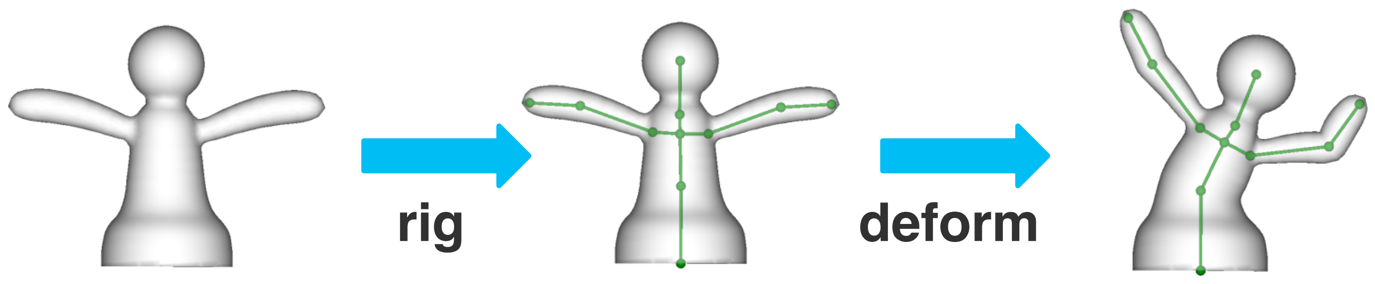
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution?



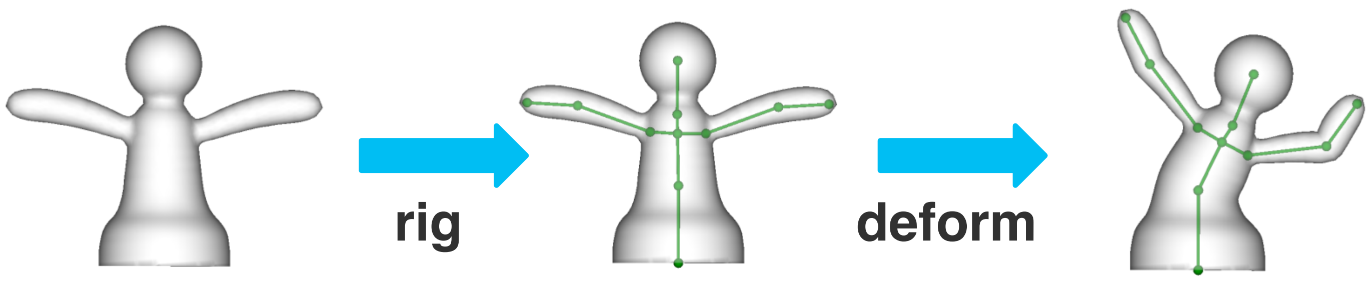
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution?



Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution



Models rigged at all times

Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?

[click]

For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part

[click]

and use automatic rigging such as Pinocchio for the rigging part?

[click]

Unfortunately, this still does not change the linearity of the pipeline.

Also the automatic rigging methods are typically not fast enough for real-time interactions.

If the animators want to rapidly iterate on their designs, they shouldn’t have to go through the entire pipeline

[click]

Here is the solution we propose:

In order to break the sequential pipeline, the model should be rigged at all times.

[click]

As a result of that:

The animator can perform non-linear editing in terms of modeling, rigging and deformation

[click]

She can also model by parts;[short pause]

Here is what the whole modeling process should look like.

[click]

After shape creation the model is already rigged

[click]

The animator can freely pose and change the model

[click]

For example, if she’s not satisfied with the hand, she can remove it use a cutting operation

[click]

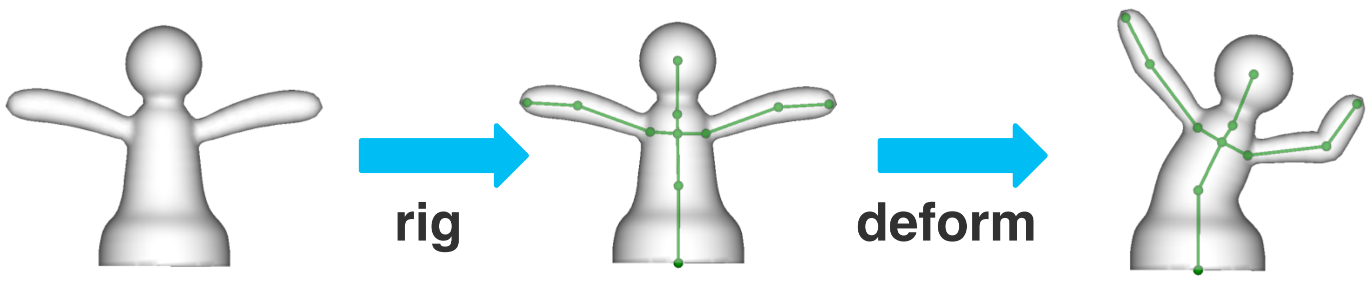
And replace it with a new part by using the merging operation

[click]

All of these operations should happen in real time

[click]

Solution

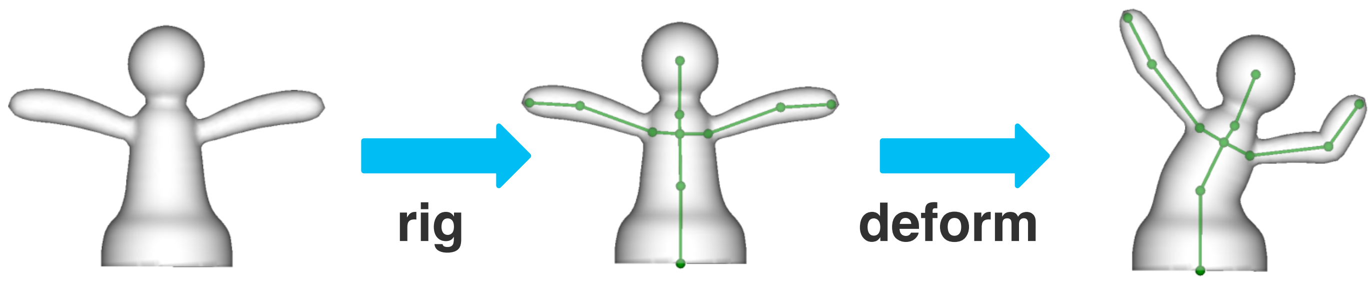


Models rigged at all times

- **Non-linear editing**

Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution

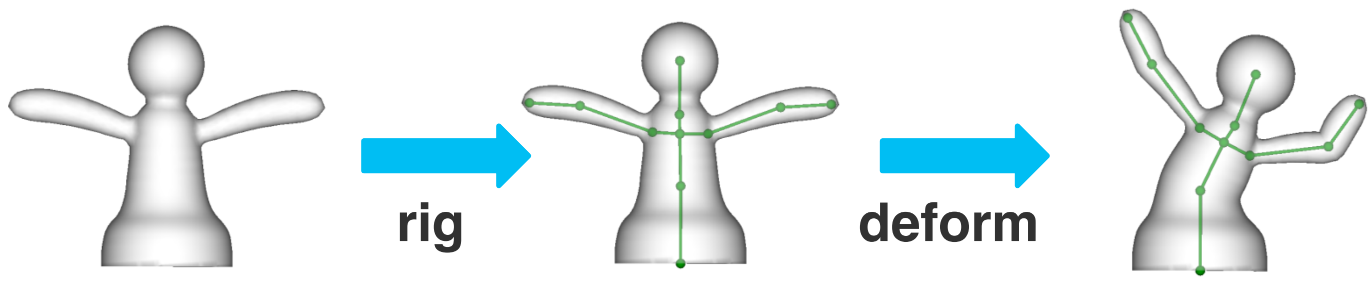


Models rigged at all times

- **Non-linear editing**
- **Modeling by parts**

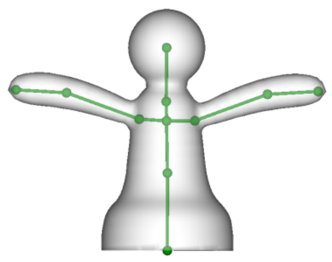
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution



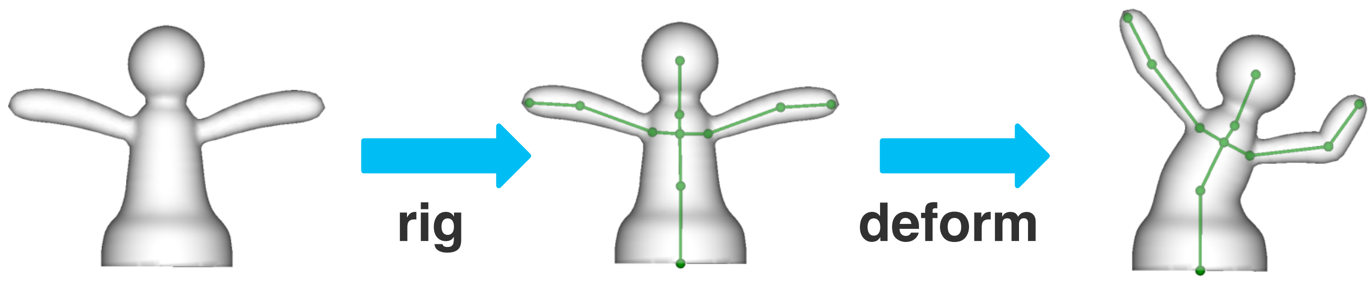
Models rigged at all times

- **Non-linear editing**
- **Modeling by parts**



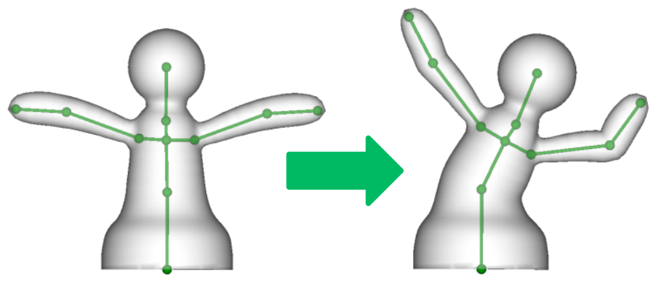
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution



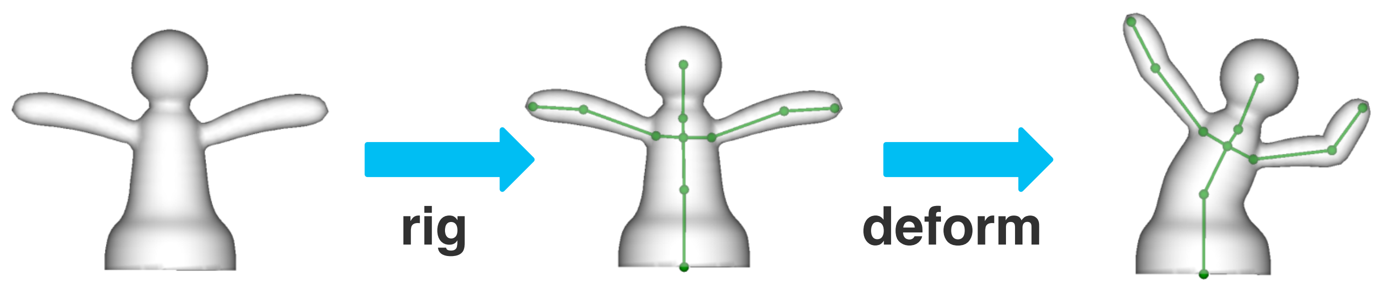
Models rigged at all times

- **Non-linear editing**
- **Modeling by parts**



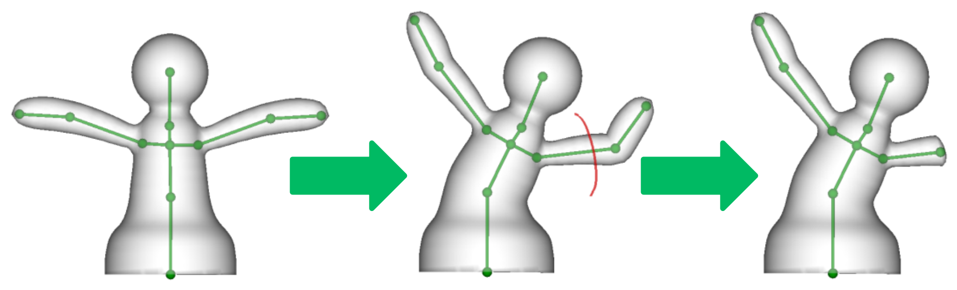
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn’t have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she’s not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution



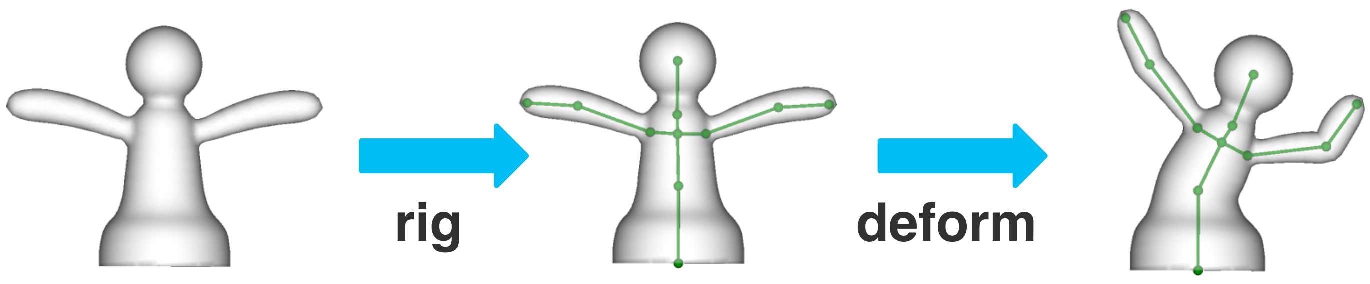
Models rigged at all times

- **Non-linear editing**
- **Modeling by parts**



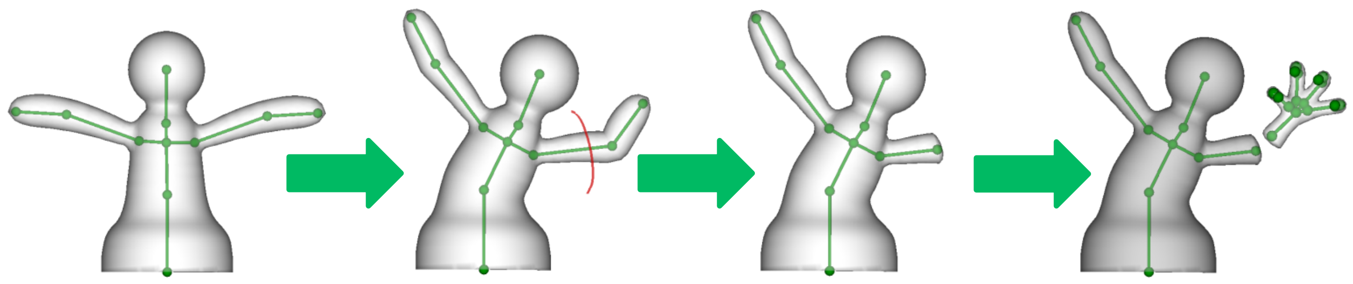
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution



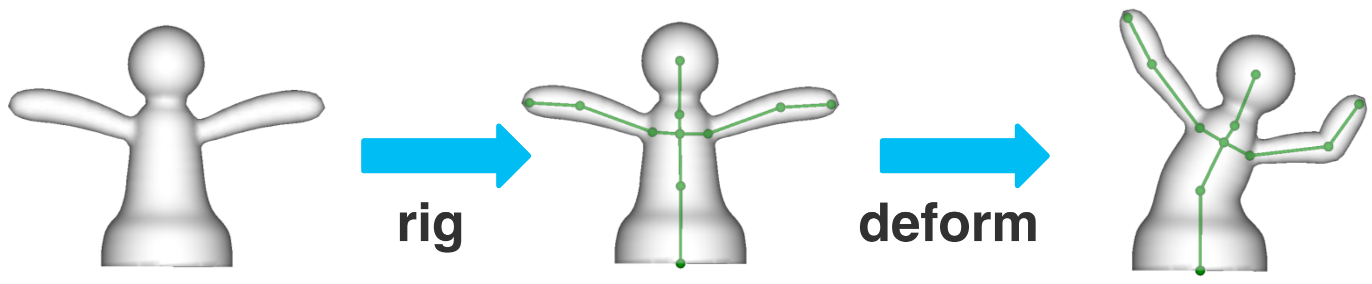
Models rigged at all times

- **Non-linear editing**
- **Modeling by parts**



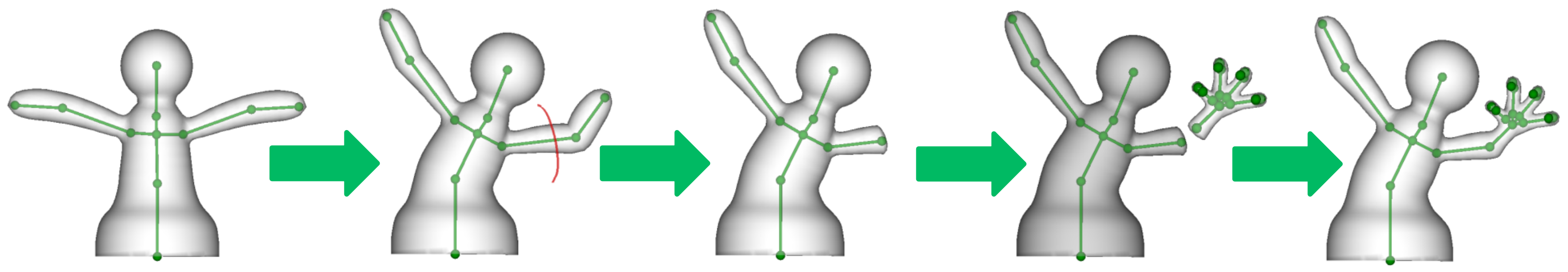
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

Solution



Models rigged at all times

- **Non-linear editing**
- **Modeling by parts**



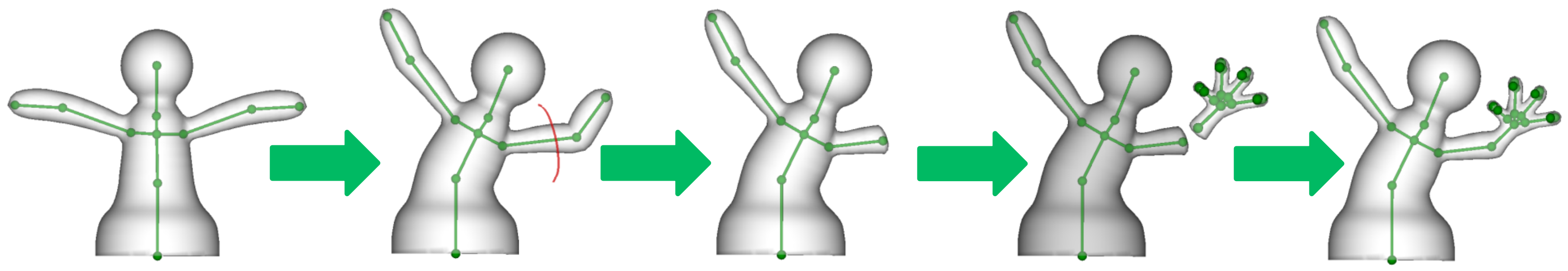
Now, if we take a look again at the previous pipeline, can we use these existing methods to solve the problem?
[click]
For example, we can use sketch-based modeling tools such as FiberMesh for the modeling part
[click]
and use automatic rigging such as Pinocchio for the rigging part?
[click]
Unfortunately, this still does not change the linearity of the pipeline.
Also the automatic rigging methods are typically not fast enough for real-time interactions.
If the animators want to rapidly iterate on their designs, they shouldn't have to go through the entire pipeline
[click]
Here is the solution we propose:
In order to break the sequential pipeline, the model should be rigged at all times.
[click]
As a result of that:
The animator can perform non-linear editing in terms of modeling, rigging and deformation
[click]
She can also model by parts;[short pause]
Here is what the whole modeling process should look like.
[click]
After shape creation the model is already rigged
[click]
The animator can freely pose and change the model
[click]
For example, if she's not satisfied with the hand, she can remove it use a cutting operation
[click]
And replace it with a new part by using the merging operation
[click]
All of these operations should happen in real time
[click]

RigMesh

Contributions

Models rigged at all times

- Non-linear editing
- Modeling by parts



Sponsored by ACM SIGGRAPH

Here are our contributions in light of the proposed solution:

[click]

In addition to enabling non-linear editing, the animator can model by parts using three basic operations of sketching, cutting and merging;

[click]

We propose a method for automatically determining the bone skeleton from the user sketch; specifically determining the number and placement of the joints, which is the key issue in our system for controls and degrees of freedom;

[click]

Because the shape can be locally changed using operations like cutting and merging, we need to come up with a method for updating the skin weights in real-time, without globally re-skinning the whole shape.

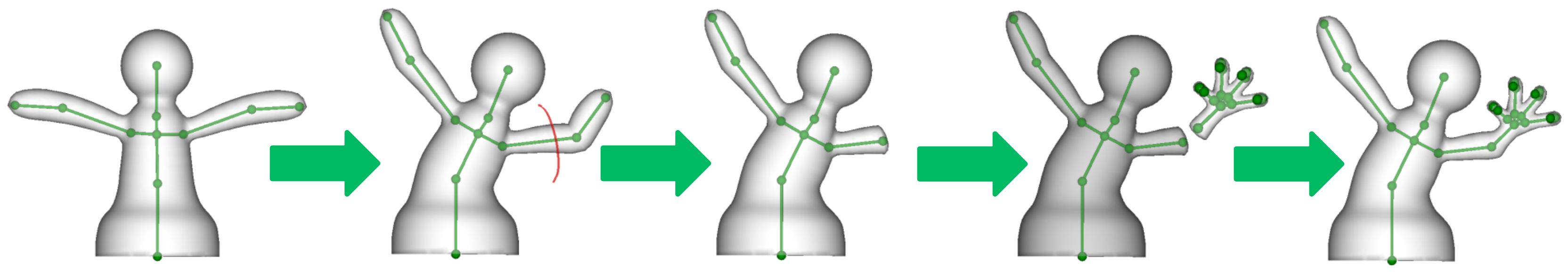
[click]

RigMesh

Contributions

Models rigged at all times

- Non-linear editing
- Modeling by parts: Sketch, Cut & Merge



Sponsored by ACM SIGGRAPH

Here are our contributions in light of the proposed solution:

[click]

In addition to enabling non-linear editing, the animator can model by parts using three basic operations of sketching, cutting and merging;

[click]

We propose a method for automatically determining the bone skeleton from the user sketch; specifically determining the number and placement of the joints, which is the key issue in our system for controls and degrees of freedom;

[click]

Because the shape can be locally changed using operations like cutting and merging, we need to come up with a method for updating the skin weights in real-time, without globally re-skinning the whole shape.

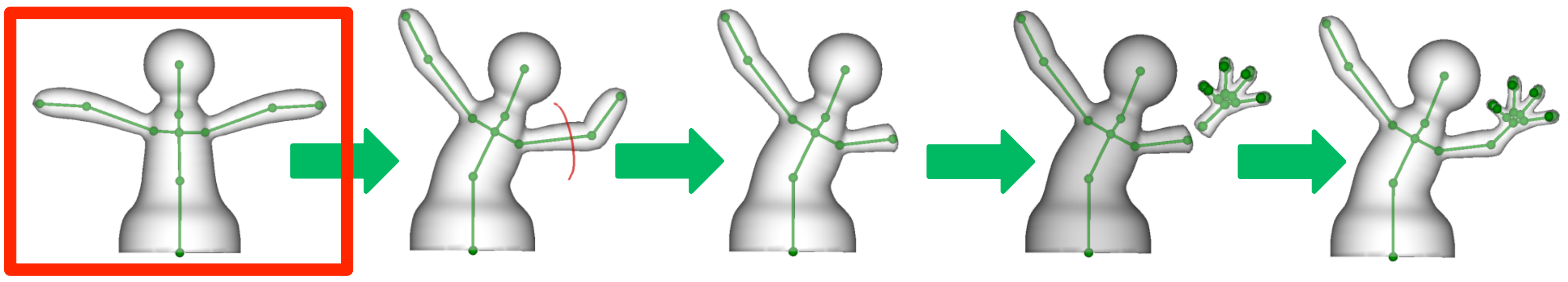
[click]

RigMesh

Contributions

Models rigged at all times

- Non-linear editing
- Modeling by parts: Sketch, Cut & Merge
- Automatically determining skeletal structure from sketch



Sponsored by ACM SIGGRAPH

Here are our contributions in light of the proposed solution:

[click]

In addition to enabling non-linear editing, the animator can model by parts using three basic operations of sketching, cutting and merging;

[click]

We propose a method for automatically determining the bone skeleton from the user sketch; specifically determining the number and placement of the joints, which is the key issue in our system for controls and degrees of freedom;

[click]

Because the shape can be locally changed using operations like cutting and merging, we need to come up with a method for updating the skin weights in real-time, without globally re-skinning the whole shape.

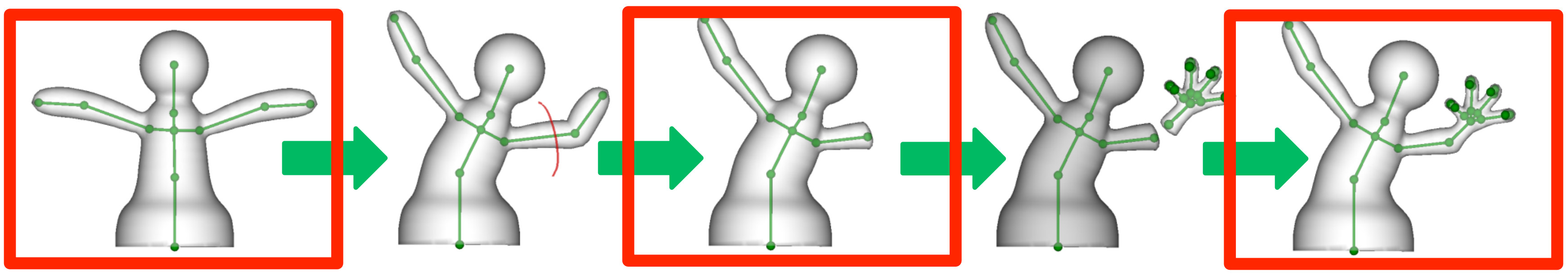
[click]

RigMesh

Contributions

Models rigged at all times

- Non-linear editing
- Modeling by parts: Sketch, Cut & Merge
- Automatically determining skeletal structure from sketch
- Efficient local skin weights computation



Sponsored by ACM SIGGRAPH

Here are our contributions in light of the proposed solution:

[click]

In addition to enabling non-linear editing, the animator can model by parts using three basic operations of sketching, cutting and merging;

[click]

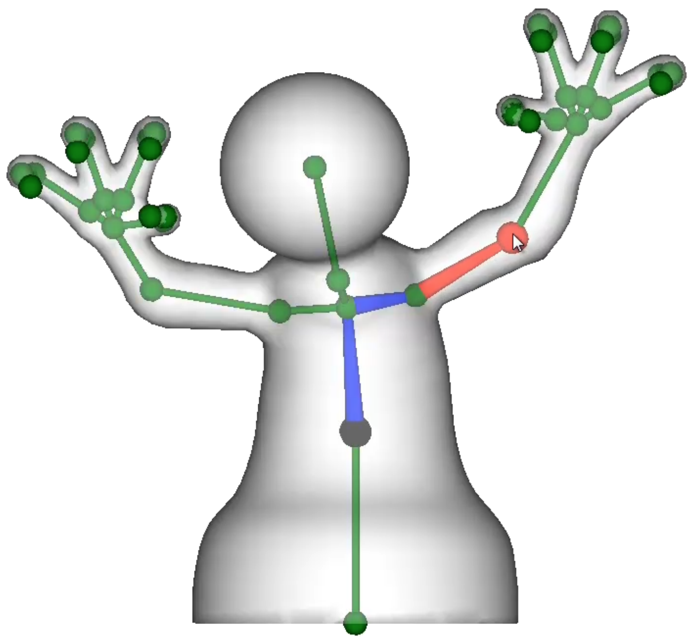
We propose a method for automatically determining the bone skeleton from the user sketch; specifically determining the number and placement of the joints, which is the key issue in our system for controls and degrees of freedom;

[click]

Because the shape can be locally changed using operations like cutting and merging, we need to come up with a method for updating the skin weights in real-time, without globally re-skinning the whole shape.

[click]

Demo



Sponsored by ACM SIGGRAPH

Now I'm going to show you an interactive demo:

So this is the RigMesh system. You can create a 3D model by sketching its silhouette, like in Teddy or FiberMesh.

The big difference is that the shape is already rigged. You can deform it using IK with our peeling interface. This is a simple operation in our system because the shapes are rigged at all times.

Now let's say you already have some existing models, for example a chess piece. You can try out your different ideas by sketching, cutting and merging.

Let's sketch some arms for this chess piece… and make a copy of it.

If you want to merge the arm with the body, you can either merge by snapping existing joints, or by adding a new joint and snapping there; you can also merge by inserting a new bone.

You can also change the existing skeleton topology by inserting new joints.

Let's attach the arm this way… and pose the shape.

Now during posing, you might notice that the shape would look better if we made some changes to it. For example, you might want a claw instead a puffy hand.

Let's cut it out, and replace it with something else. I already have a claw so let's load it in… and merge it.

Do the same thing for the other arm.

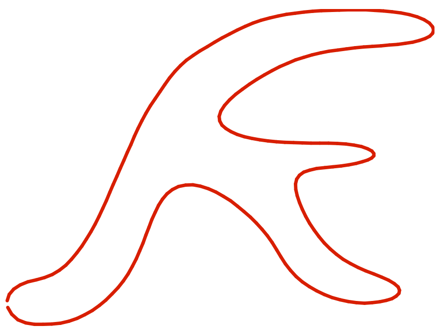
And pose the shape a little bit more.

So you see our tool is especially useful for rapid prototyping, trying out different ideas when you don't even have a specific design goal in mind.

[after demo]

Thanks a lot for your attention. I will now hand it over to Peter.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let’s see what’s going on behind the scenes.

As you’ve seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user’s sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

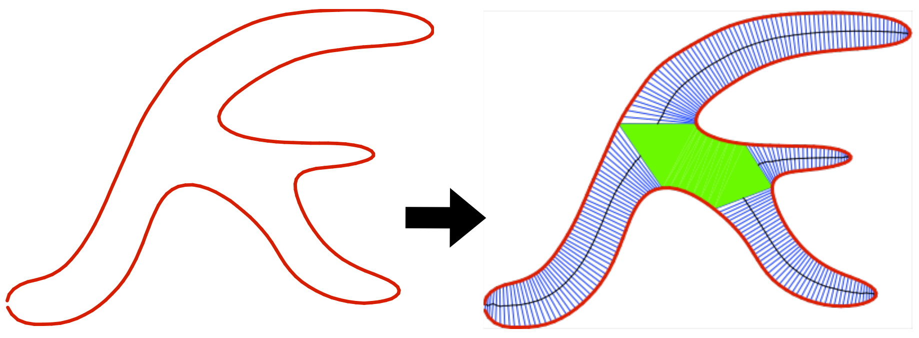
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let’s see what’s going on behind the scenes.

As you’ve seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user’s sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

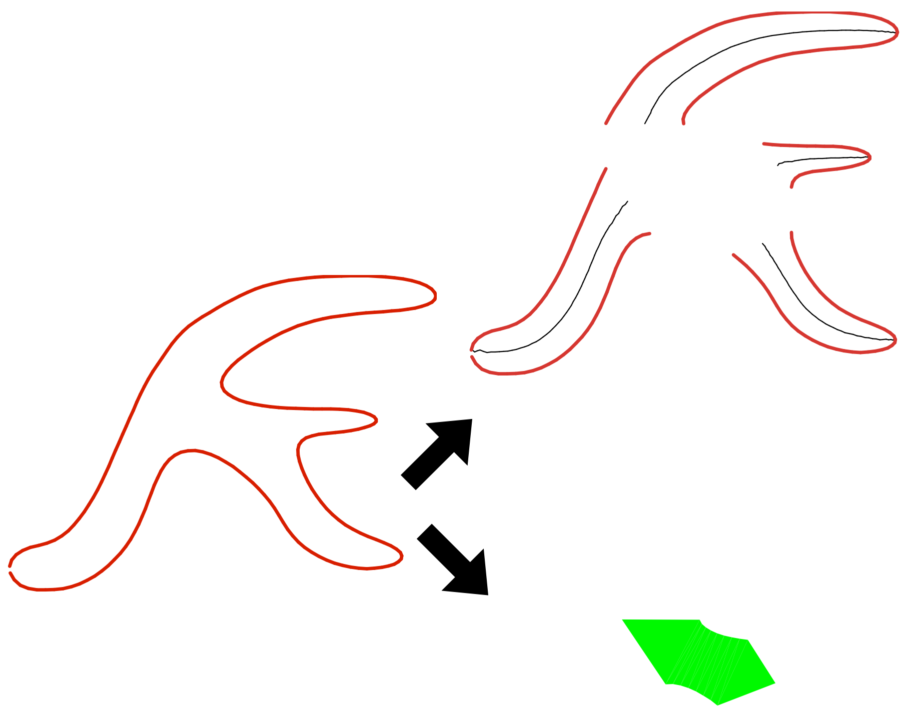
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let's see what's going on behind the scenes.

As you've seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user's sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

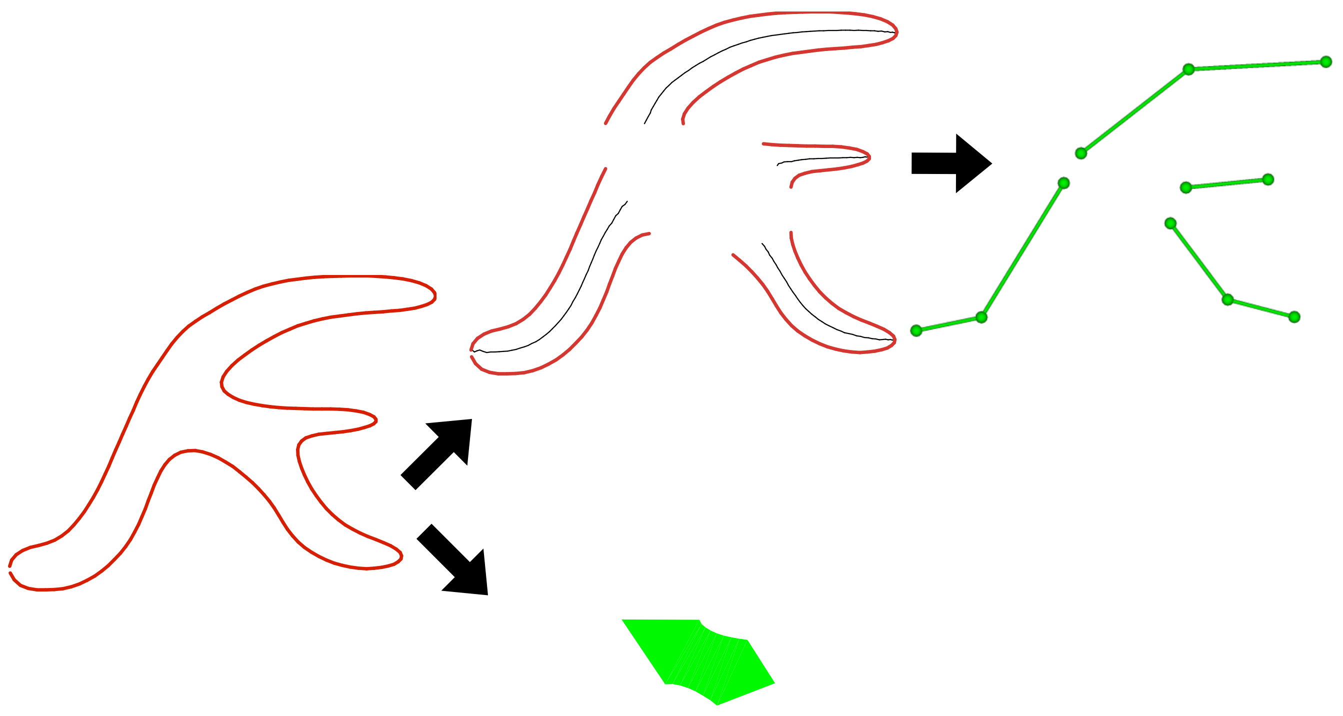
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Thank you. So let's see what's going on behind the scenes.

As you've seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user's sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

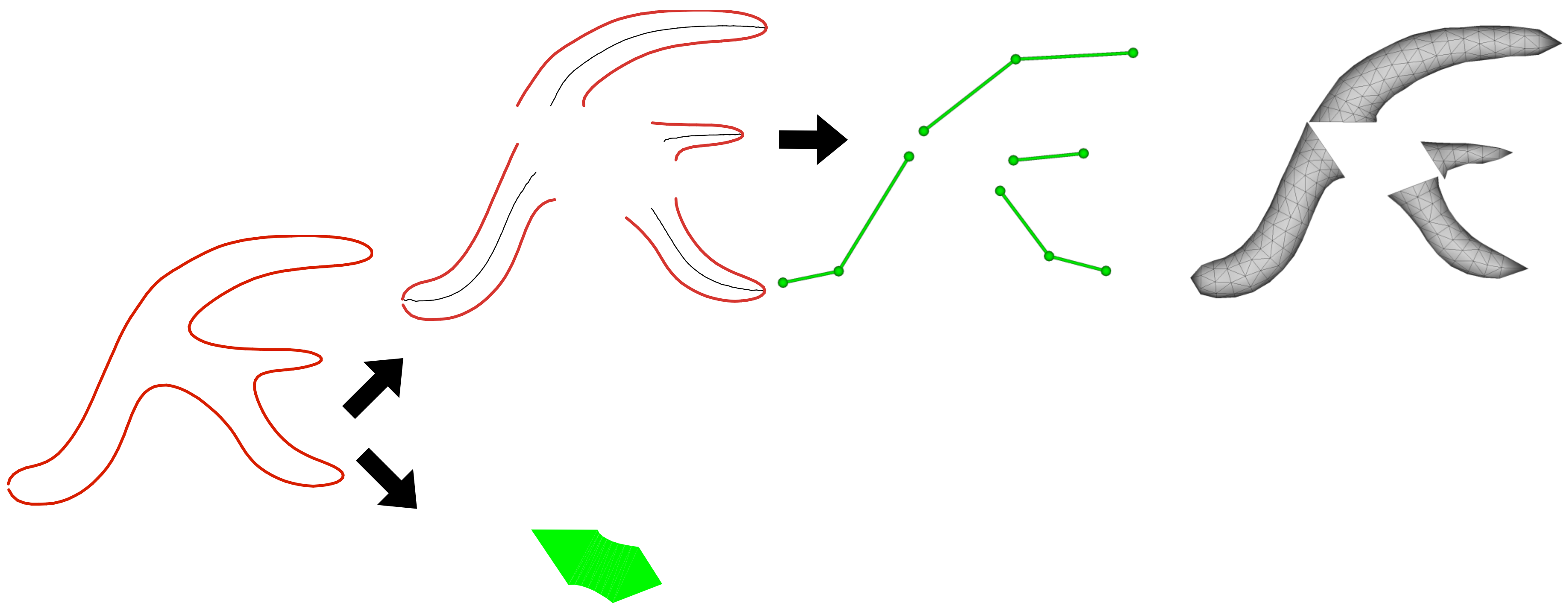
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Thank you. So let’s see what’s going on behind the scenes.

As you’ve seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user’s sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

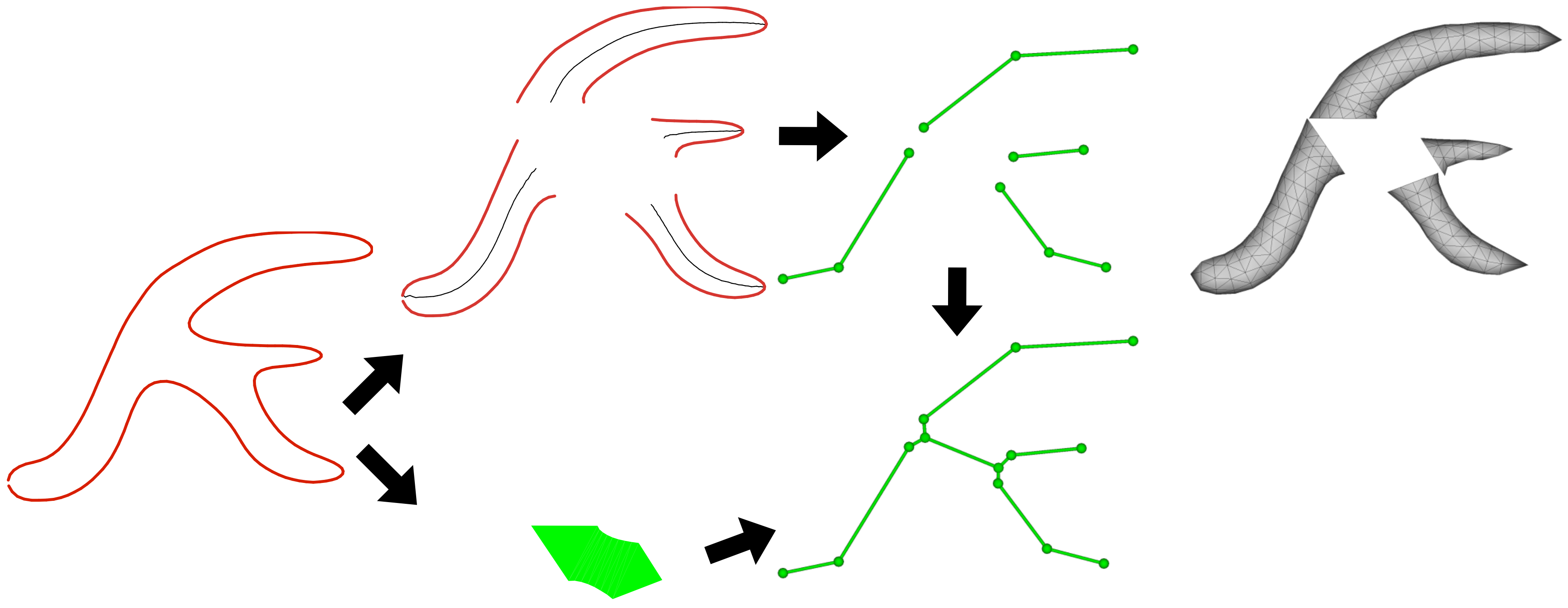
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let’s see what’s going on behind the scenes.

As you’ve seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user’s sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

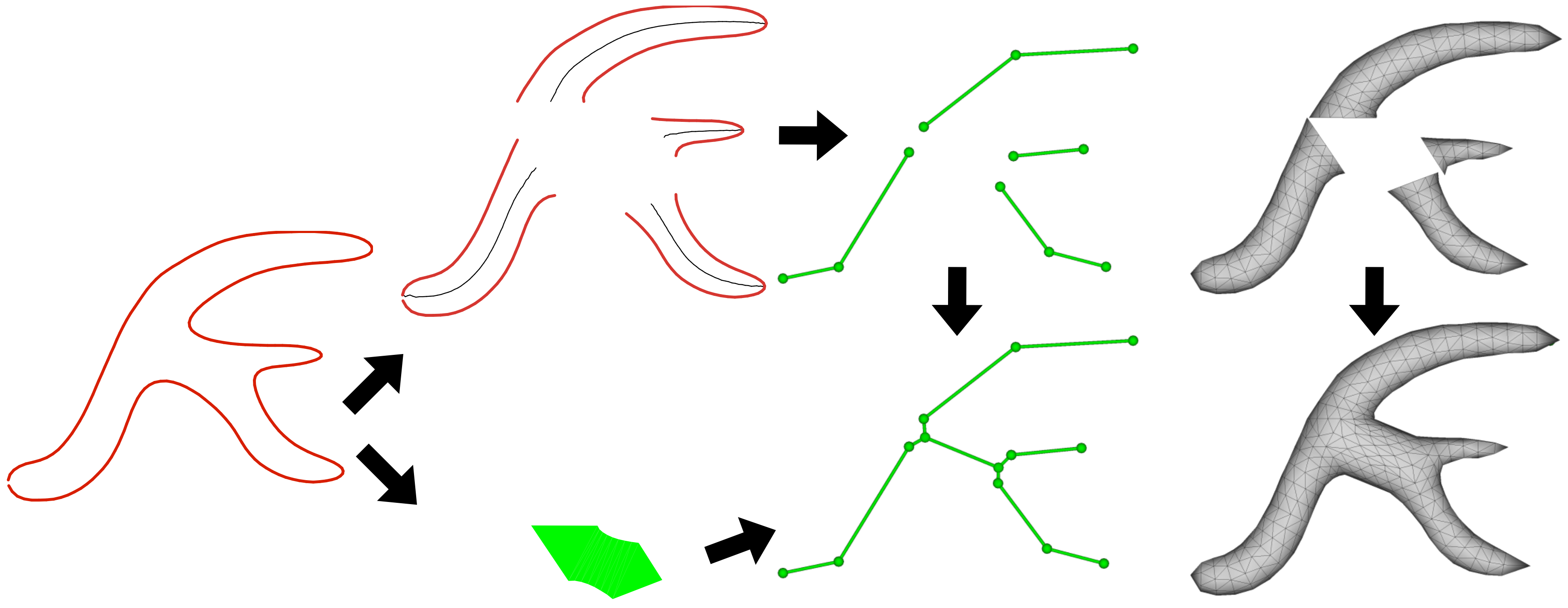
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let’s see what’s going on behind the scenes.

As you’ve seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user’s sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

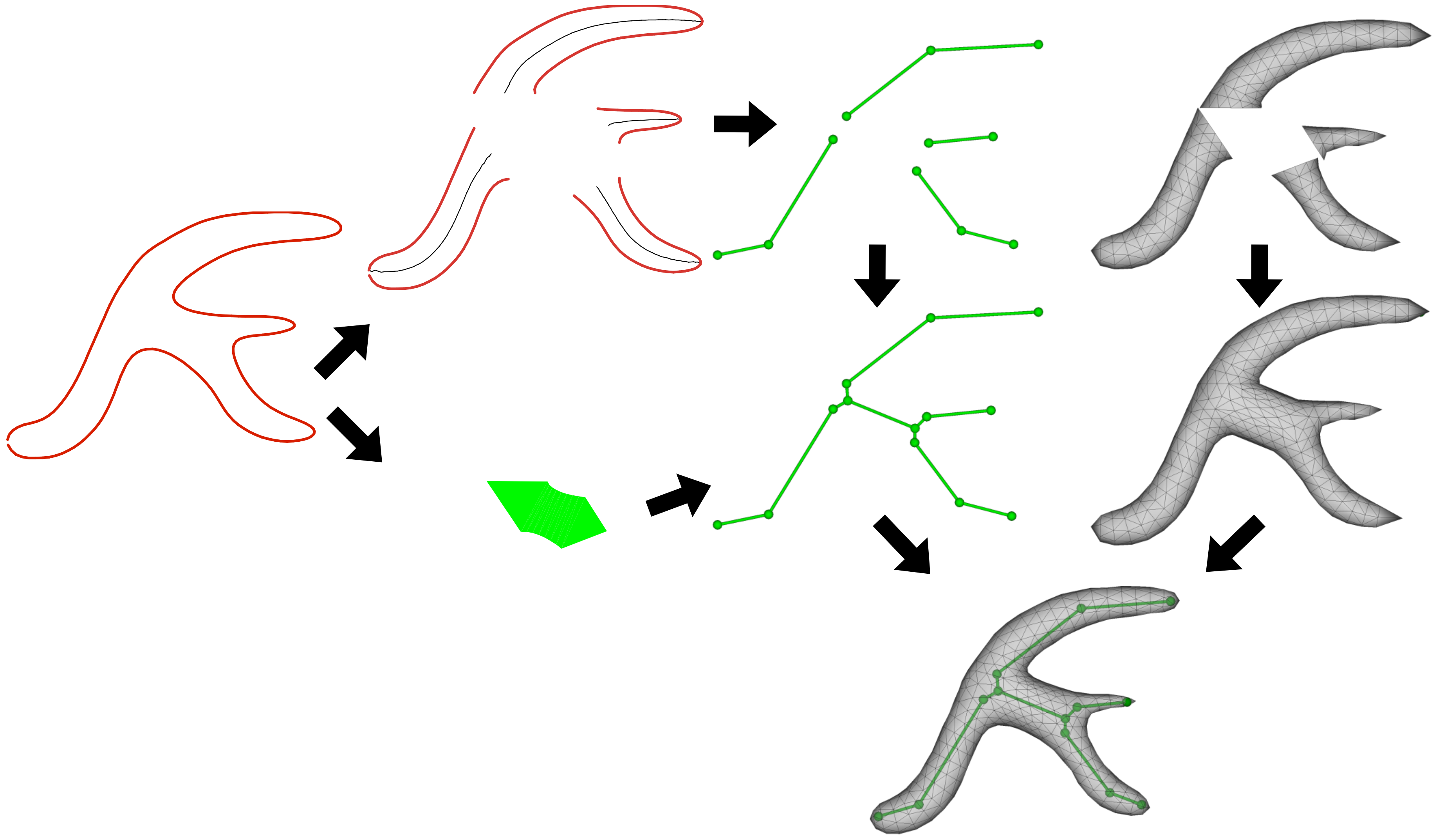
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let's see what's going on behind the scenes.

As you've seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user's sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

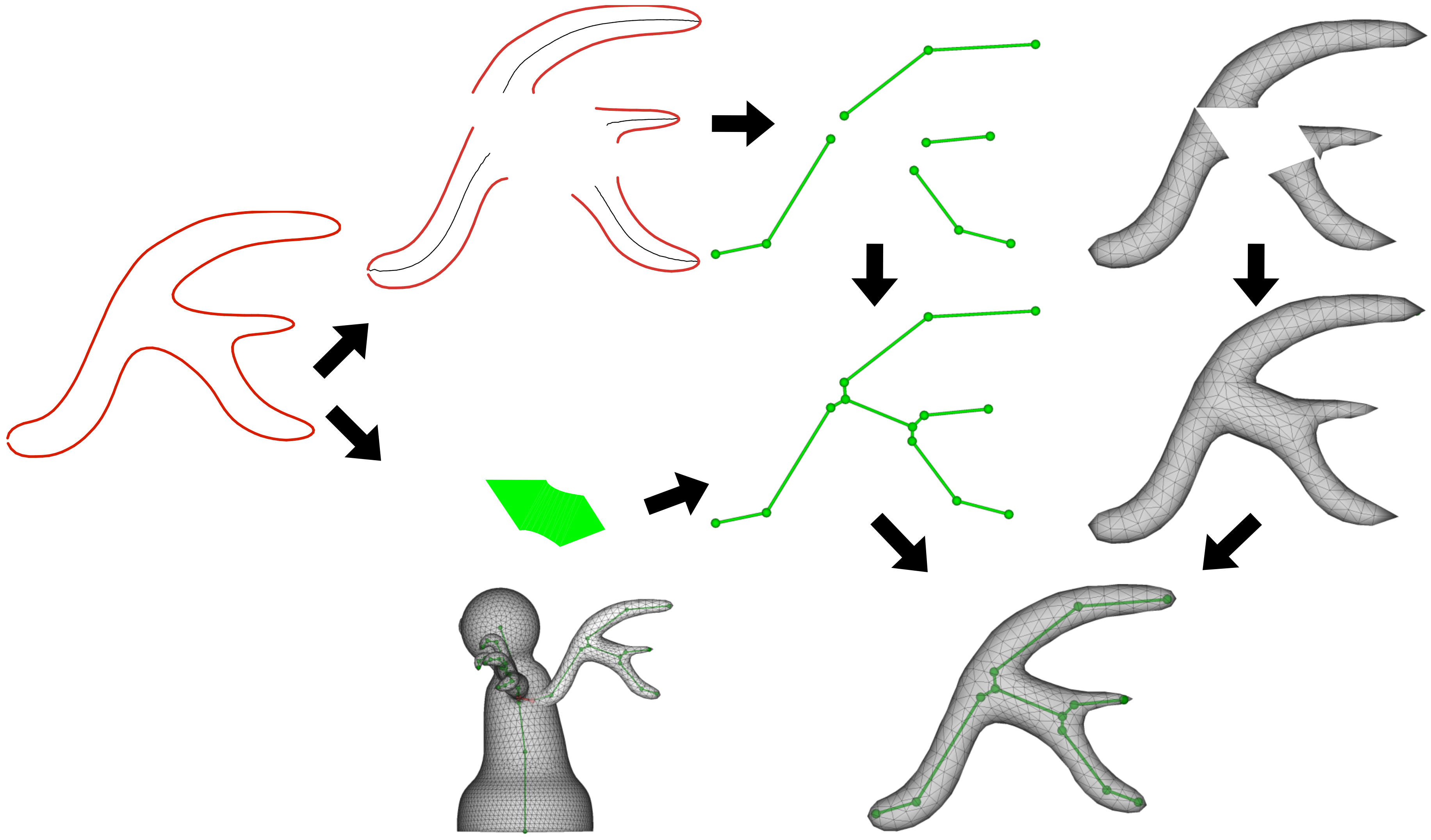
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Thank you. So let's see what's going on behind the scenes.

As you've seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user's sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

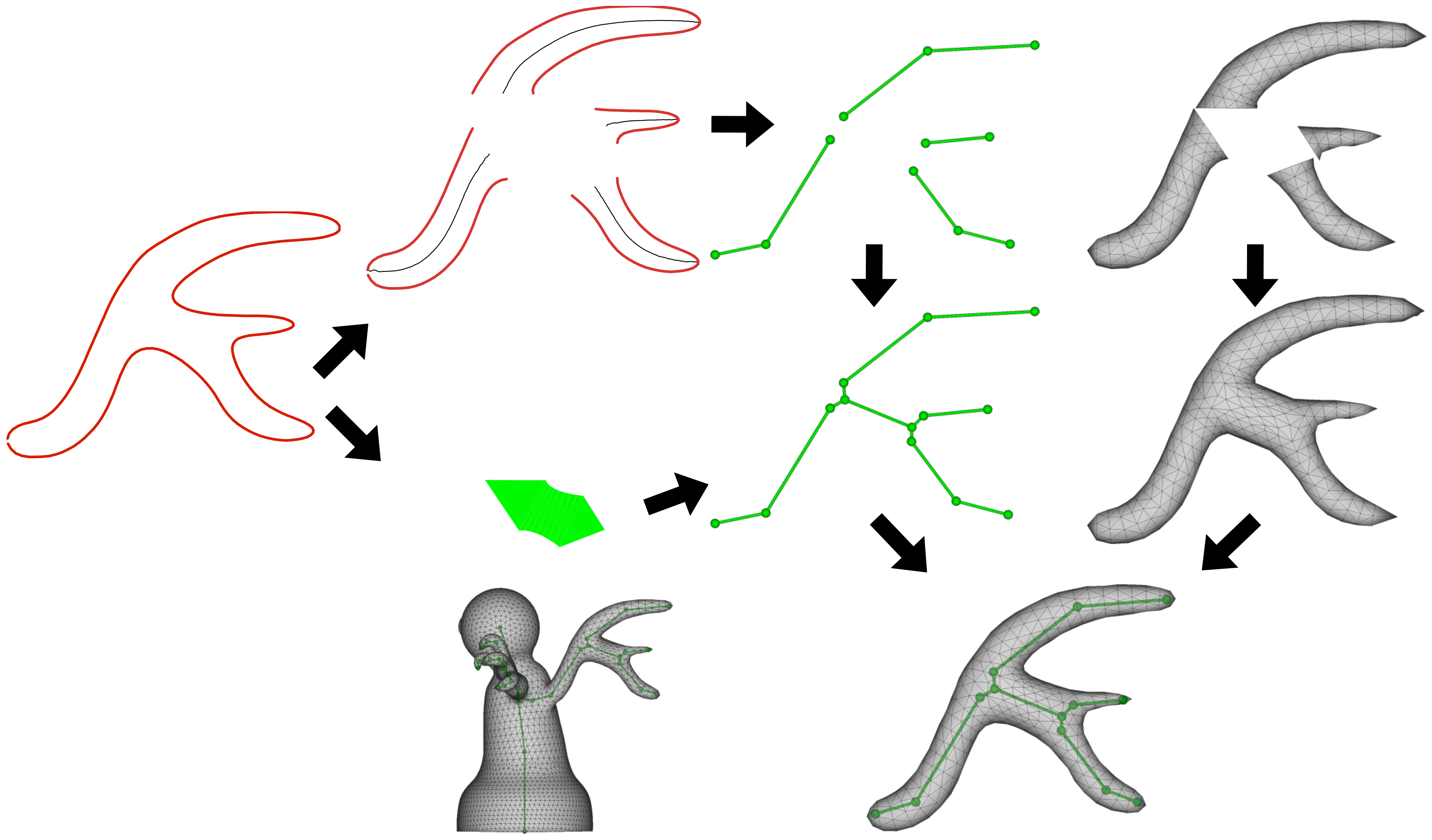
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let’s see what’s going on behind the scenes.

As you’ve seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user’s sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

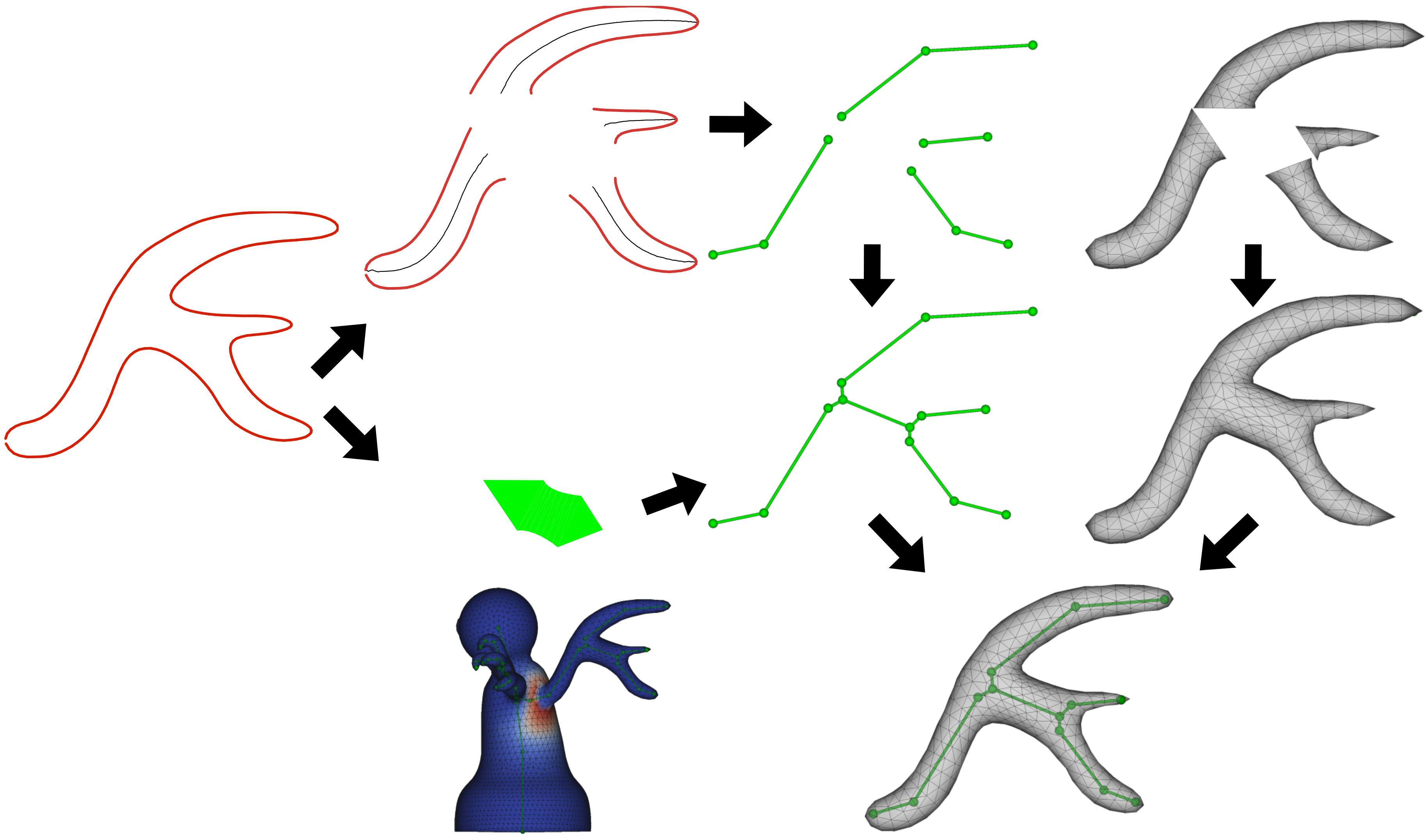
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let’s see what’s going on behind the scenes.

As you’ve seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user’s sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

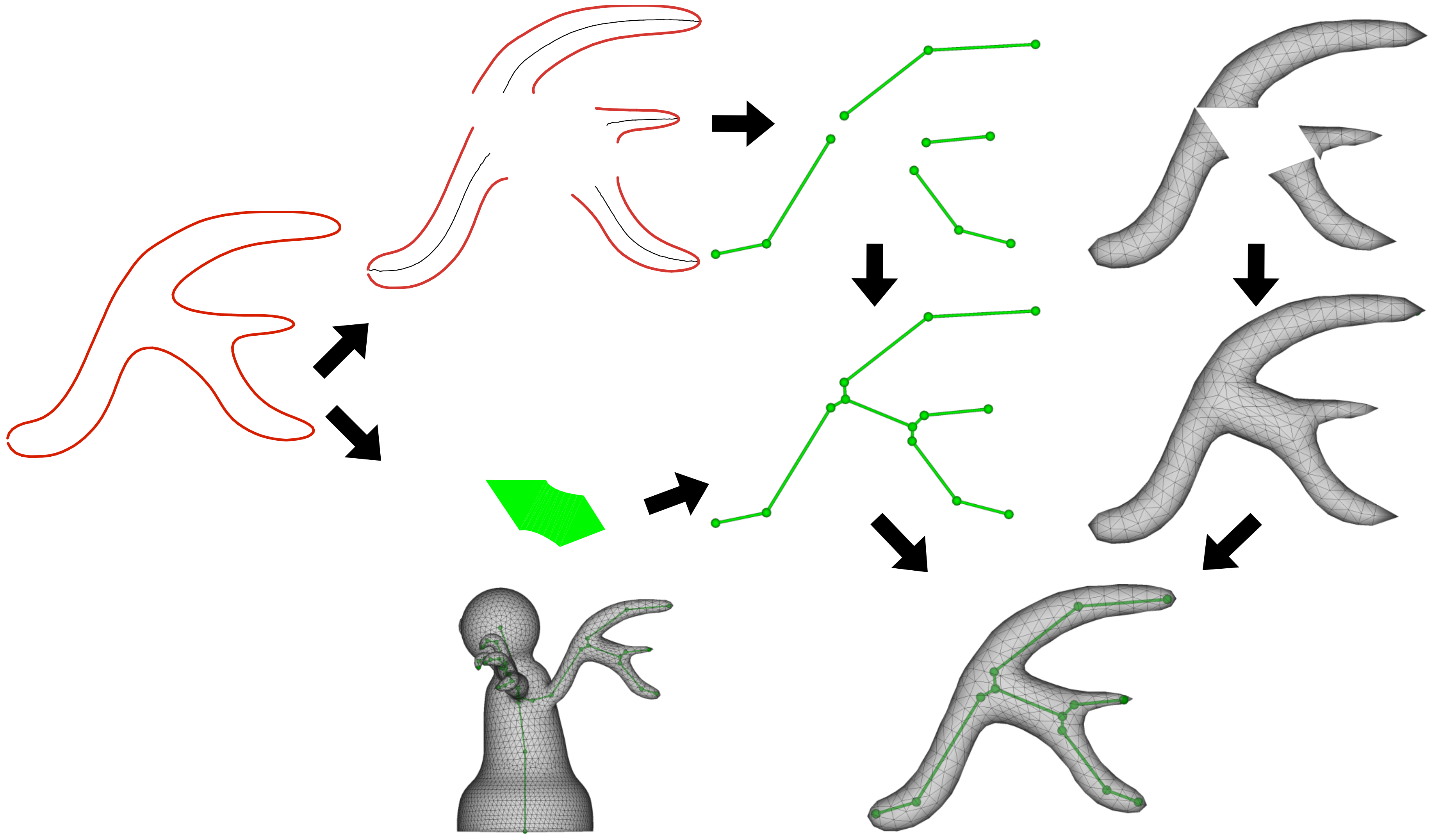
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Sponsored by ACM SIGGRAPH

Thank you. So let’s see what’s going on behind the scenes.

As you’ve seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user’s sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

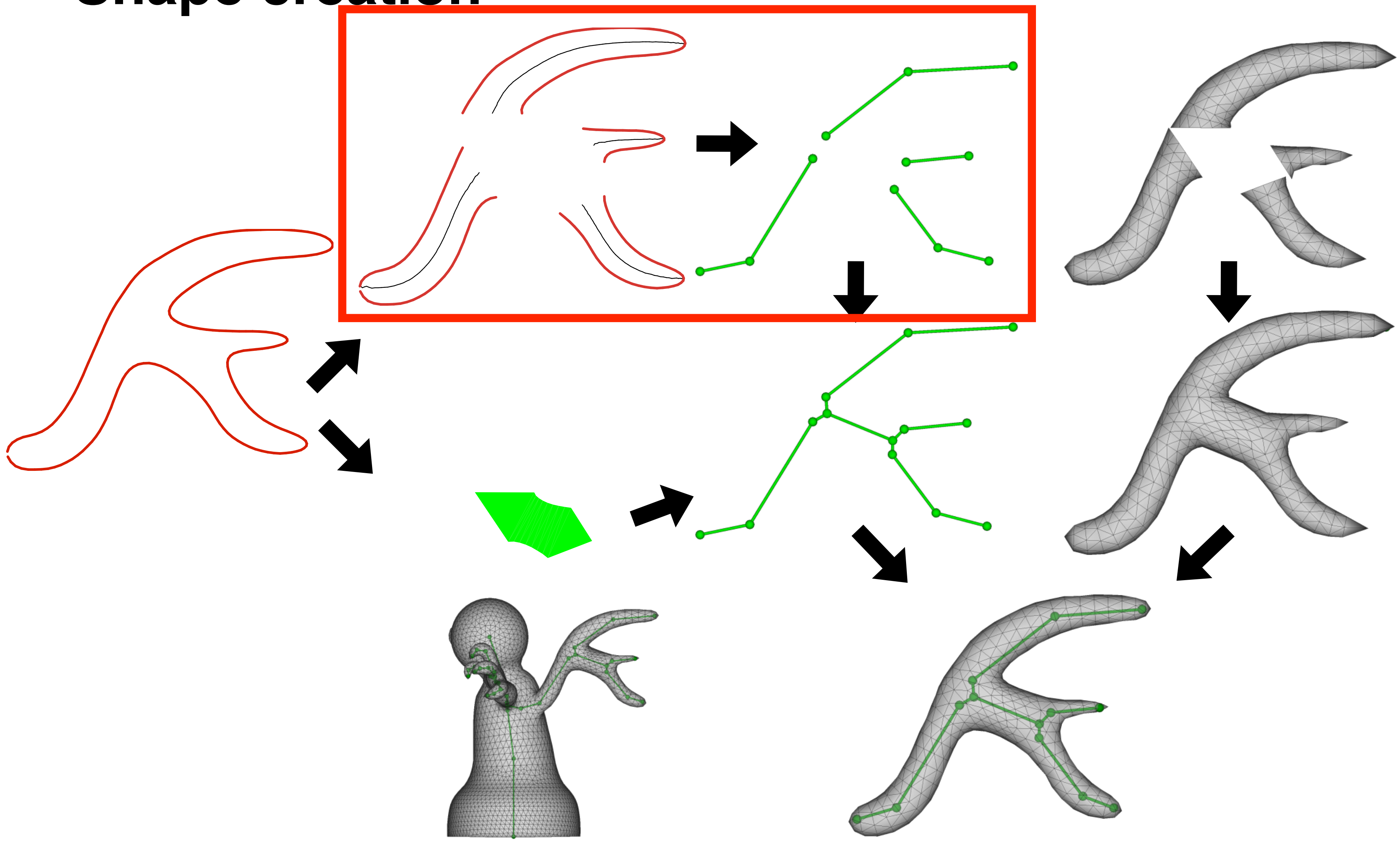
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Shape creation



Thank you. So let's see what's going on behind the scenes.

As you've seen in the demo the user can sketch the silhouette of the desired 3D shape, and our system generates a surface composed of smoothly joined generalized cylinders, a skeleton, and skin weights attaching the surface to the skeleton.

First The user's sketched curve is closed and treated as a planar polygon

[click]

and then chordal axis transform is applied to this polygon, and by refining the results we obtain a decomposition into cylindrical regions [click] with well defined local symmetry and connecting regions lacking such symmetry. These regions are then treated in a different way.

For the cylindrical regions high quality skeletons are extracted using a modification of the DP algorithm, [c] and generalized cylinder surfaces are created by using the 2D local symmetry axes as cylinder axes. [c]

Then using the information from the connecting regions the skeleton parts are connected to create an easy to use low DOF skeleton [c]

Similarly generalized cylinder surfaces are connected at connecting regions resulting in a watertight mesh. [c]

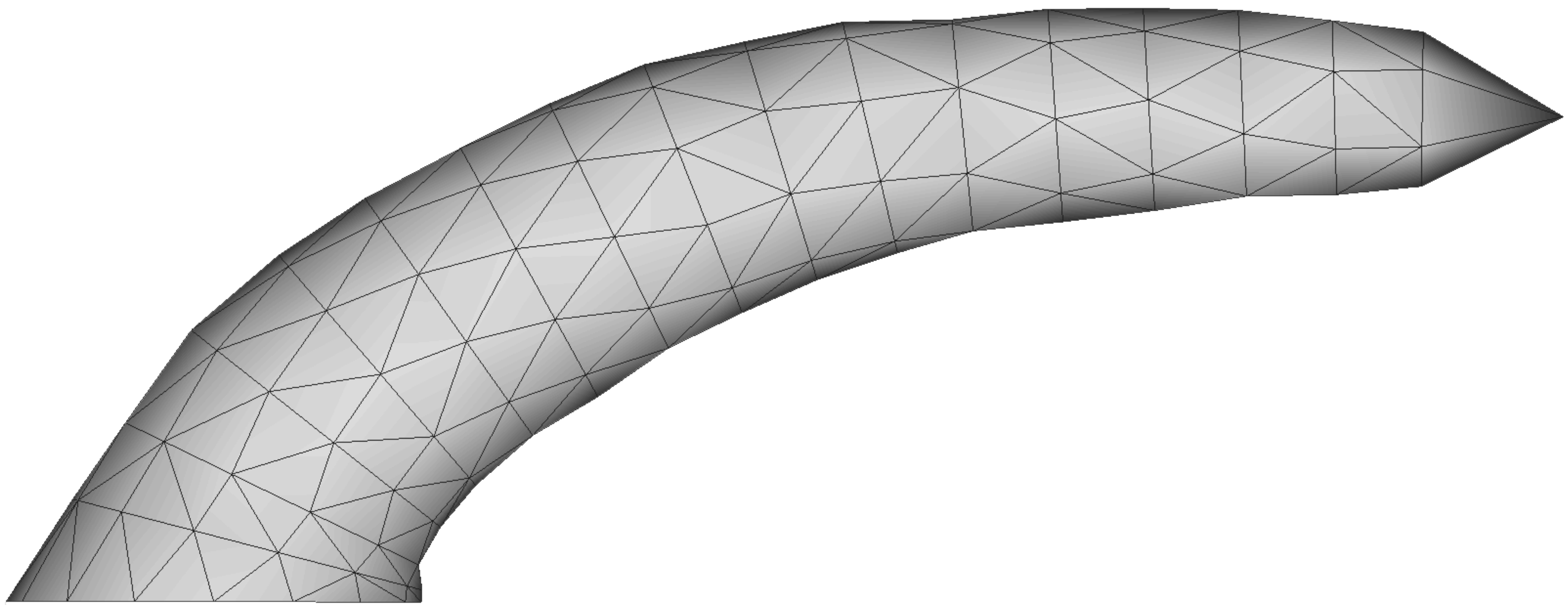
The surface is then attached to the skeleton by computing smooth skin weights following the Pinnochio method of Baran and Popovic.

This skinning algorithm has two parts: bounding the surface vertices to bones of the skeleton and ensuring smoothness by solving a Laplacian system. The bounding for vertices are trivially known for vertices in cylindrical regions, so it only has to be computed for vertices on connecting regions.

The user can then combine the resulting model with an existing one. [c] The surfaces and skeletons are merged [c], and the skin weights must be updated. While the same algorithm could be used again, it slows considerably as the complexity of the shape increases. [c] But observe that only the skin weights of vertices near the merge boundary, a small fraction of all vertices are noticeably affected. This allows for a local skin weight re-computation that greatly accelerates the process. [c]

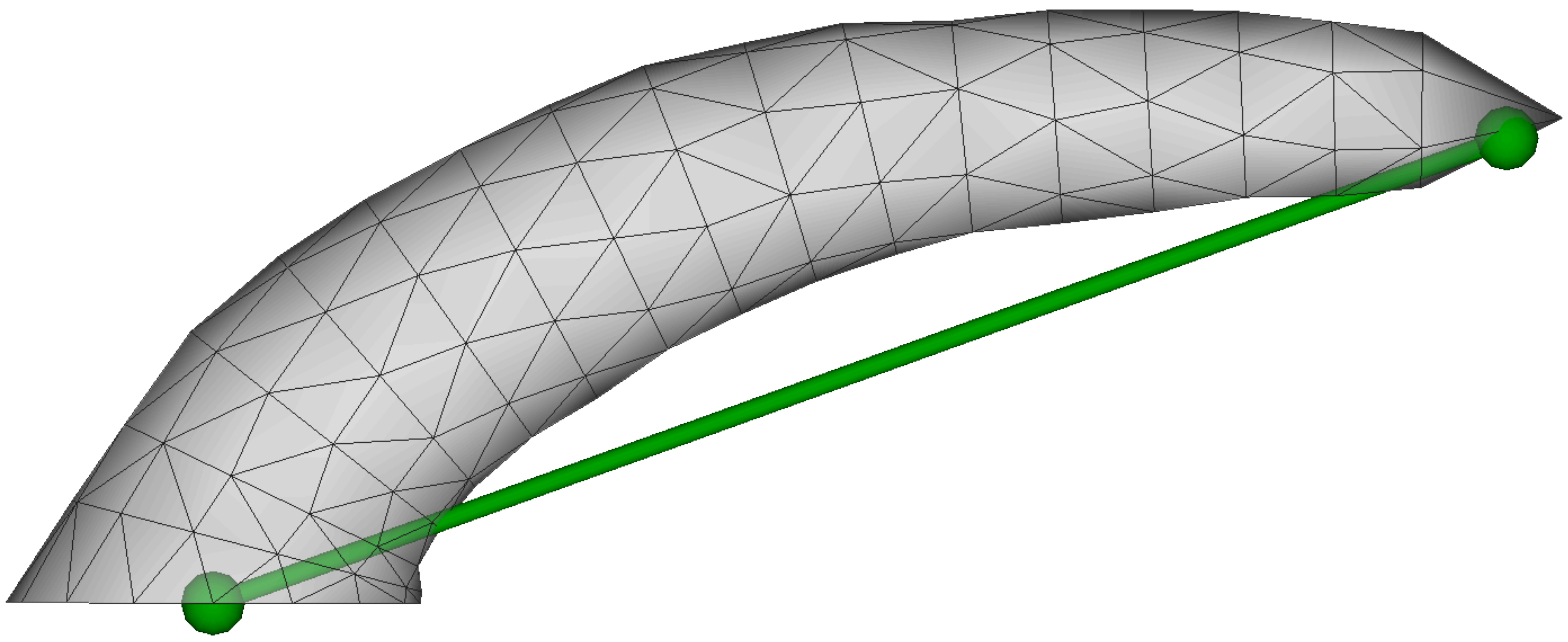
Because of the time constraint, now I will talk about the details of the skeleton construction for the cylindrical regions, if interested the other steps can be found in our paper.

Skeletonization



The 3D surface is created by using the 2D local symmetry axes of the cylindrical regions as the axes of generalized cylinders. This is the surface for which we need a bone skeleton. It should obviously have a linear structure, so the question remains how many joints should we have and where should they be placed

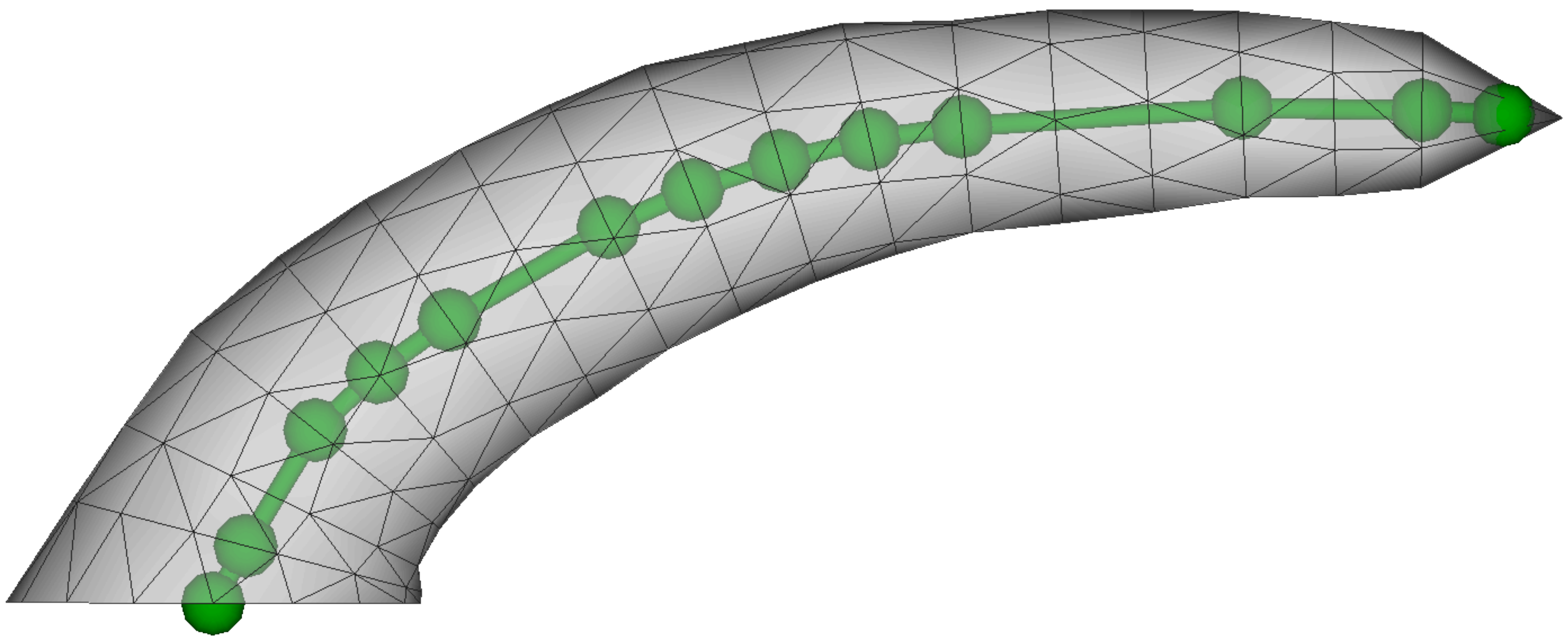
Skeletonization



Sponsored by ACM SIGGRAPH

An easy solution would be having a single bone for the entire cylindrical region. But this way the whole cylindrical region stays rigid, we can only pose it relatively to the other parts of the shape

Skeletonization

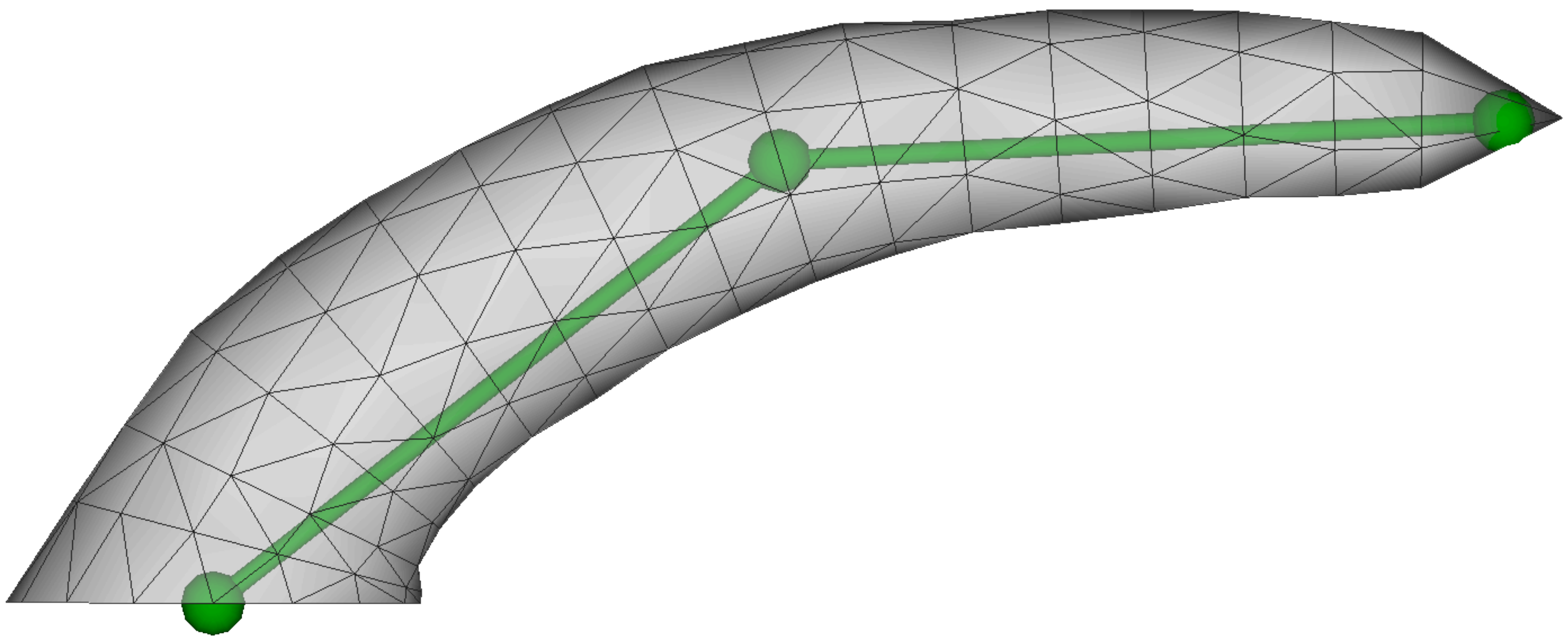


Sponsored by ACM SIGGRAPH

The other extremum is having many joints and this way many degrees of freedom.

But then posing becomes tedious using forward kinematics, and if we use IK, deformations may occur in regions that we would want to stay rigid.

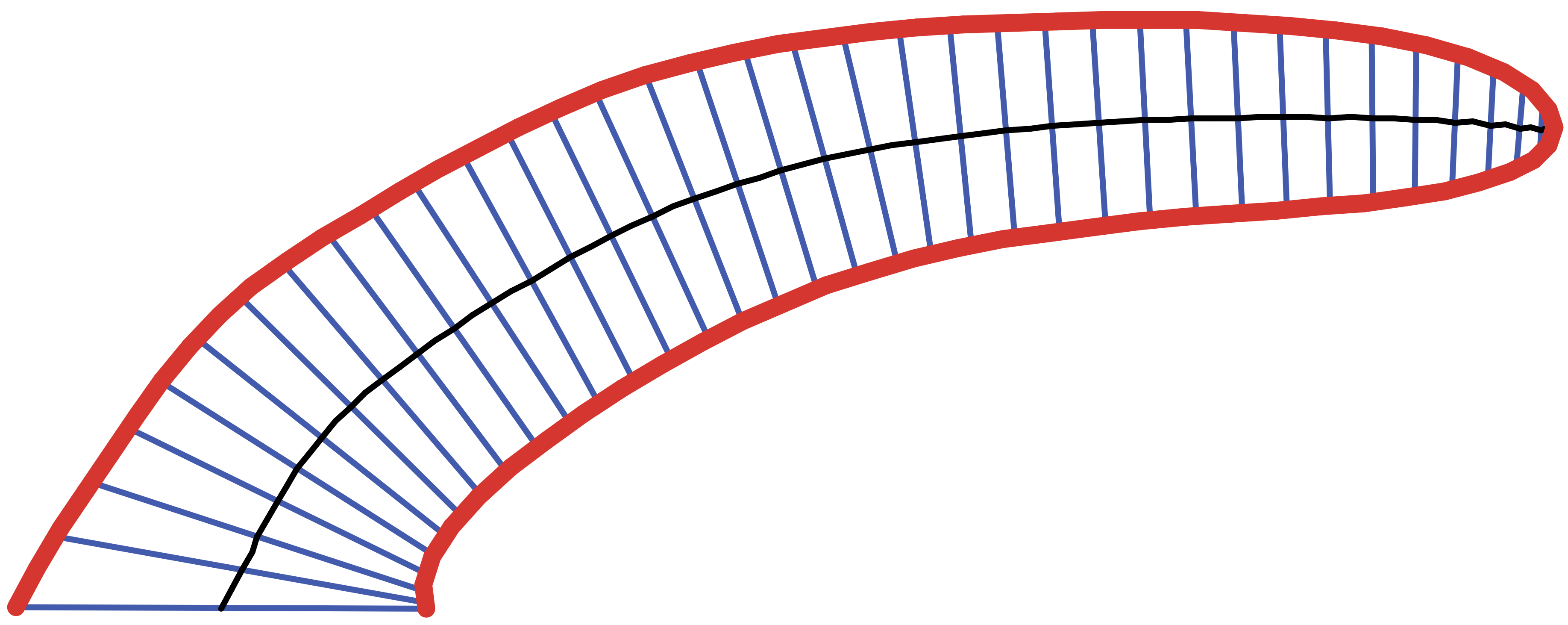
Skeletonization



Sponsored by ACM SIGGRAPH

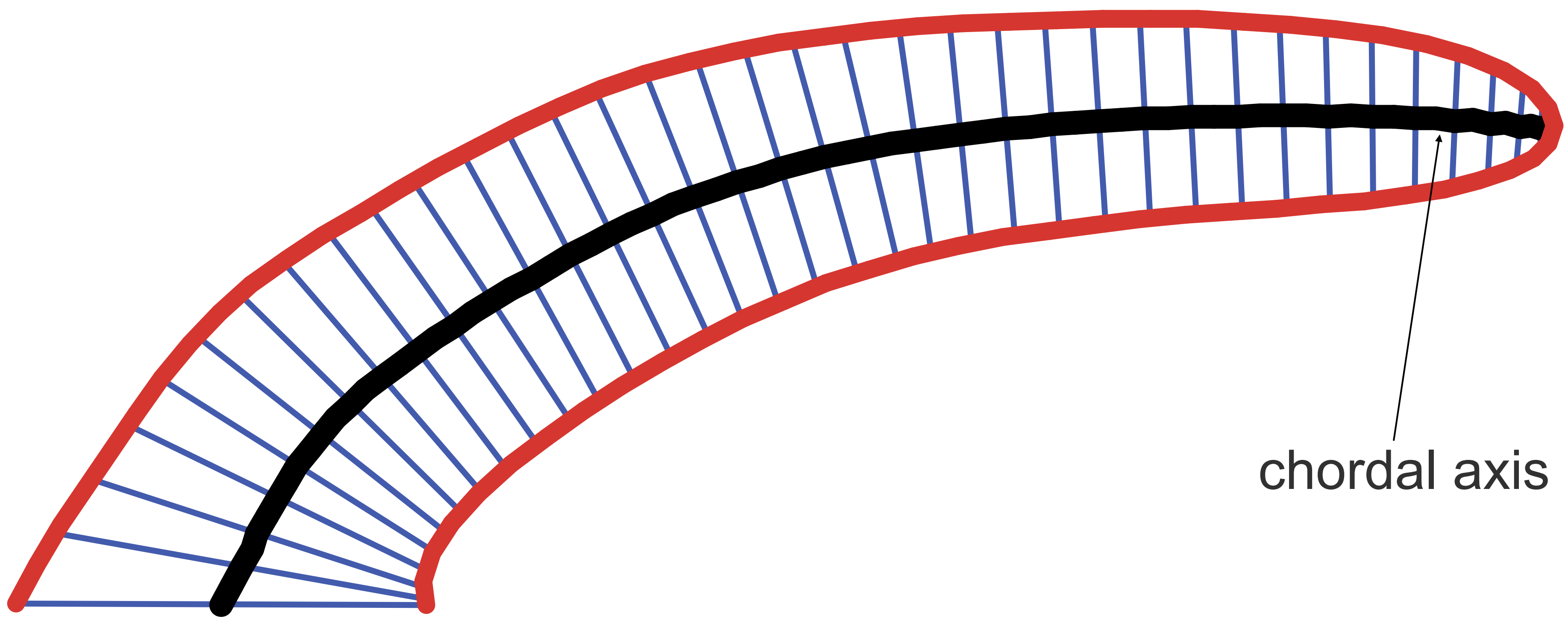
So we have to find a balance between these extrema, we have to find a tradeoff between usability and the number of degrees of freedom

Skeletonization



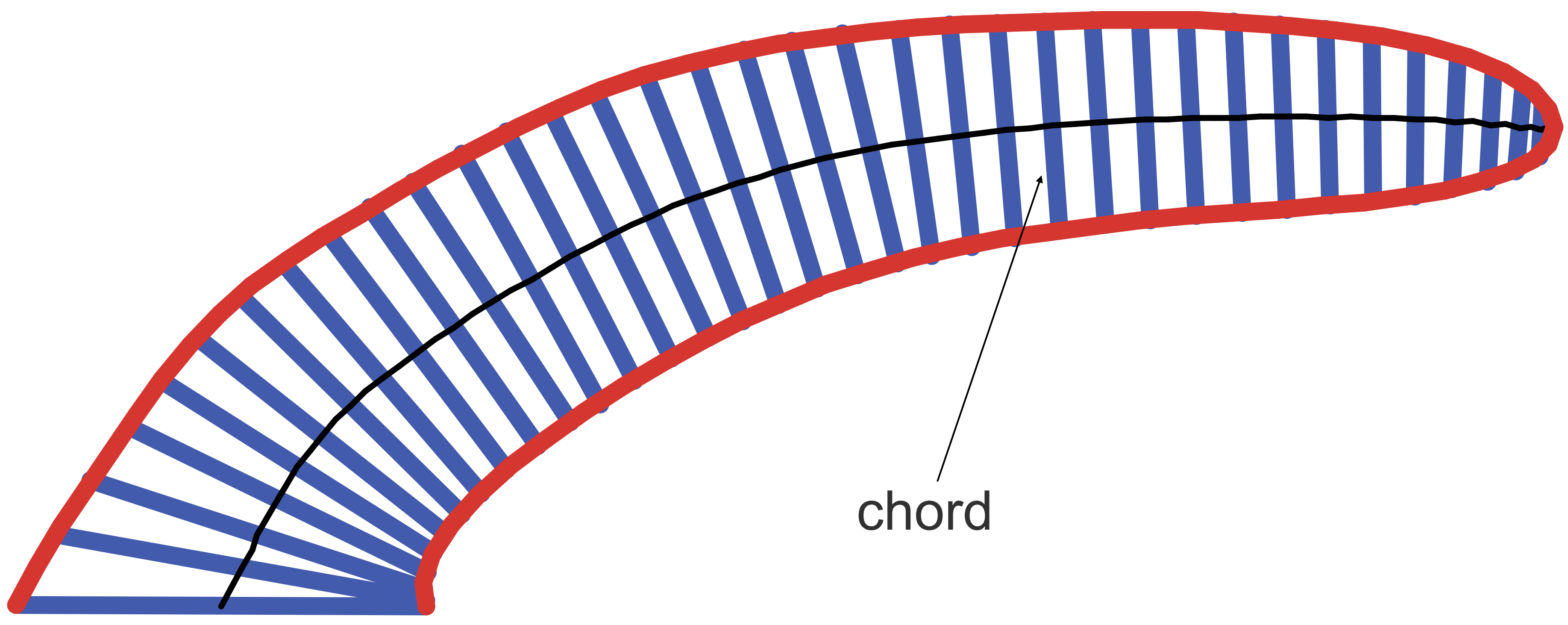
So what do we have to work with? For each cylindrical region the local symmetry is described by the chords and the chordal axis.

Skeletonization



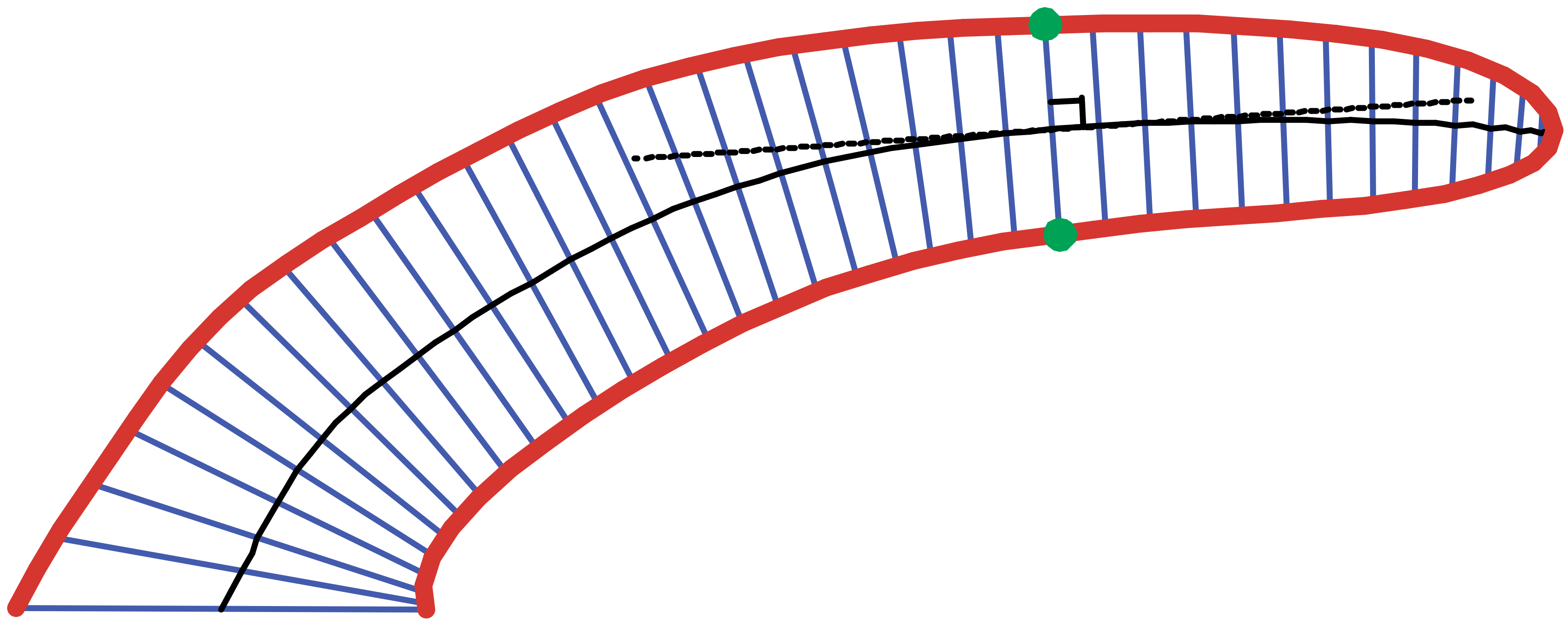
The chordal axis is an approximation of the local symmetry axis

Skeletonization



And the chords tell us the thickness of the region along the axis

Skeletonization



Sponsored by ACM SIGGRAPH

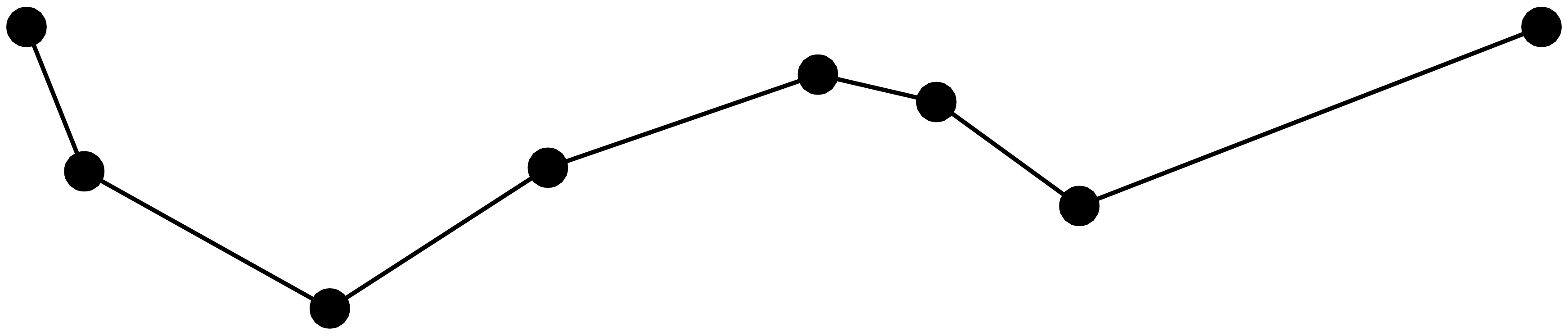
The end points of the chords are approximately symmetric point pairs on the sketch.

The chordal axis represents well the 3D surface, so we use it for creating the bone-skeleton.

Since bones are straight, parts of the shape corresponding to a single bone should also be roughly straight.

The bone skeleton then can be constructed by down sampling the chordal axis to line segments. A popular algorithm for such poly-line simplification is the DP algorithm

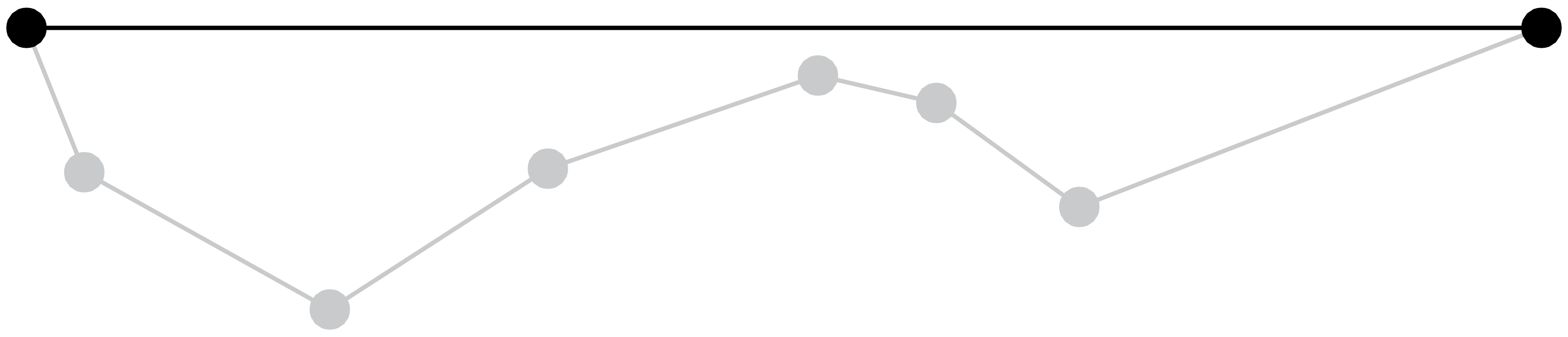
Douglas-Peucker algorithm



Recall, the DP algorithm is a greedy recursive algorithm.

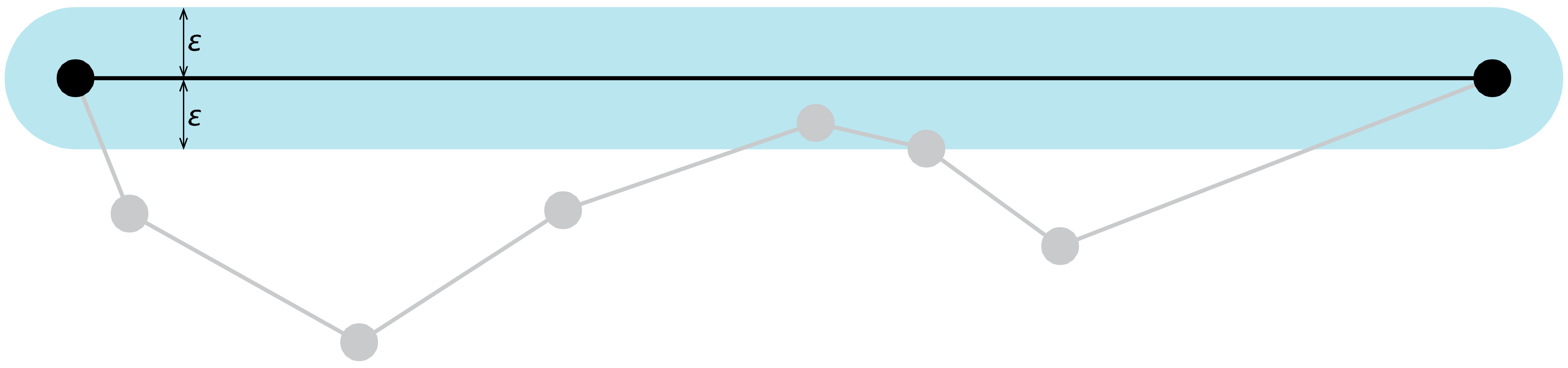
It down samples the input poly line represented by a series of points, such that all of the original points lie within a predetermined error threshold of the resulting poly line

Douglas-Peucker algorithm



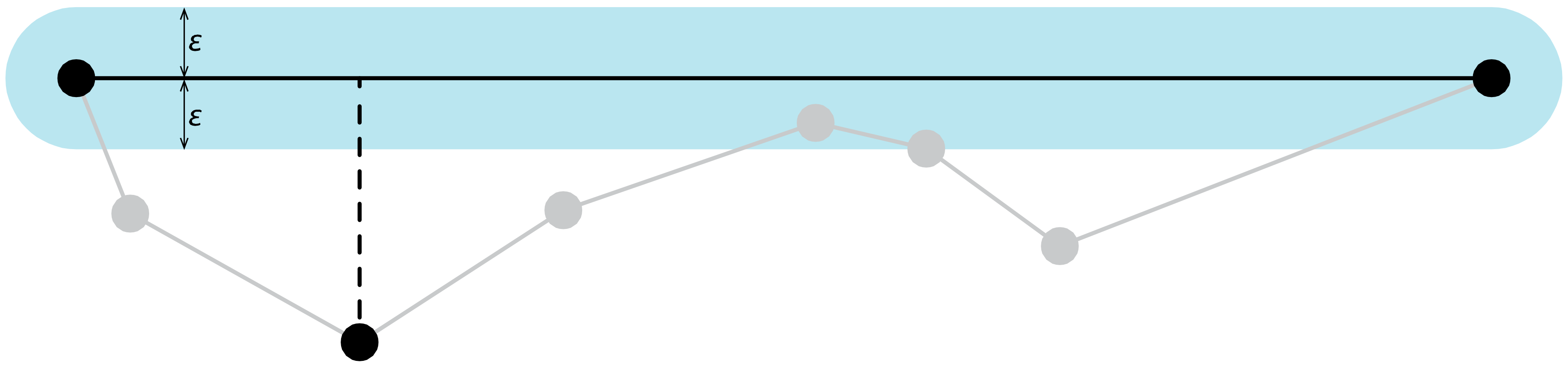
A line segment is fit between the first and last points

Douglas-Peucker algorithm



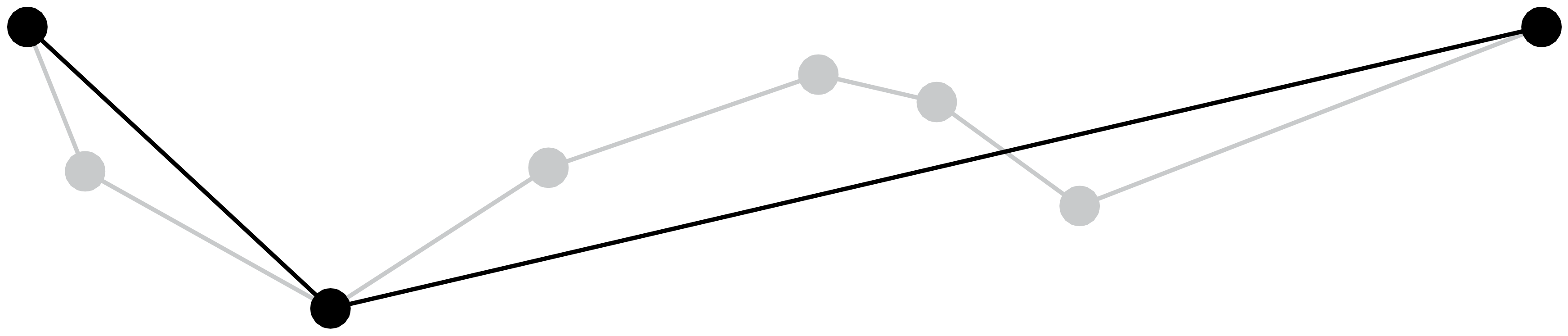
If not all the points are within the threshold of this segment

Douglas-Peucker algorithm

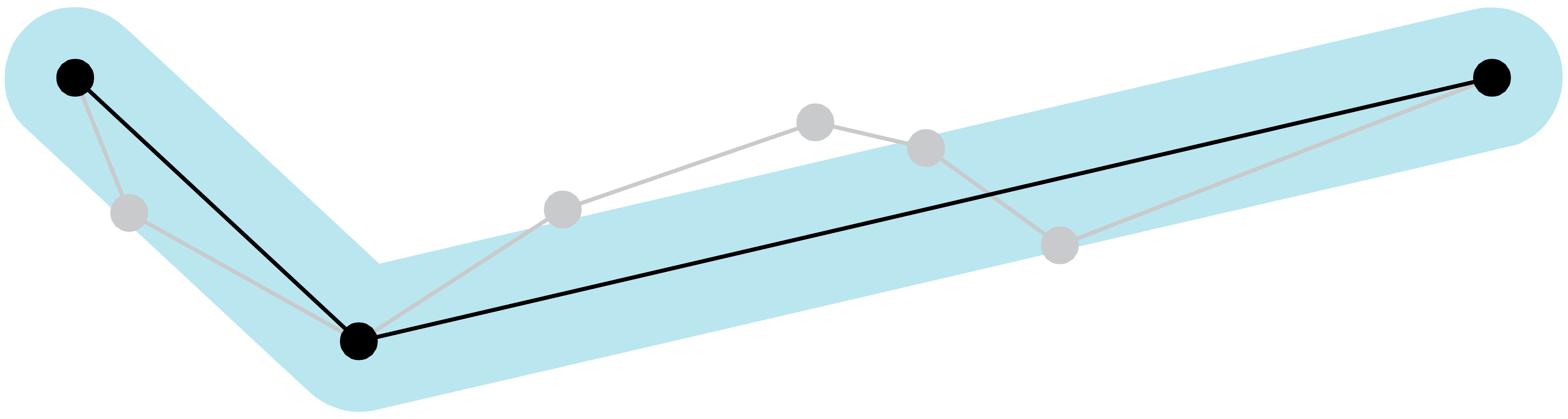


The point farthest from the line segment is chosen and two recursive calls are made, for the parts before and after the chosen point

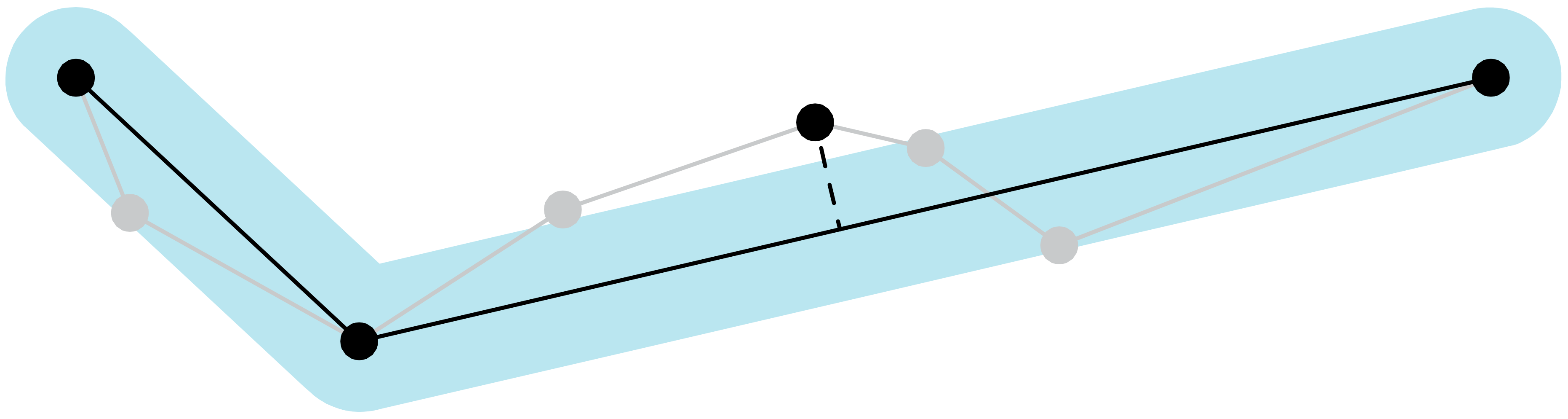
Douglas-Peucker algorithm



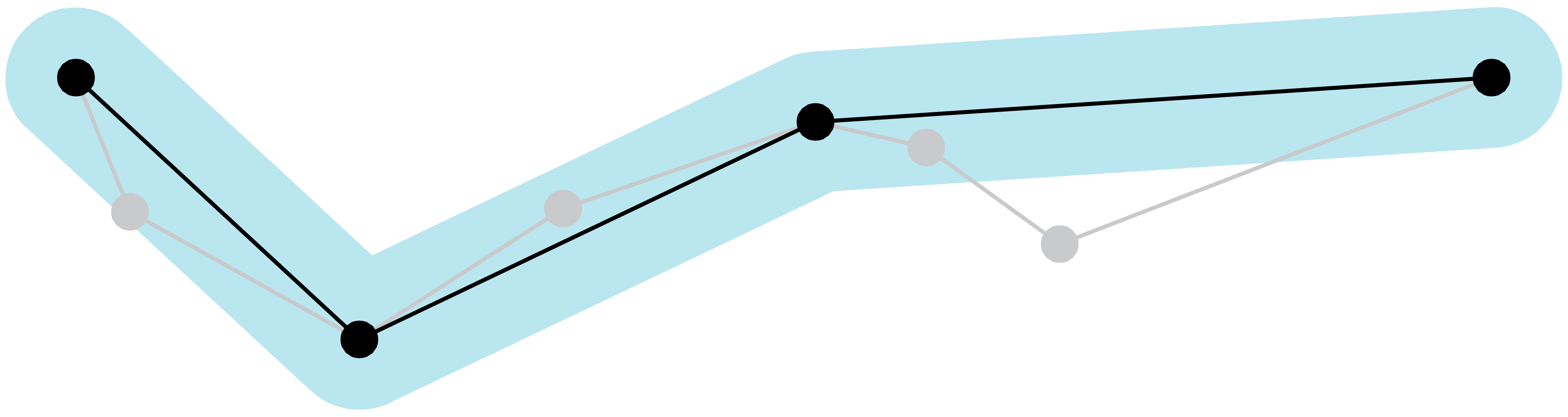
Douglas-Peucker algorithm



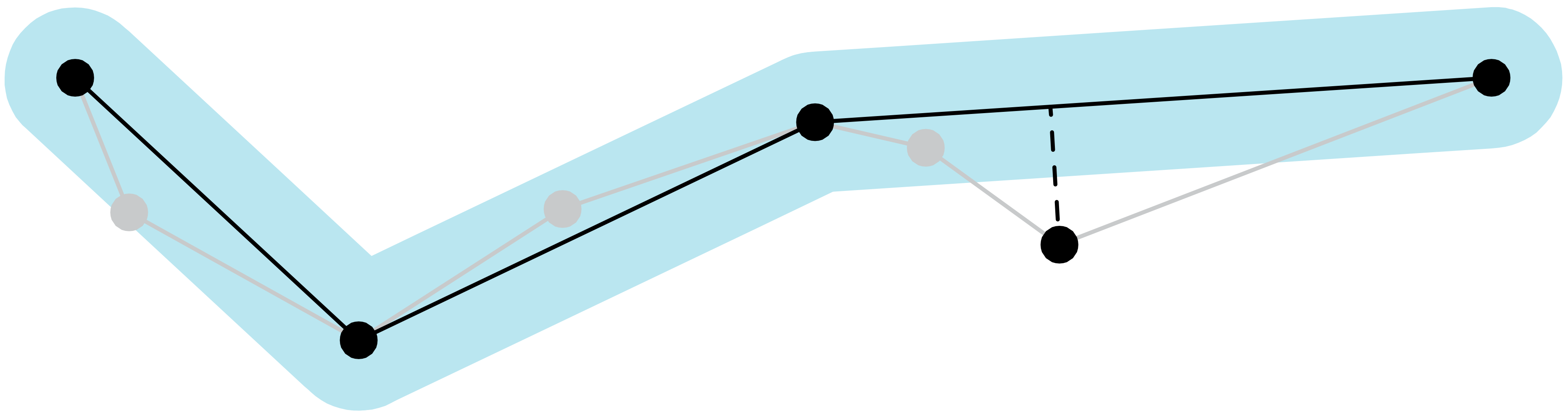
Douglas-Peucker algorithm



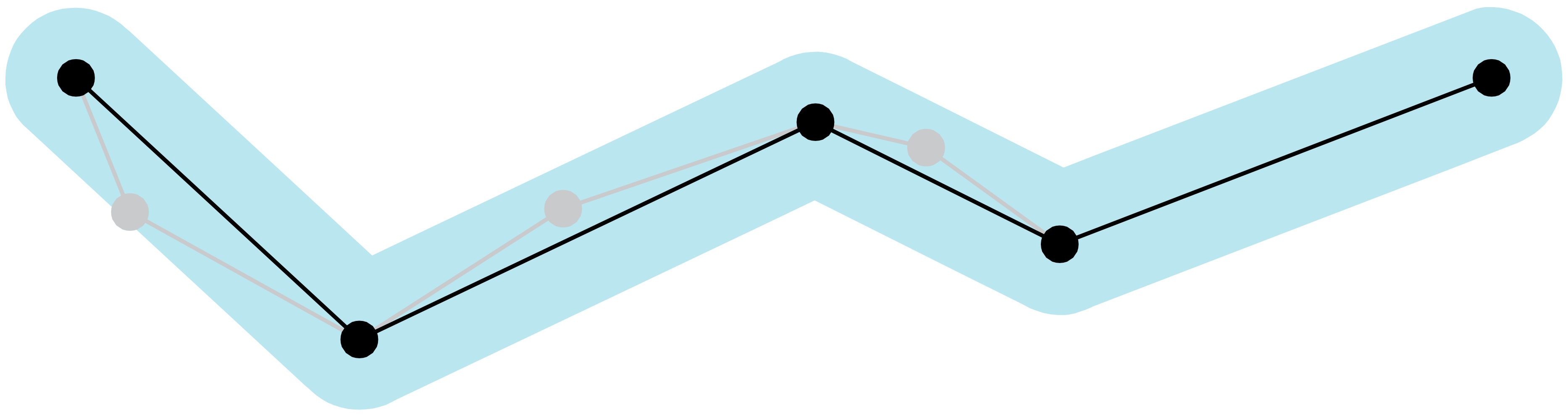
Douglas-Peucker algorithm



Douglas-Peucker algorithm

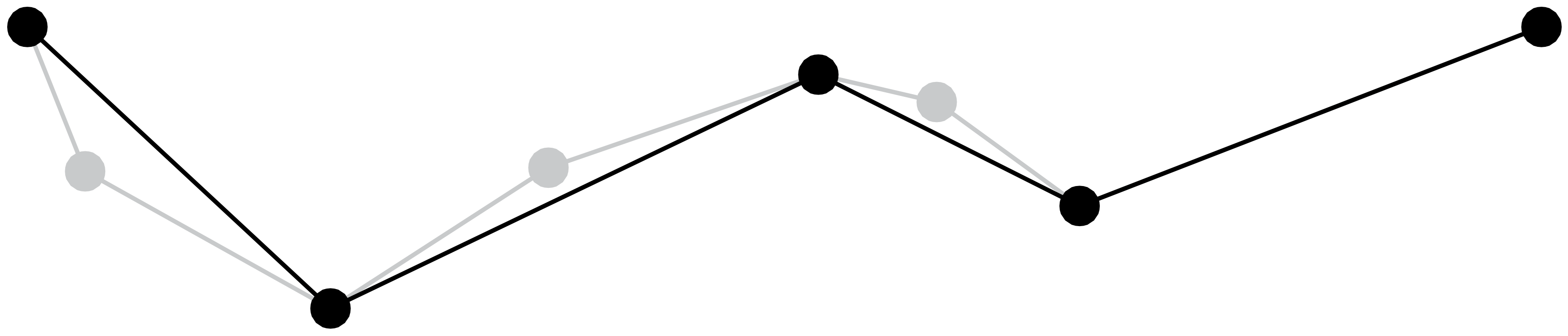


Douglas-Peucker algorithm



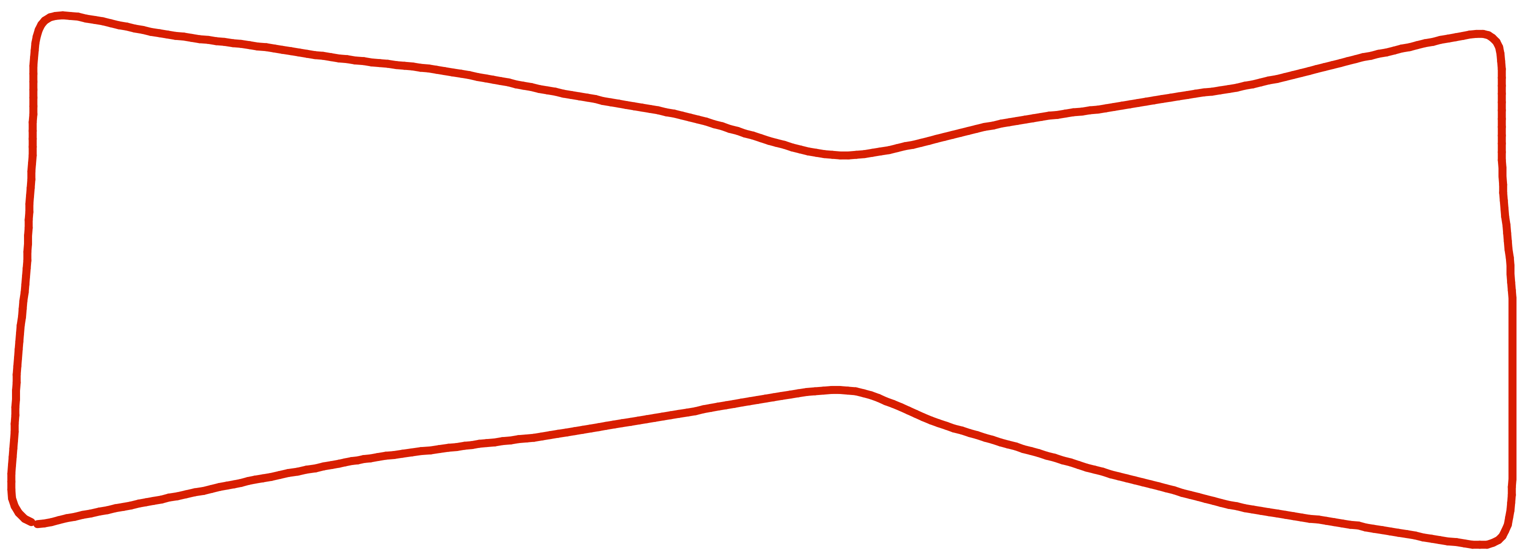
When all points are within the threshold

Douglas-Peucker algorithm



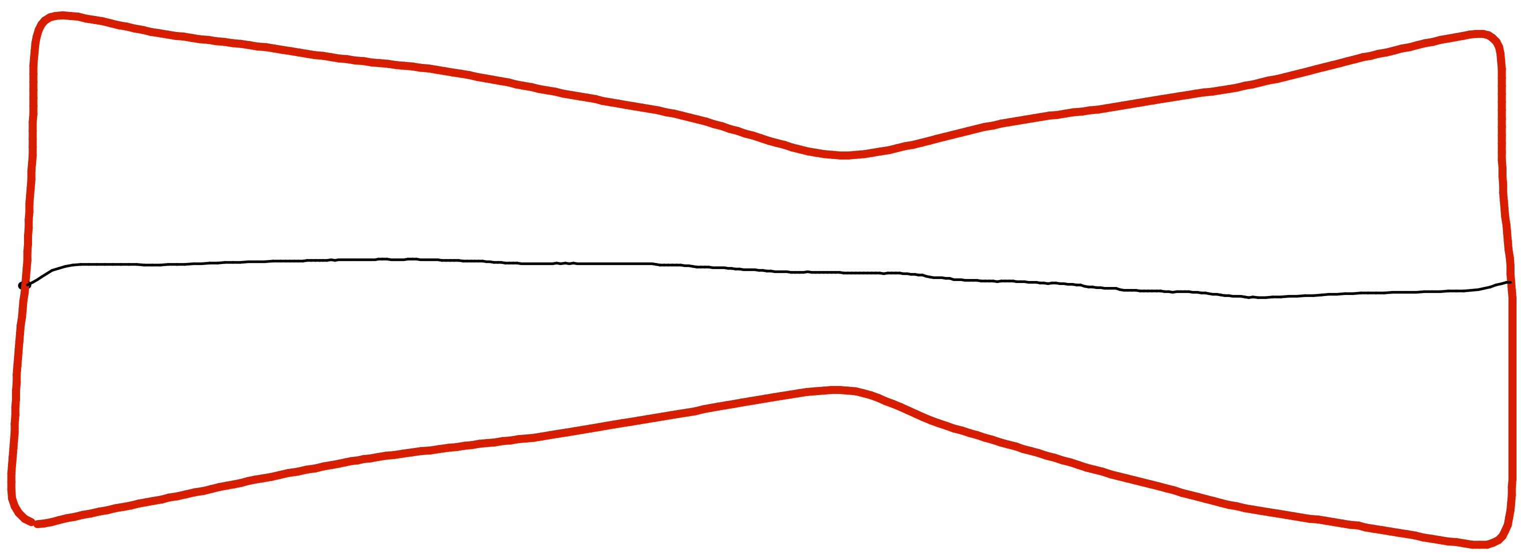
The algorithm stops, and we can use the resulting line segments as the bones of the skeleton

What happens if…



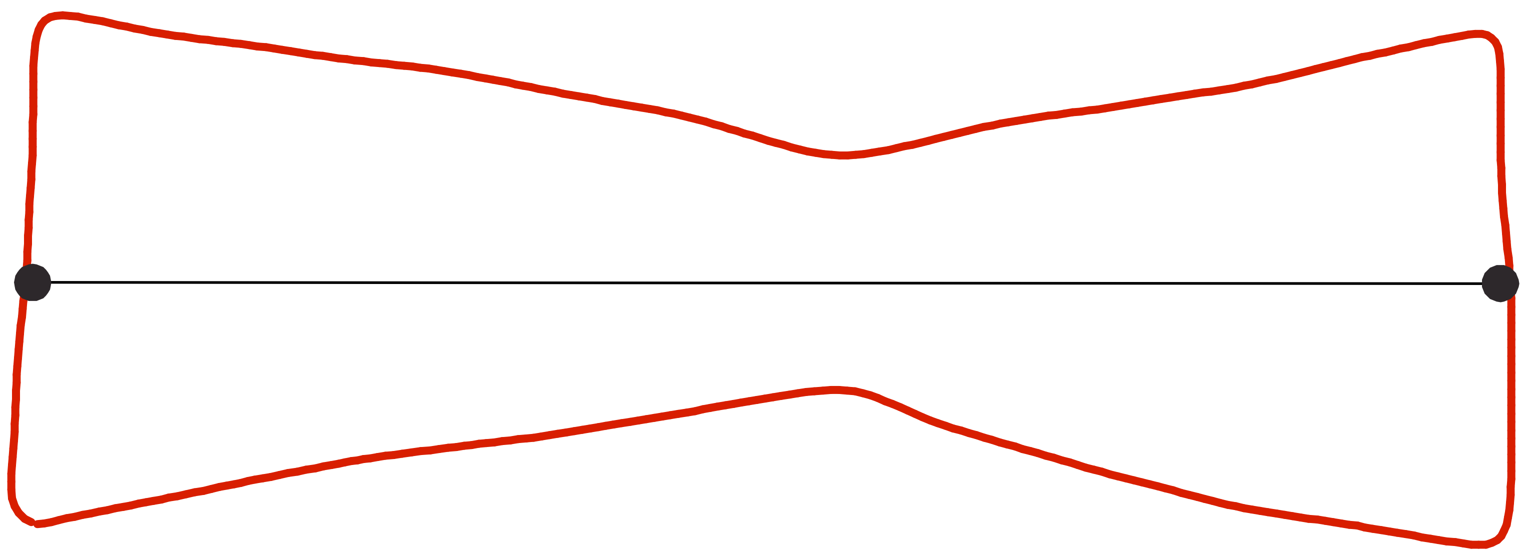
But life is not always that simple and we might have a shape like this

…the chordal axis is almost straight



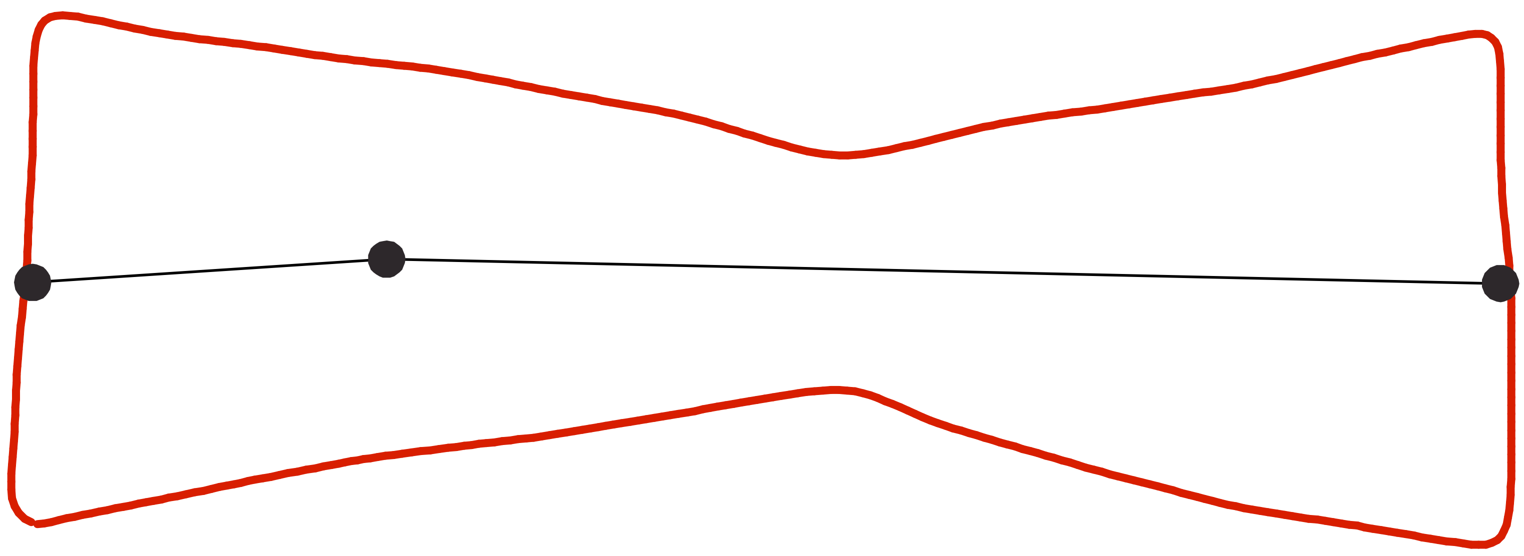
Where the chordal axis is almost straight, but the thickness varies substantially along the chordal axis

Result of Douglas-Peucker



Applying DP would result in a single line segment, therefore the skeleton would have a single bone, even though we feel that a shape like this does in fact have two parts, and therefore the skeleton should have two bones connected at a joint at the minimum thickness

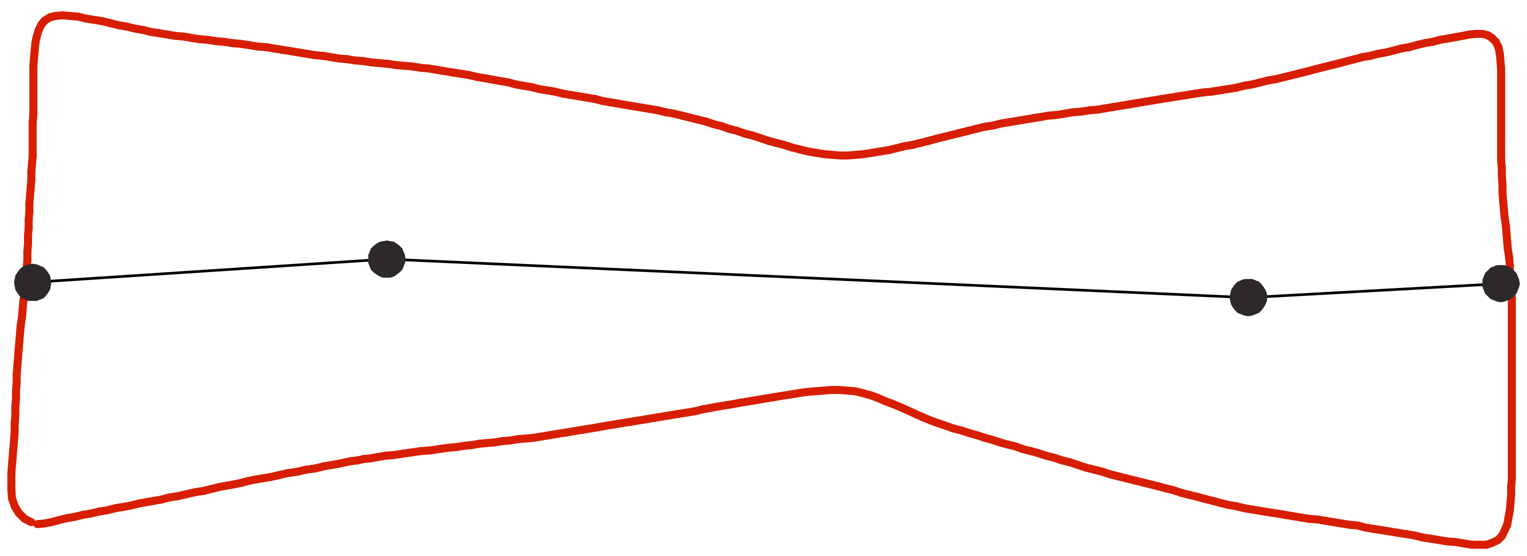
Result of Douglas-Peucker



Sponsored by ACM SIGGRAPH

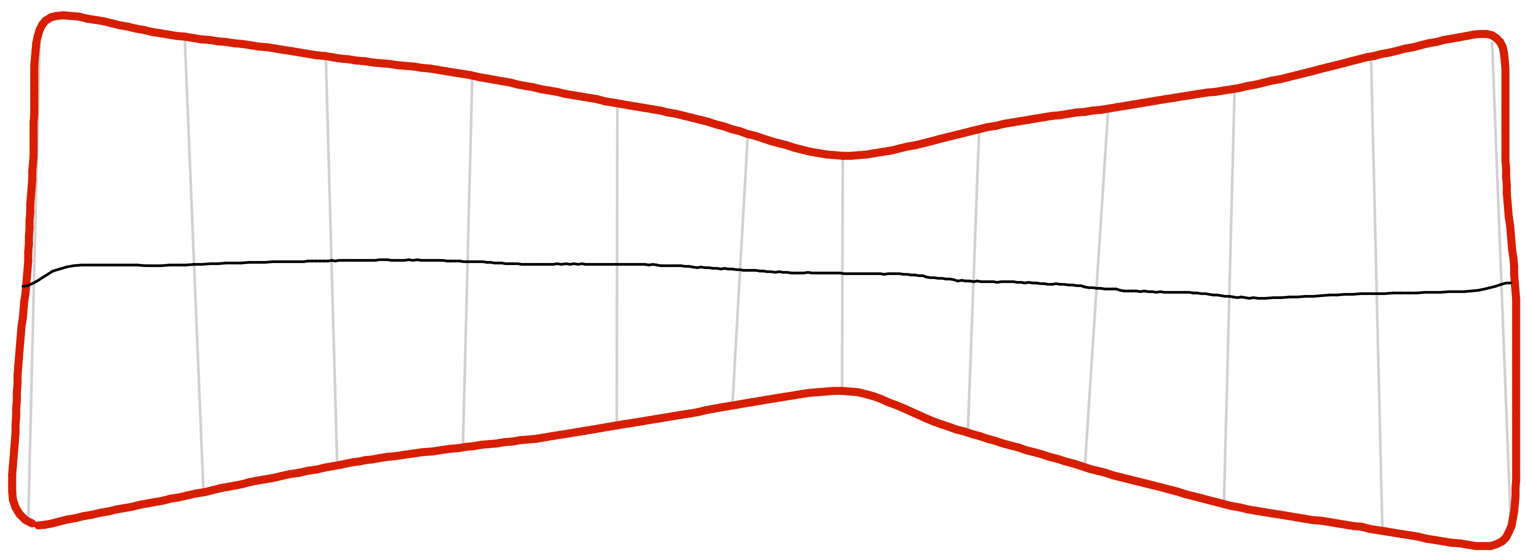
We can lower the error threshold to get more line segments, but the additional joints are not placed at the desired location

Result of Douglas-Peucker



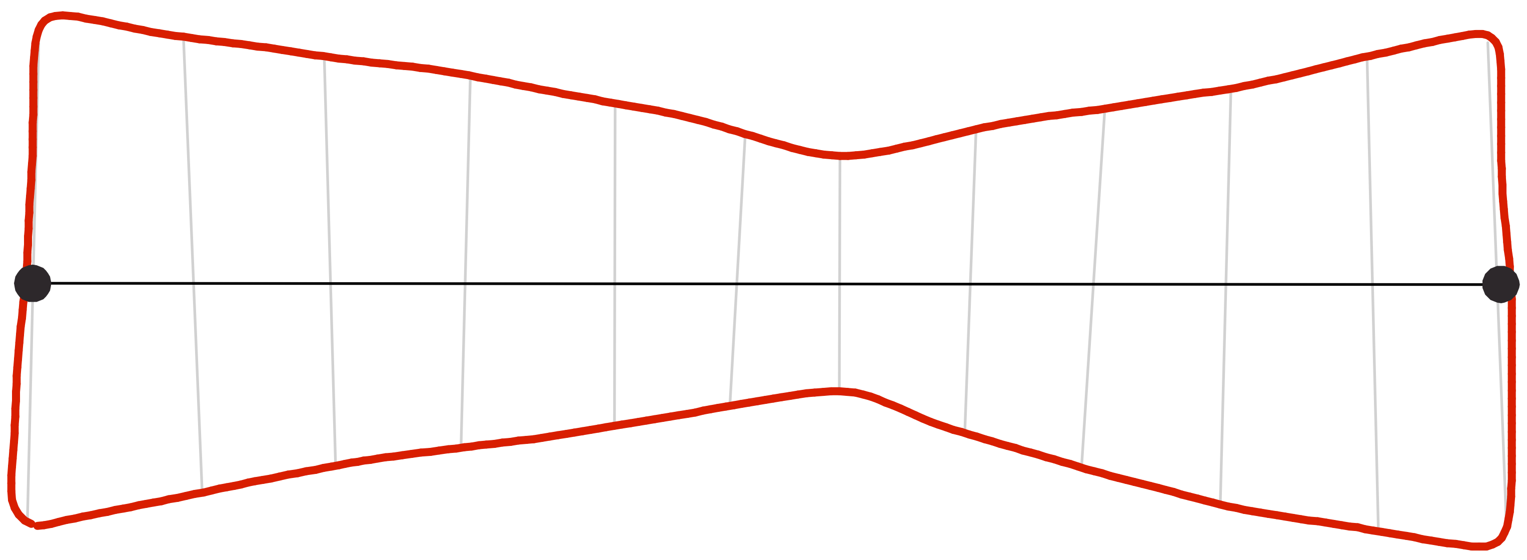
But instead at sort of random locations according to noise and slight perturbations in the chordal axis. This shouldn't really be a surprise, since we didn't use information on the thickness

Use thickness (chords)



So to alleviate this problem we extended the DP algorithm to also use thickness information from the chords. So instead of approximating the chordal axis with a poly line we are approximating the entire cylindrical region with symmetric trapezoids.

Fit line segment

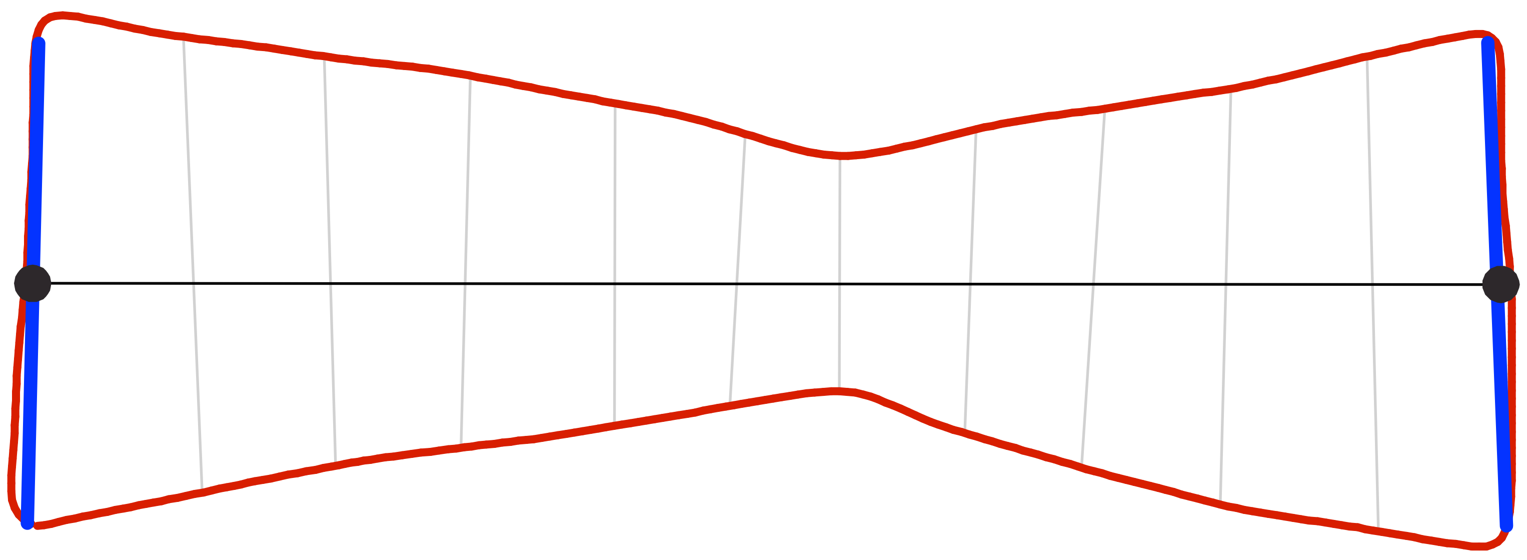


Sponsored by ACM SIGGRAPH

So the recursive function is modified in the following way:

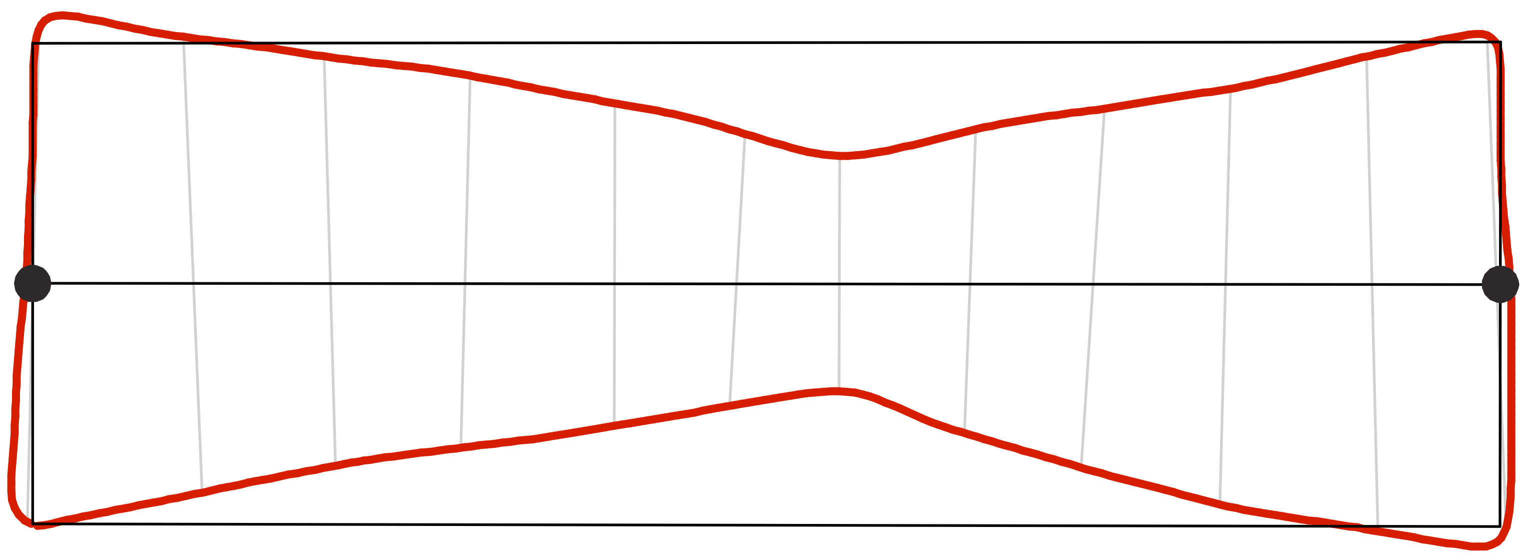
First a line segment is fit between the midpoints of the first and last chords

Fit trapezoid



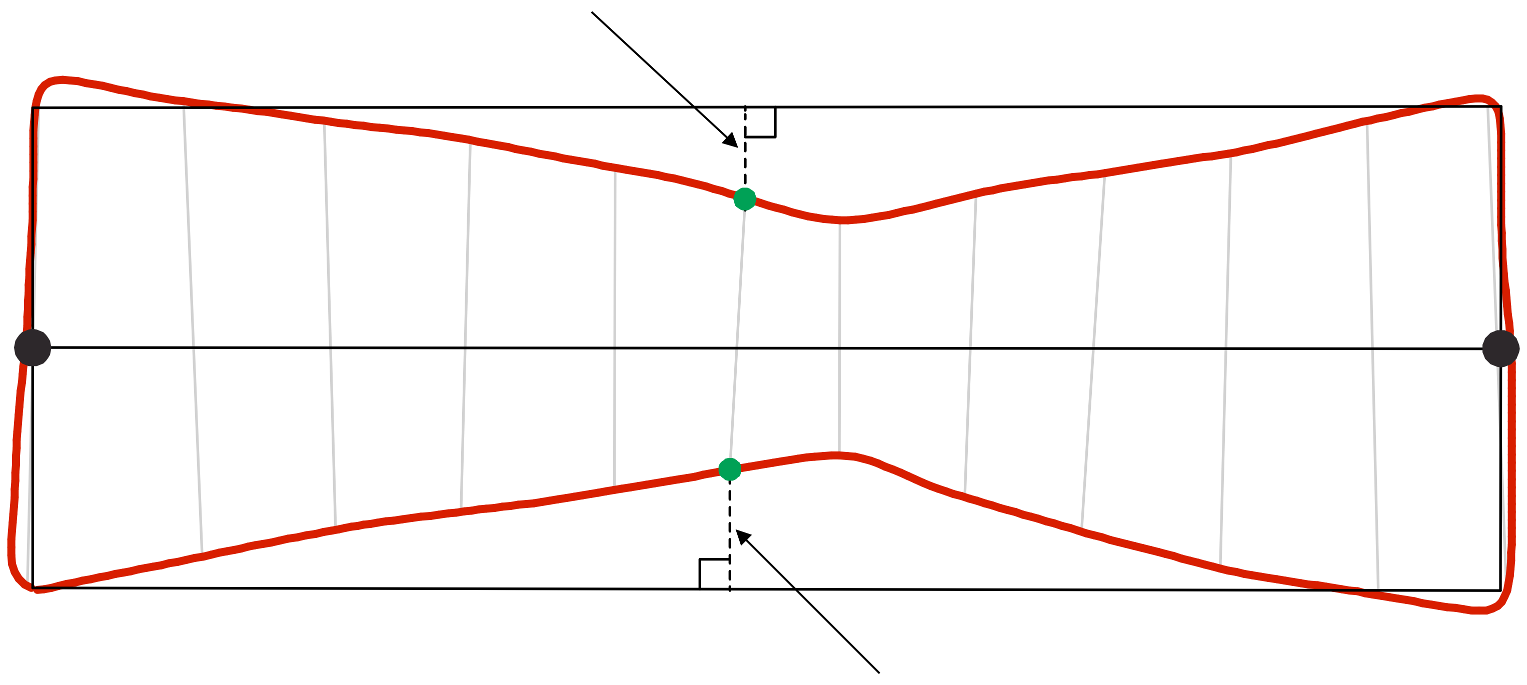
And then a symmetric trapezoid is also added using the line segment as its symmetry axis, and the bases of the trapezoid are determined by projecting the first and last chords onto the normal direction of the segment.

Fit trapezoid



Then we have to evaluate if it this trapezoid is a good approximation of the cylindrical region.

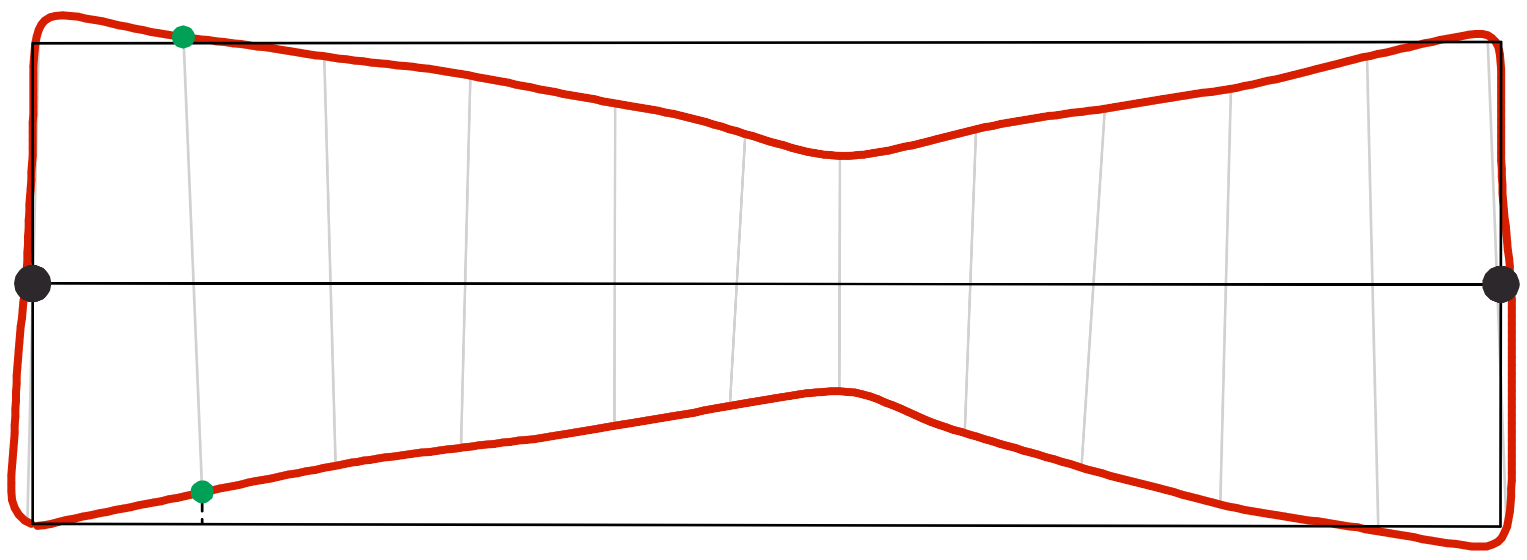
Cylindrical Douglas-Peucker error



Sponsored by ACM SIGGRAPH

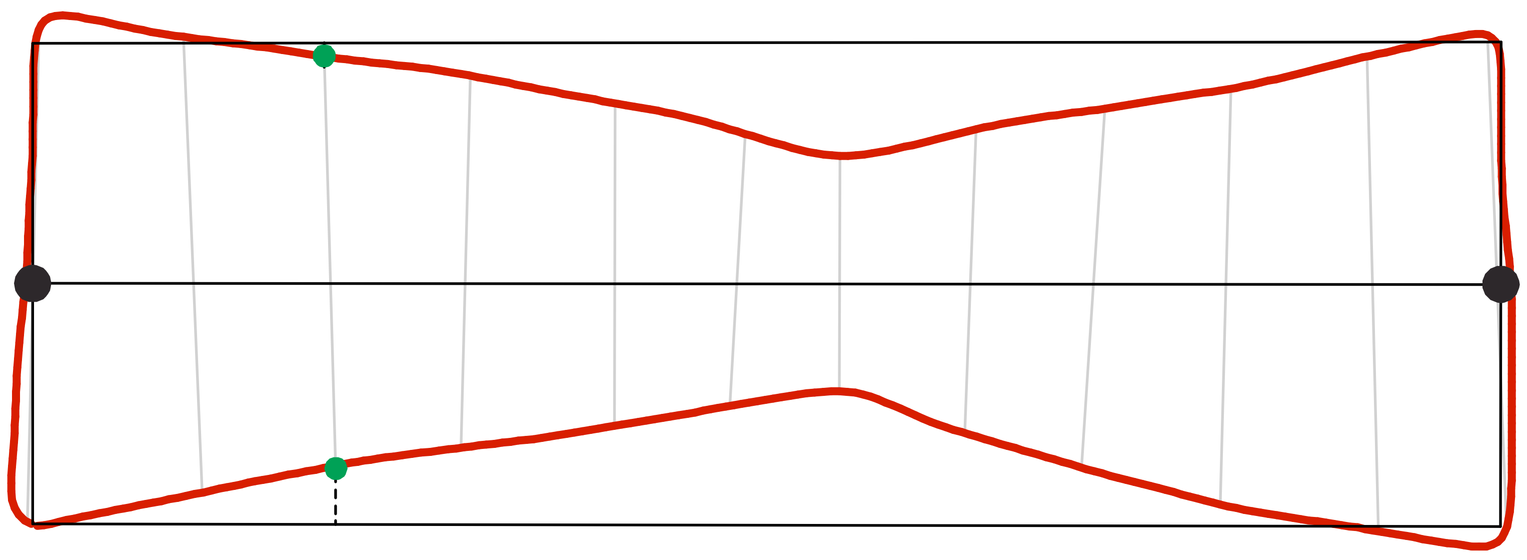
We define the error at each chord as the sum of the squared distances between the end points of the chord and the two sides of the trapezoid.

Cylindrical Douglas-Peucker error

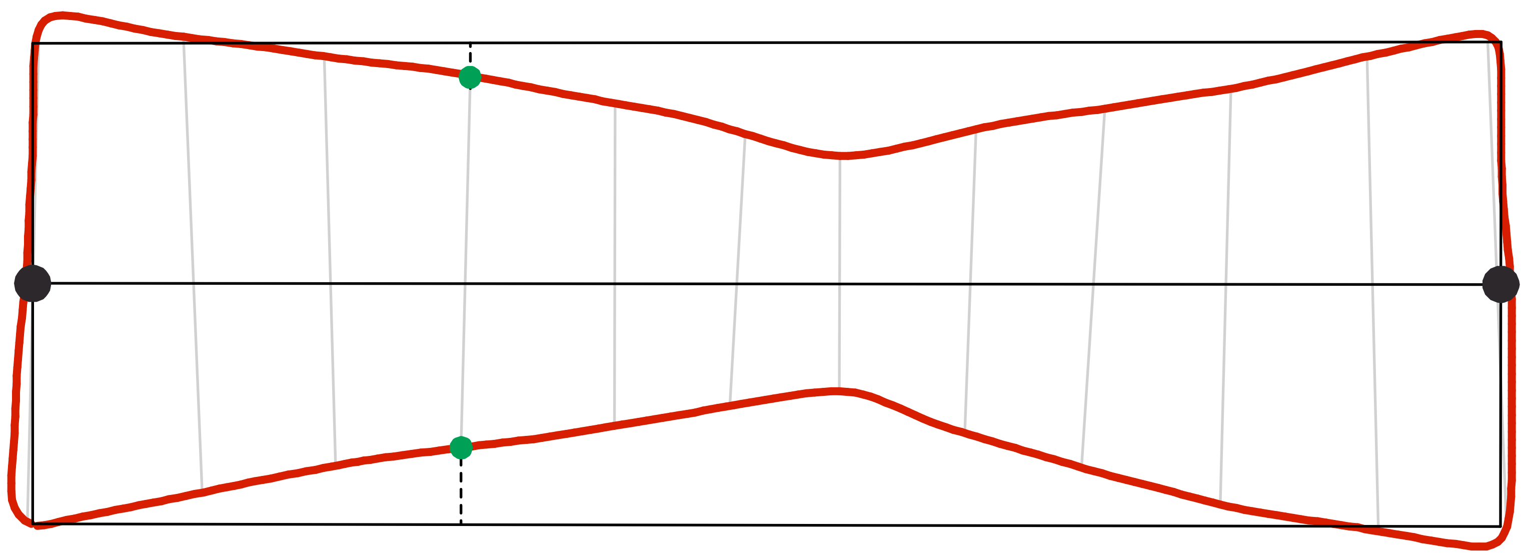


We compute the error at each chord

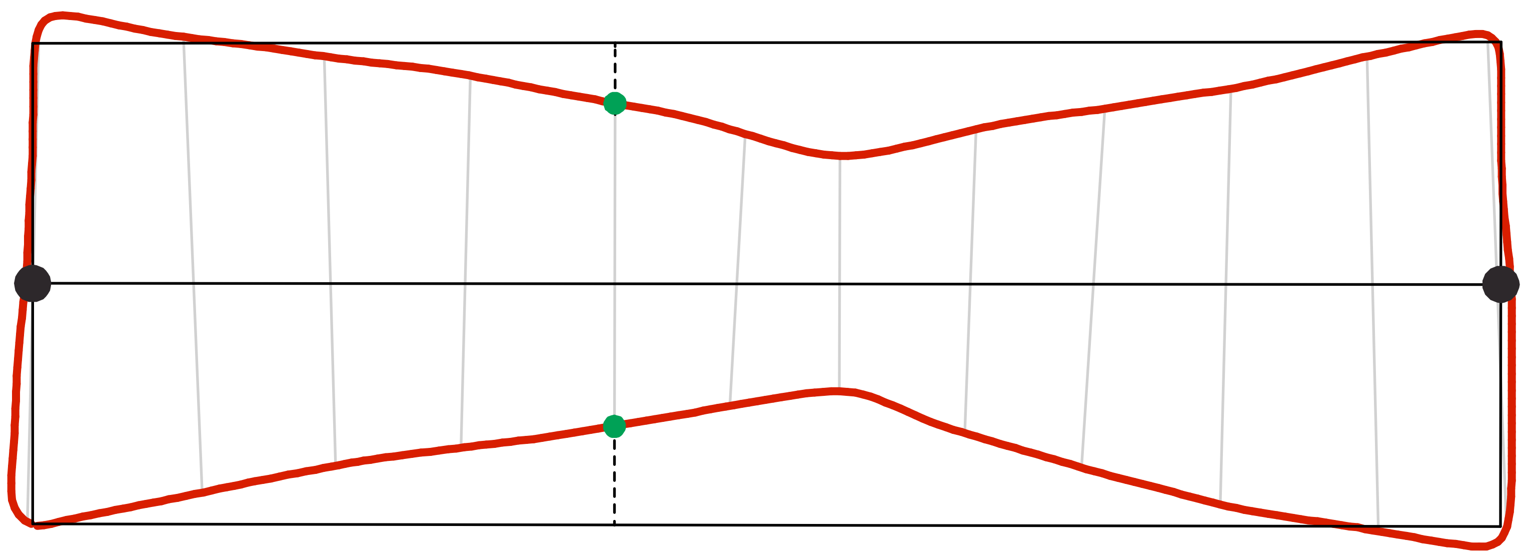
Cylindrical Douglas-Peucker error



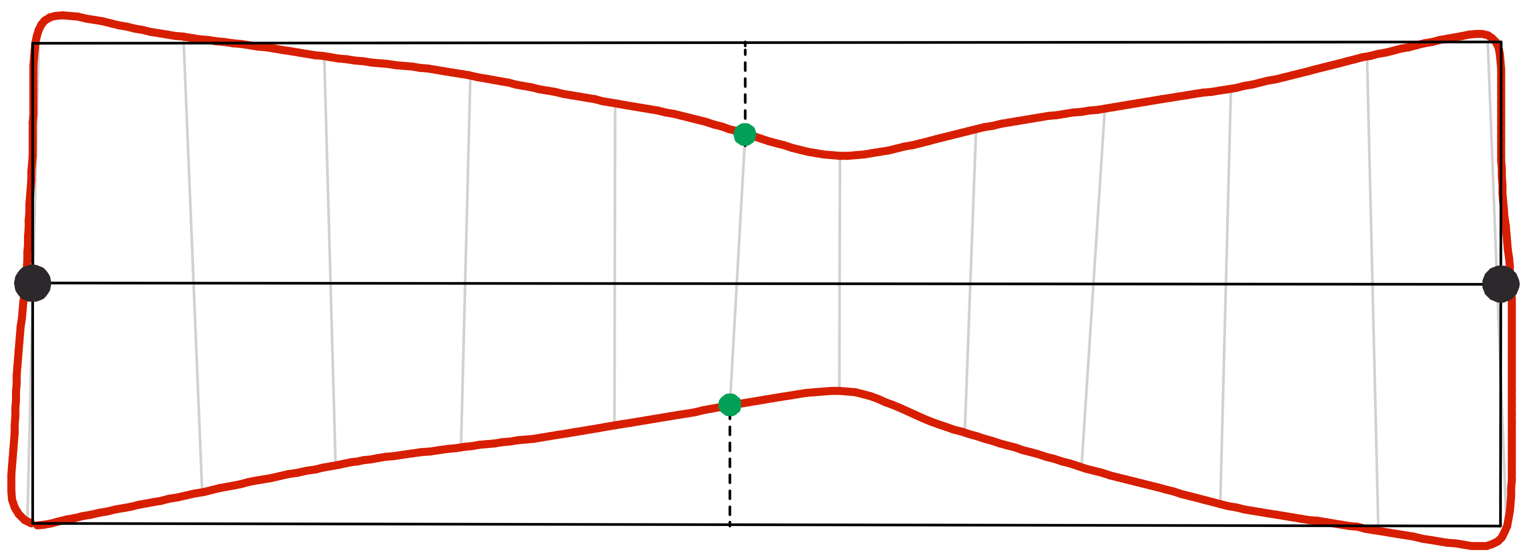
Cylindrical Douglas-Peucker error



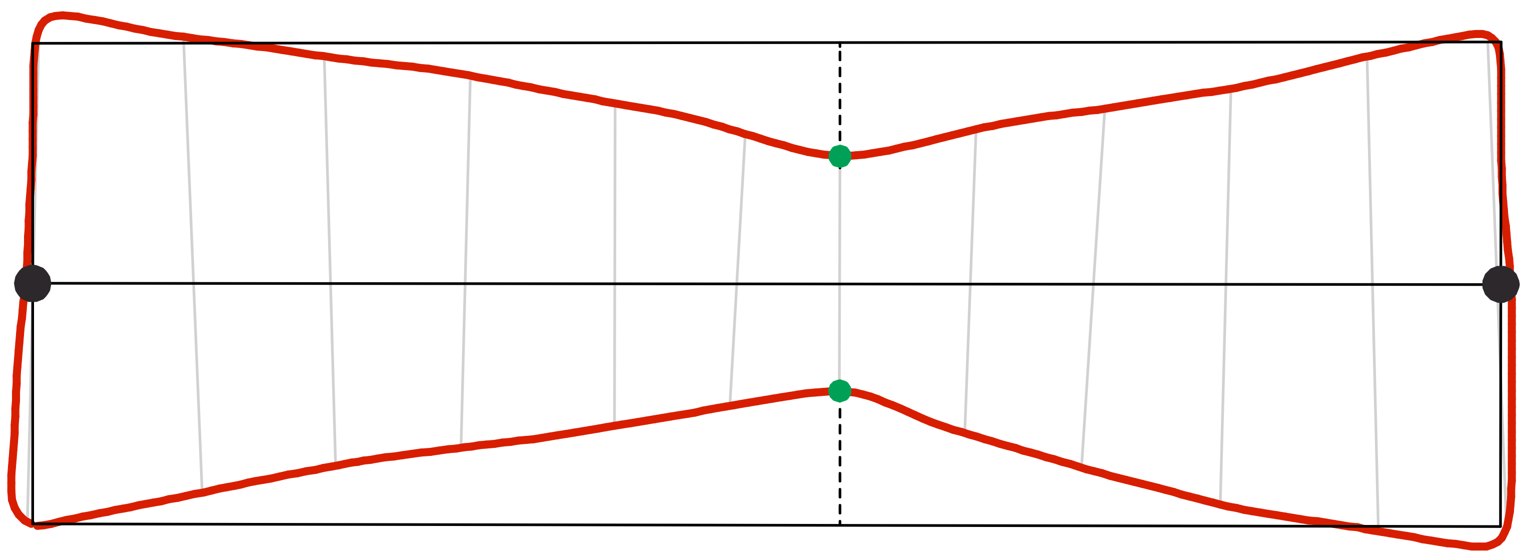
Cylindrical Douglas-Peucker error



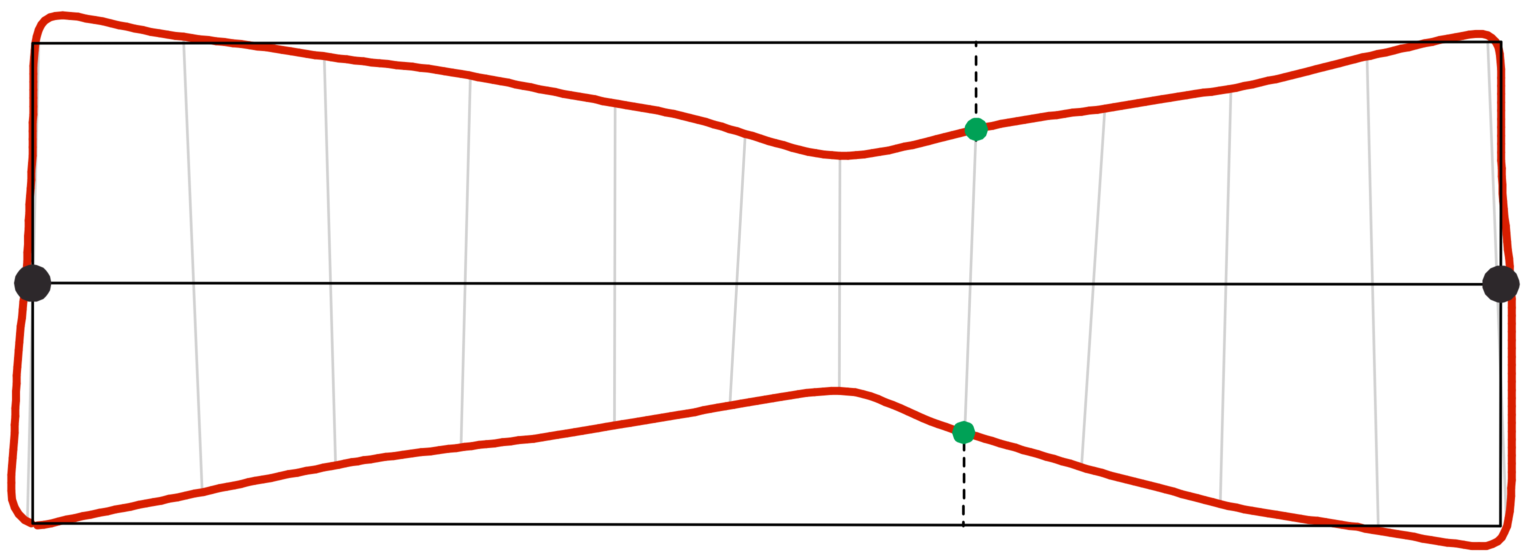
Cylindrical Douglas-Peucker error



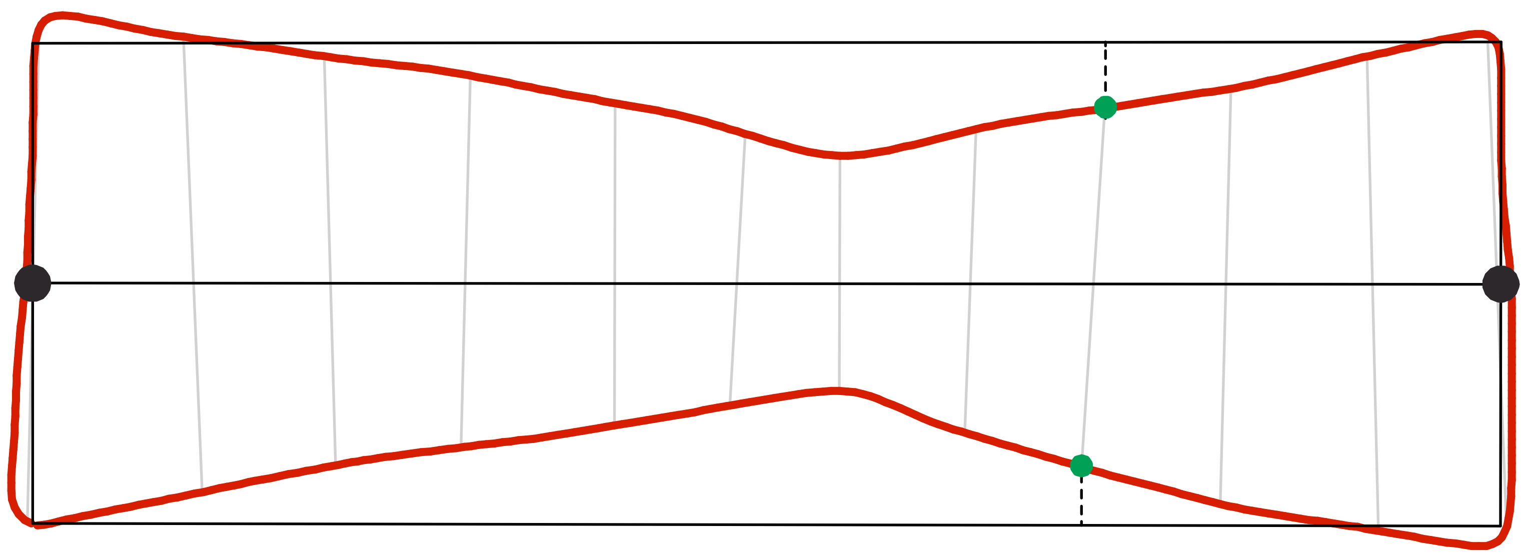
Cylindrical Douglas-Peucker error



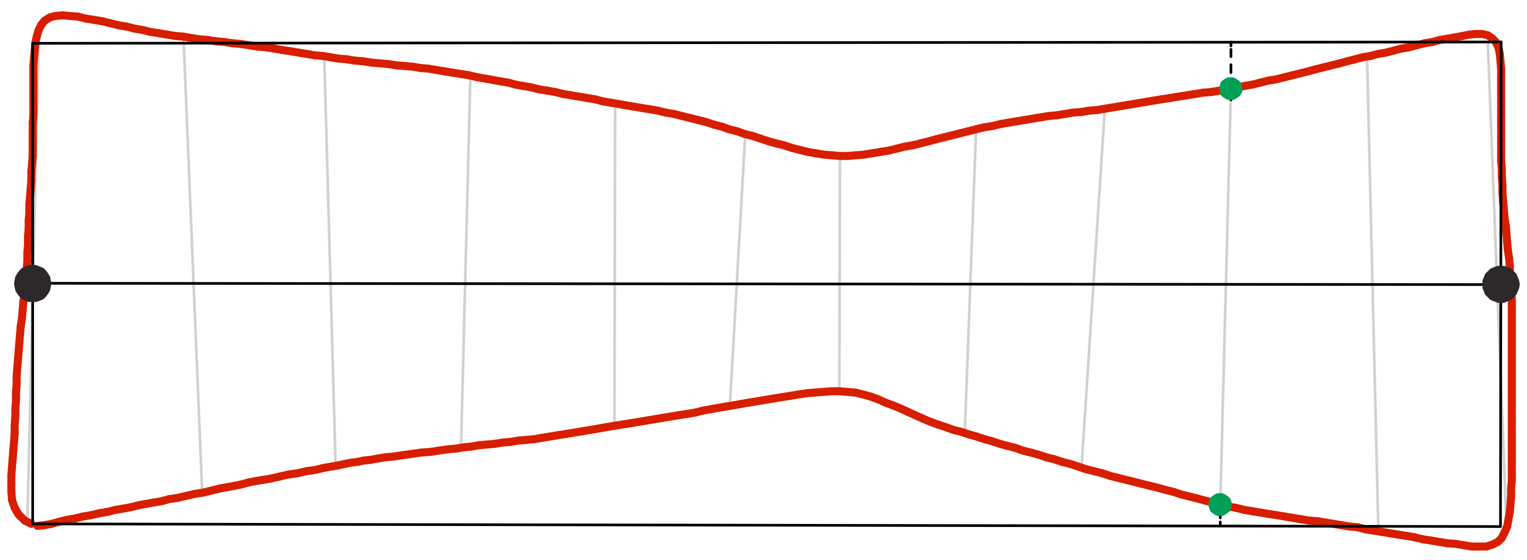
Cylindrical Douglas-Peucker error



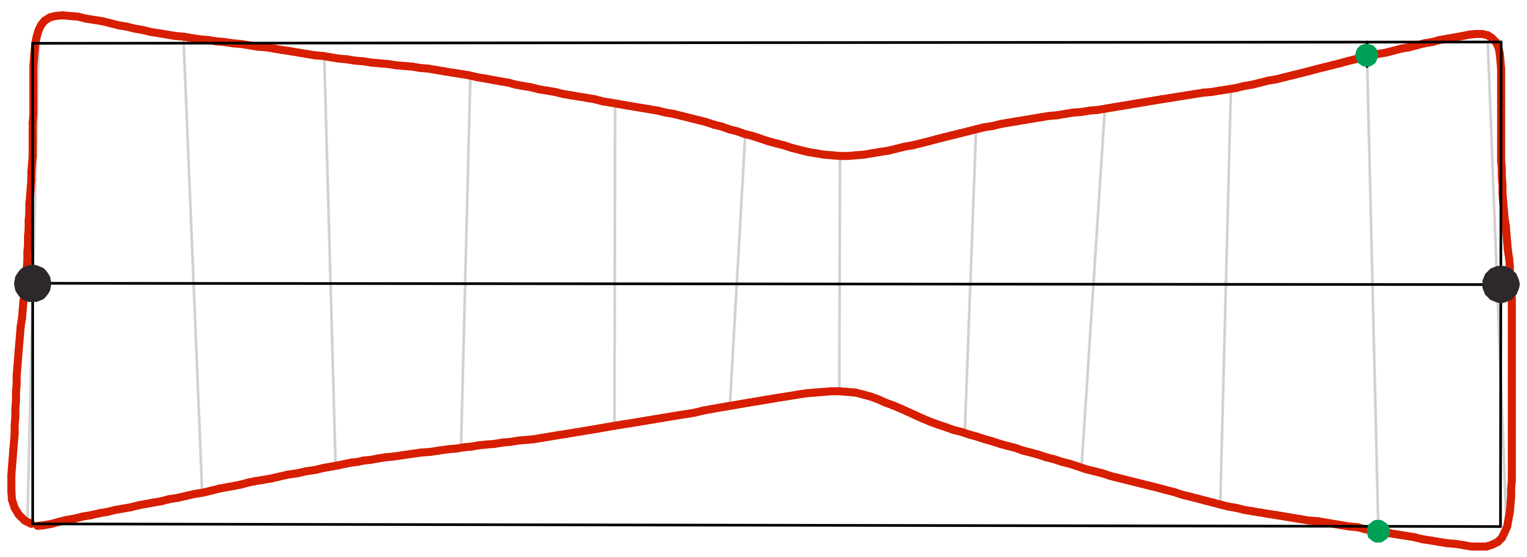
Cylindrical Douglas-Peucker error



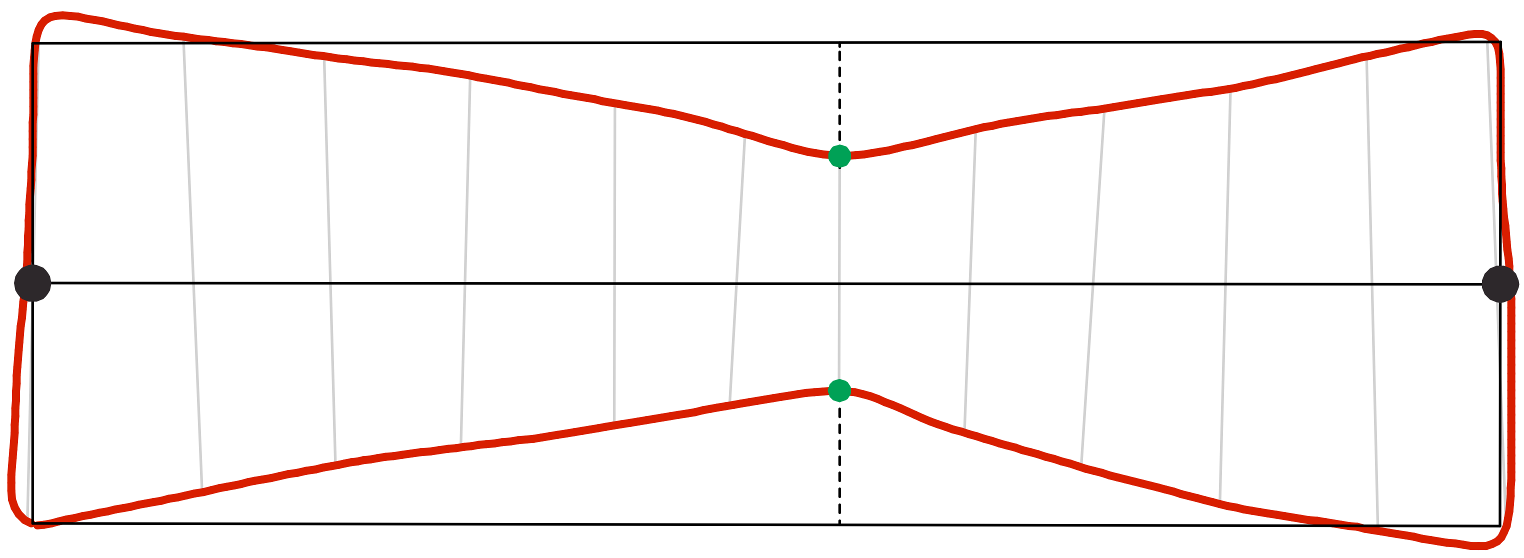
Cylindrical Douglas-Peucker error



Cylindrical Douglas-Peucker error

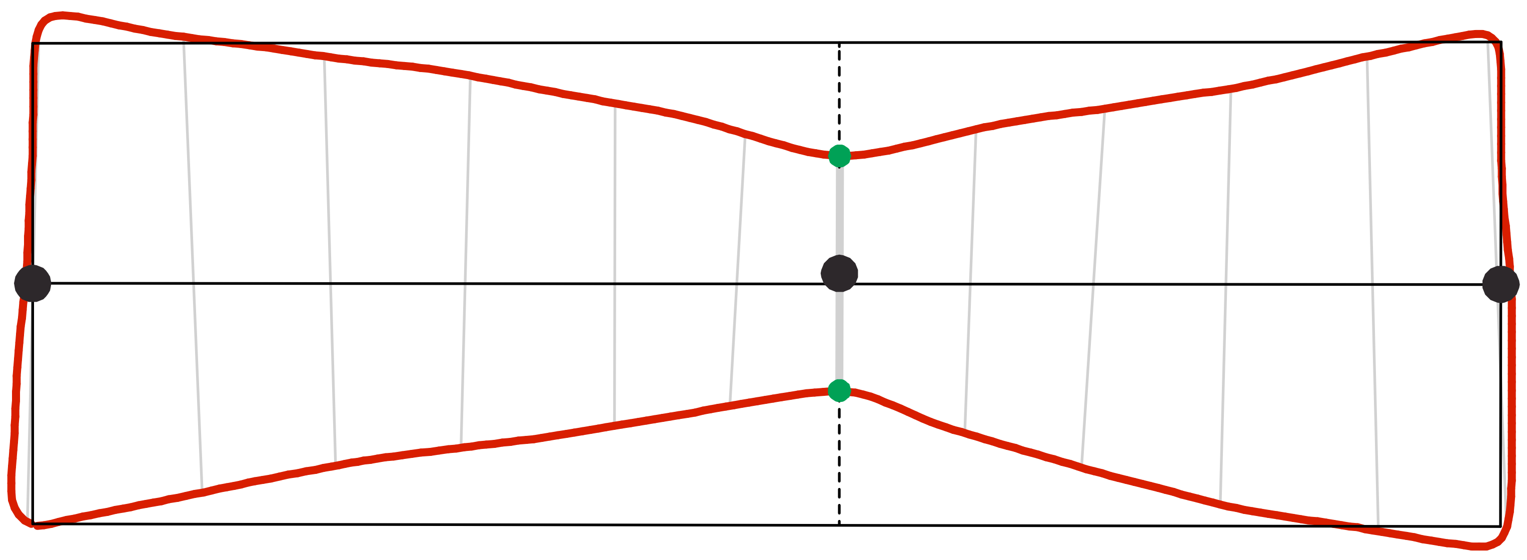


Cylindrical Douglas-Peucker error



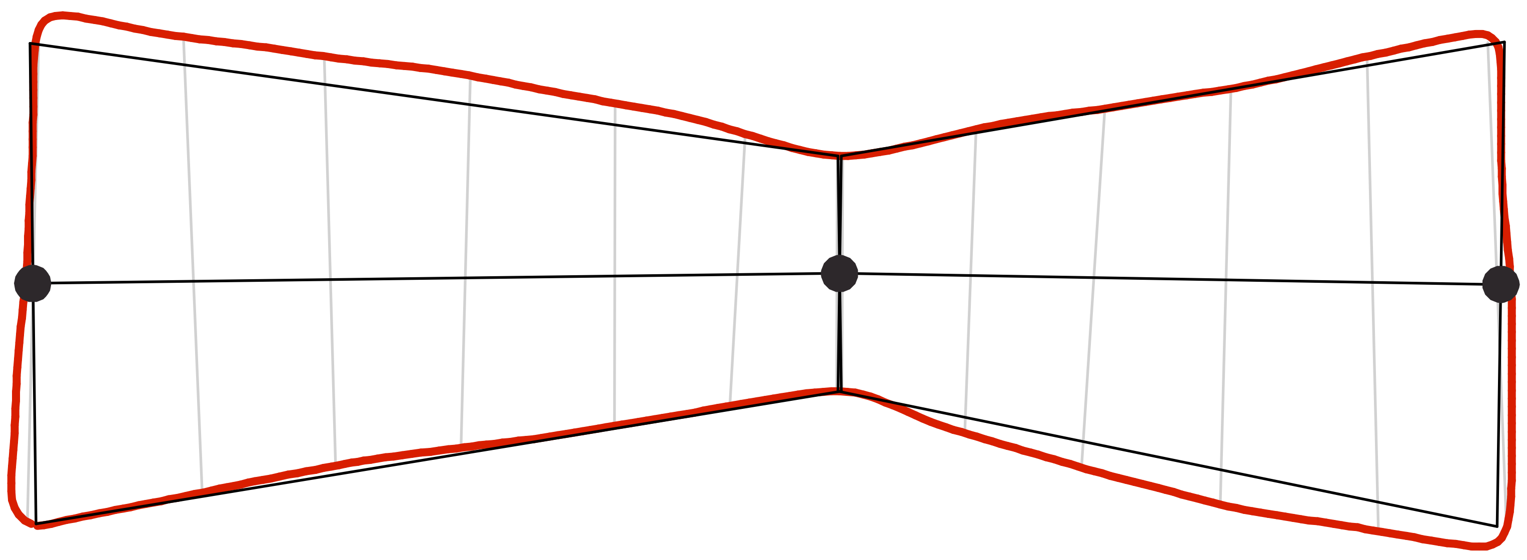
If the maximum is greater than the predefined threshold

Add new point



Then we take the midpoint of the chord, and recursively call for the parts before and after this chord

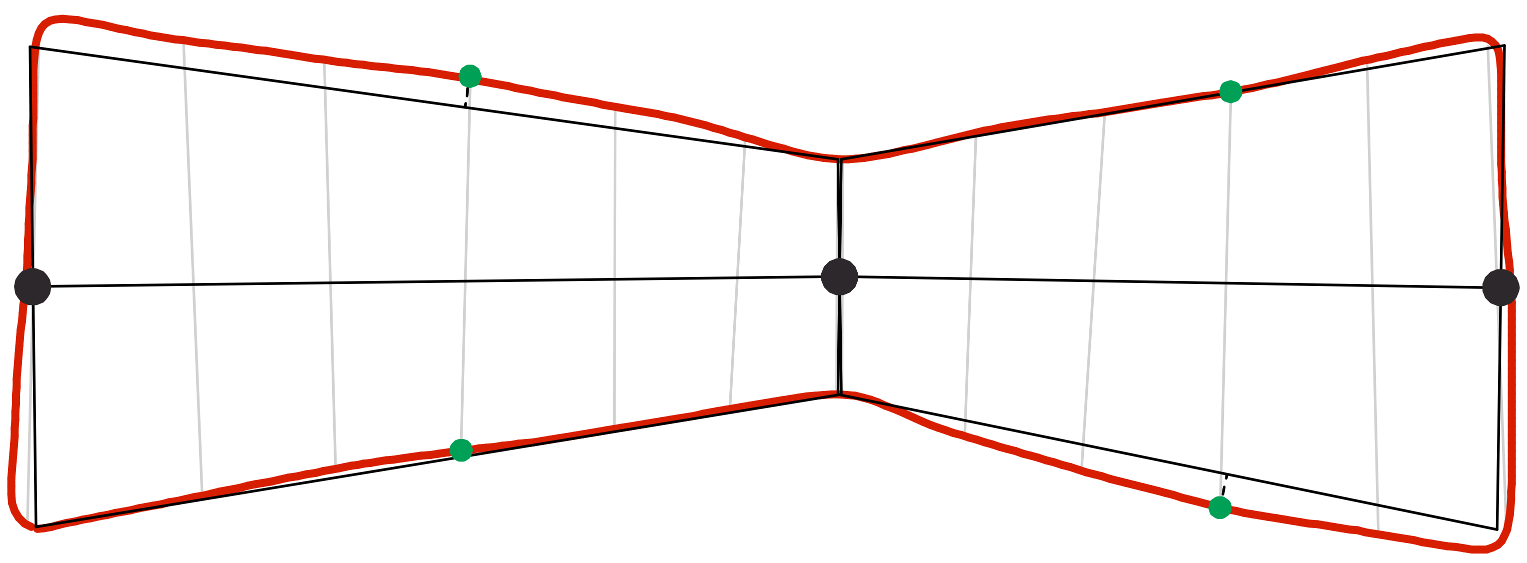
Fit trapezoids



Sponsored by ACM SIGGRAPH

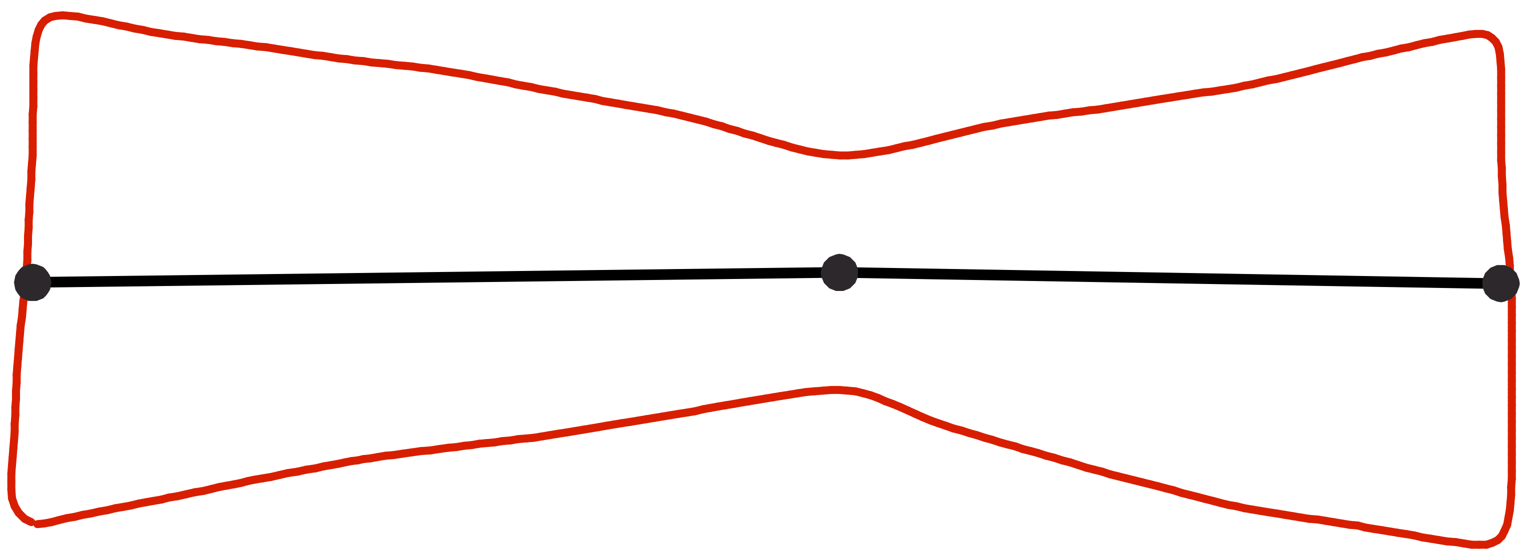
We fit trapezoids again

Error below threshold



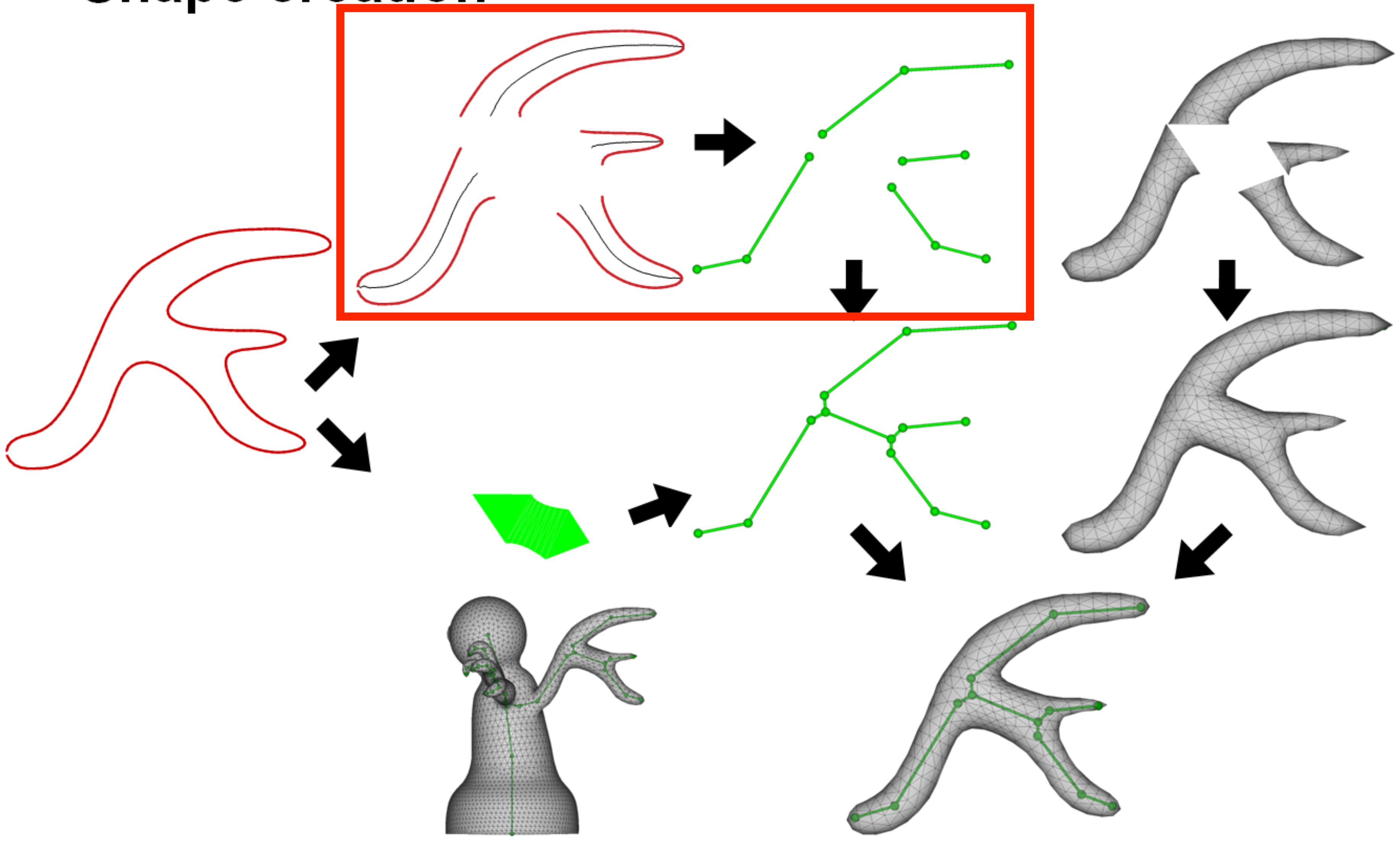
And if the errors are below the threshold, the algorithm terminates

Bones



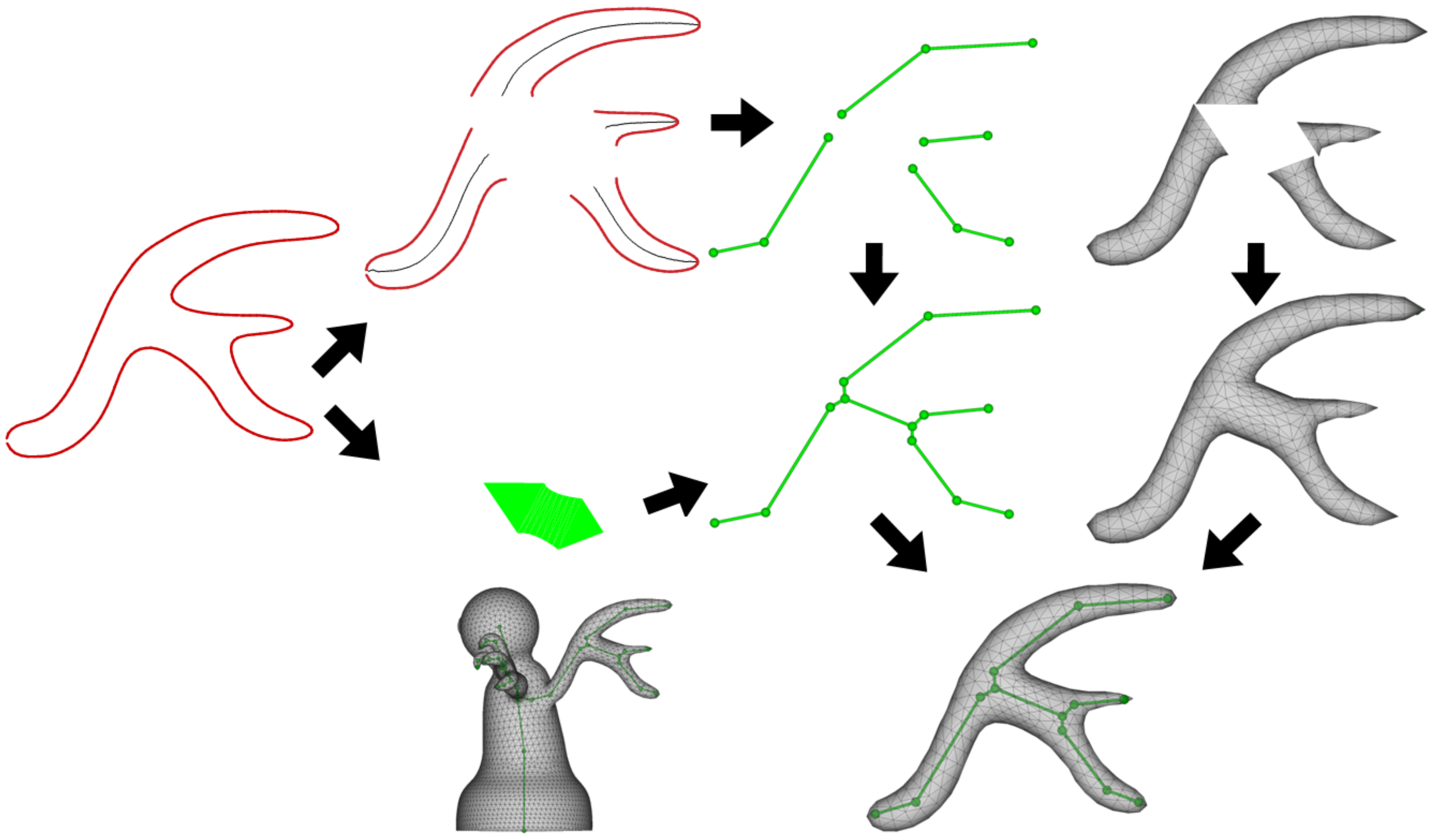
and the axes of the trapezoids are used as the bones of the skeleton

Shape creation



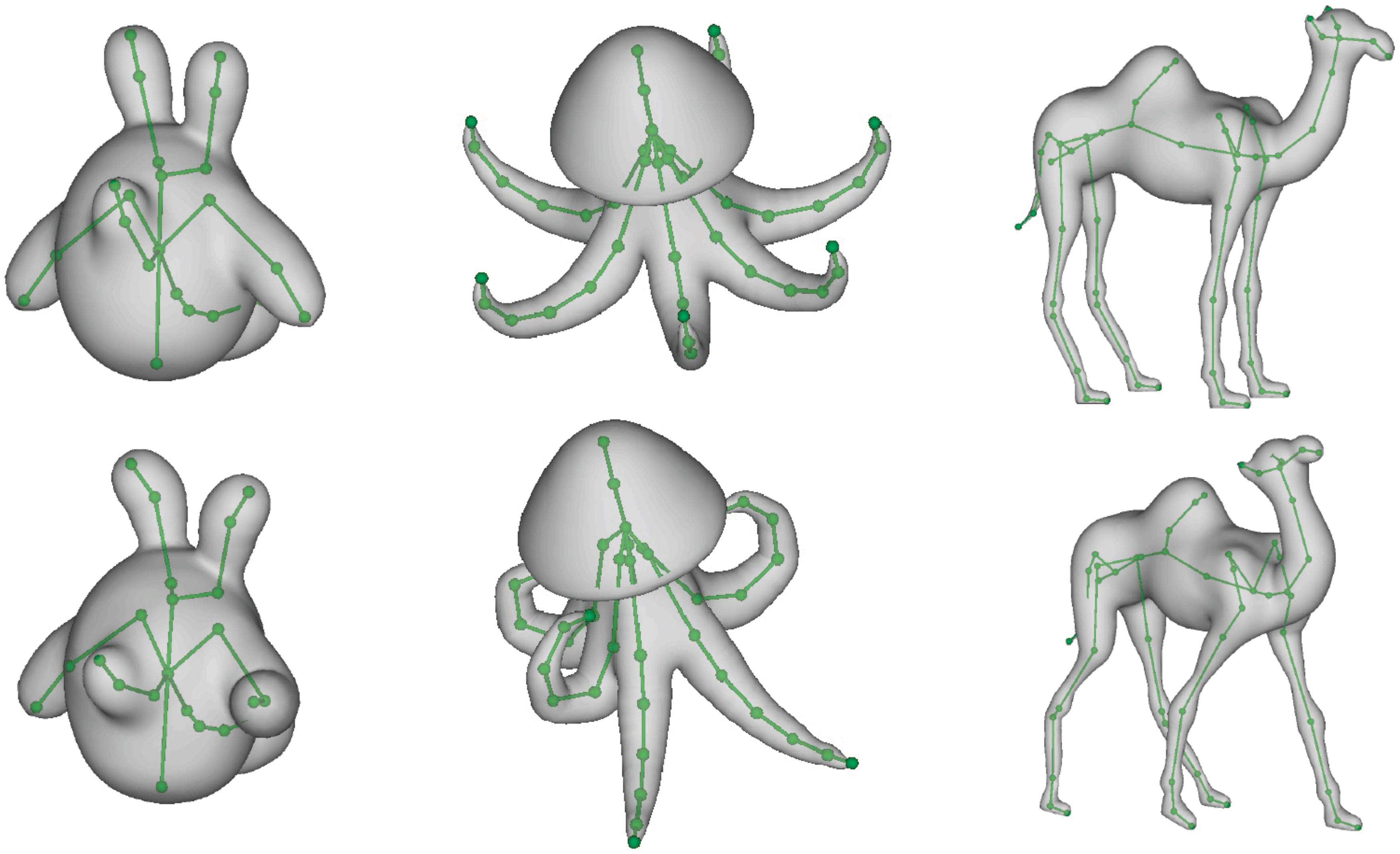
So using the CDP algorithm we create the skeleton for each cylindrical region. [c] For details on other steps, please see our paper.

Shape creation



So using the CDP algorithm we create the skeleton for each cylindrical region. [c] For details on other steps, please see our paper.

Results

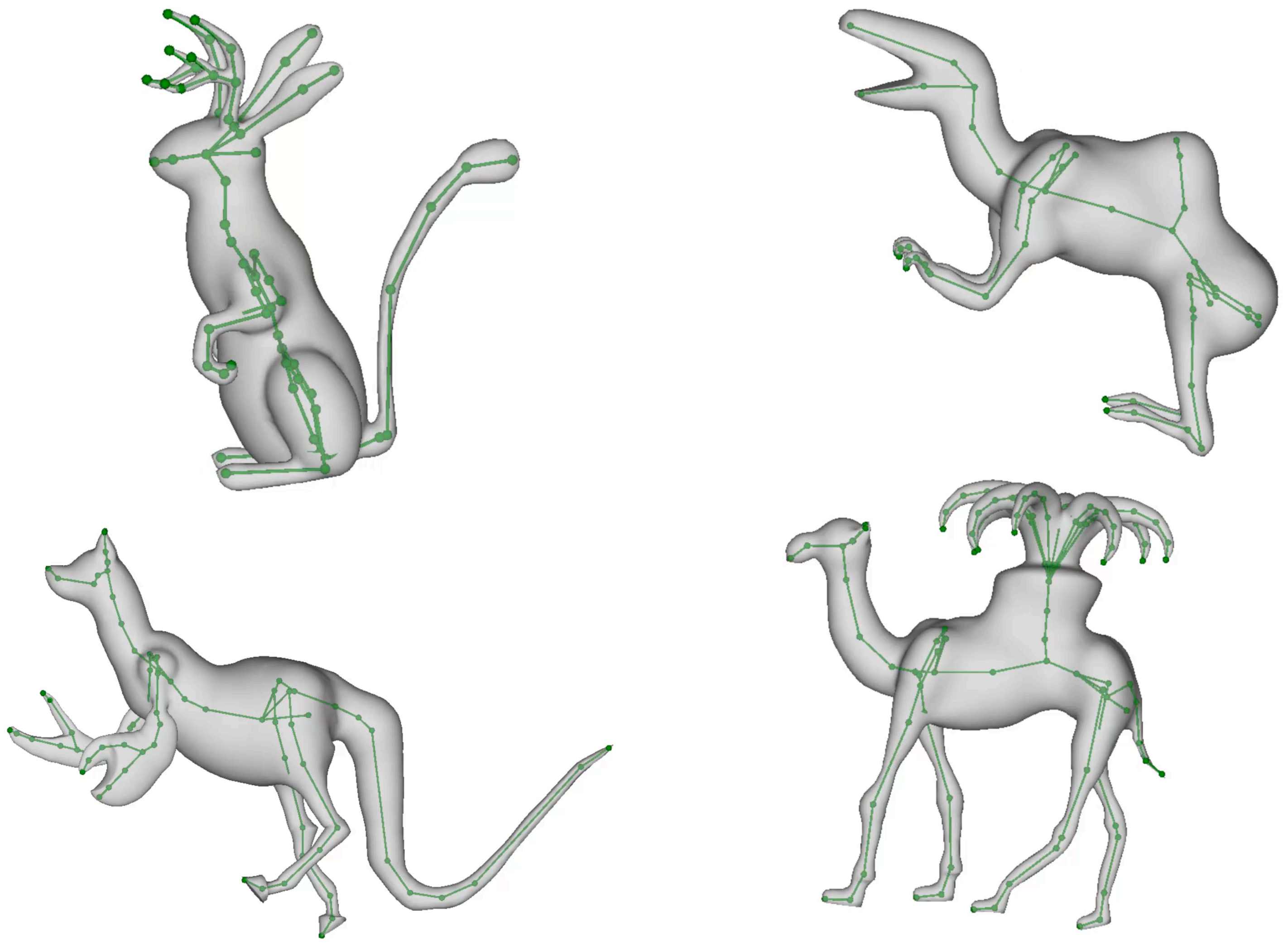


Sponsored by ACM SIGGRAPH

So this way users can easily create complex posed models.

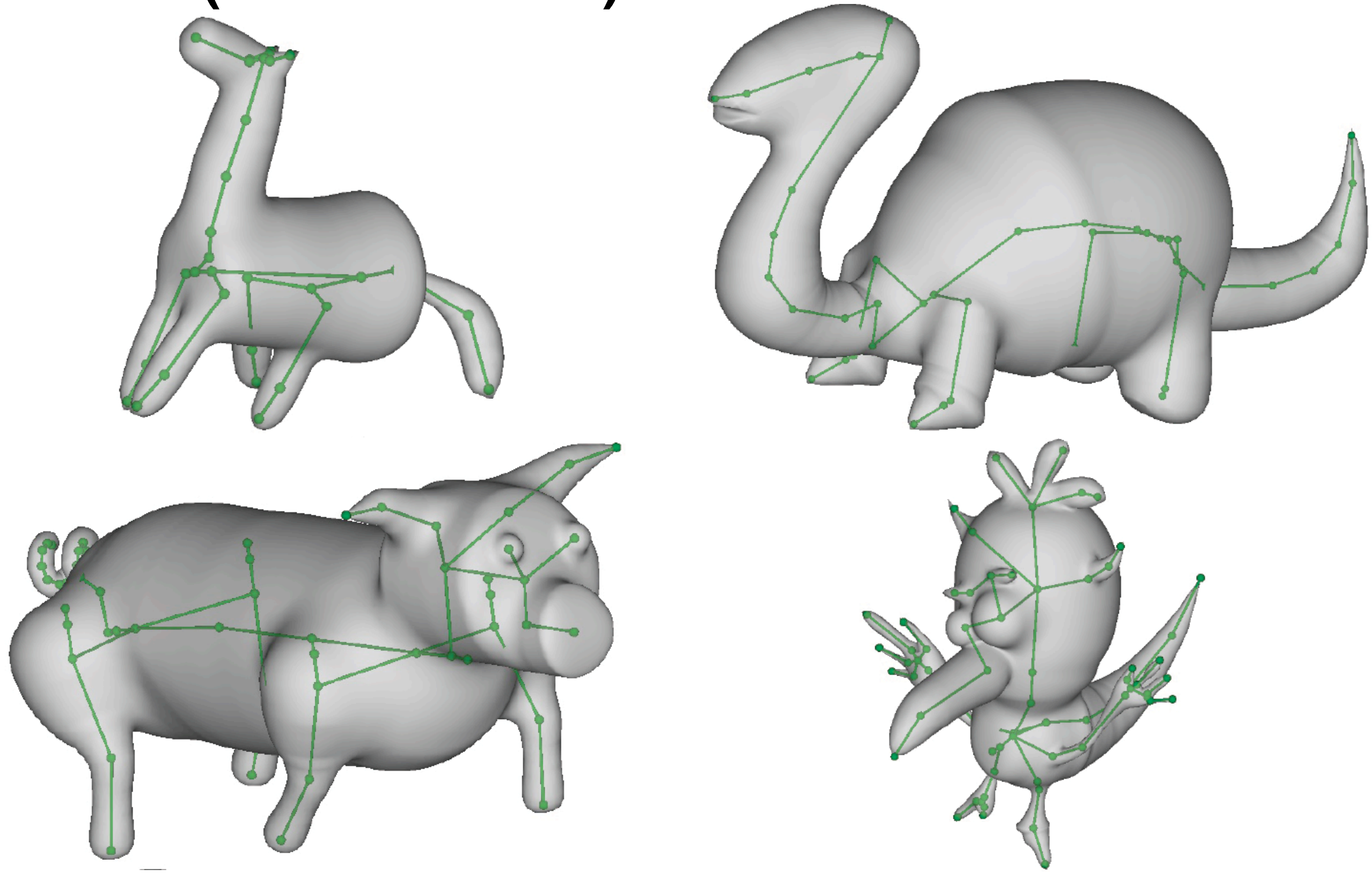
They don't have to worry about creating the rig, the skeleton and skin weights are constructed automatically with the shape, and maintained during the modeling process, so they can really concentrate on the design of the individual parts; which can be re-used like in the case of the squid example in the middle. Users can pose the shape while modeling, and thanks to the local re-skinning the modeling flow is never interrupted.

Results (reusing existing models)



An other advantage of modeling by parts is that users can combine existing models, they can quickly try out different ideas without any pausing due to rigging.

Results (1st time users)



Sponsored by ACM SIGGRAPH

Users don't have to be professional modelers to be able to create rigged shapes. The models you see here were created by first time users after a 15 minute training session.

Limitations

Sponsored by ACM SIGGRAPH

Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

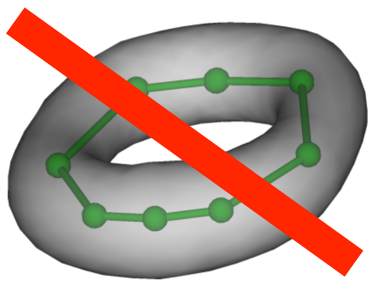
And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

We are in fact currently working on it.

This concludes are talk,

Limitations

Only acyclic skeletons



Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

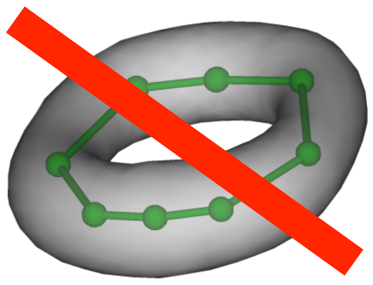
We are in fact currently working on it.

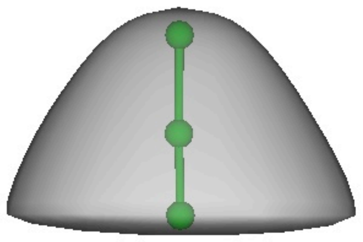
This concludes are talk,

Limitations

Only acyclic skeletons

Objects without strong, stable axial symmetry





Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

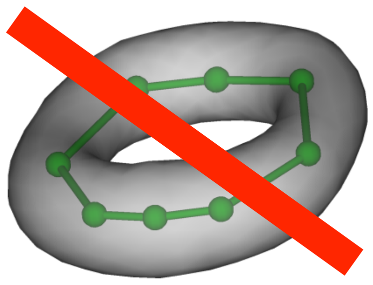
We are in fact currently working on it.

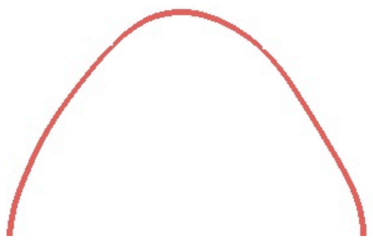
This concludes are talk,

Limitations

Only acyclic skeletons

Objects without strong, stable axial symmetry





Sponsored by ACM SIGGRAPH

Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

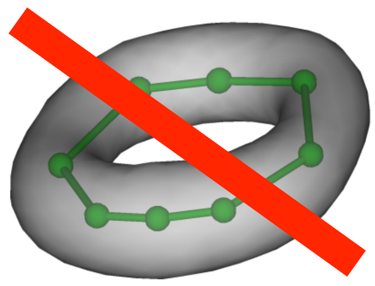
We are in fact currently working on it.

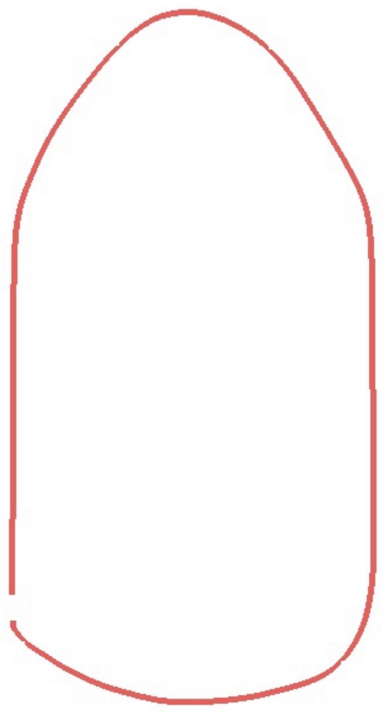
This concludes are talk,

Limitations

Only acyclic skeletons

Objects without strong, stable axial symmetry





Sponsored by ACM SIGGRAPH

Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

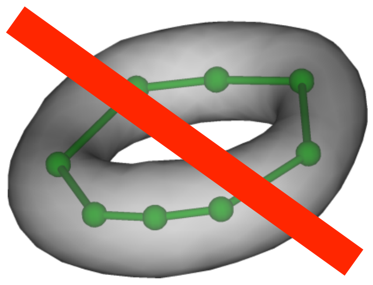
We are in fact currently working on it.

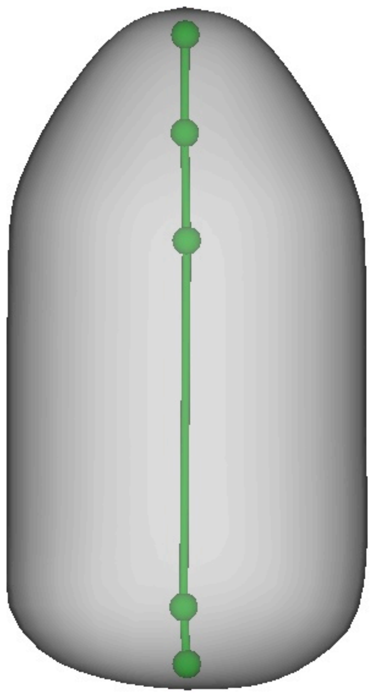
This concludes are talk,

Limitations

Only acyclic skeletons

Objects without strong, stable axial symmetry





Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

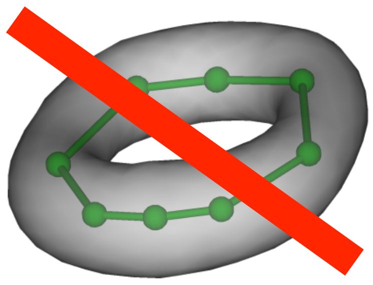
We are in fact currently working on it.

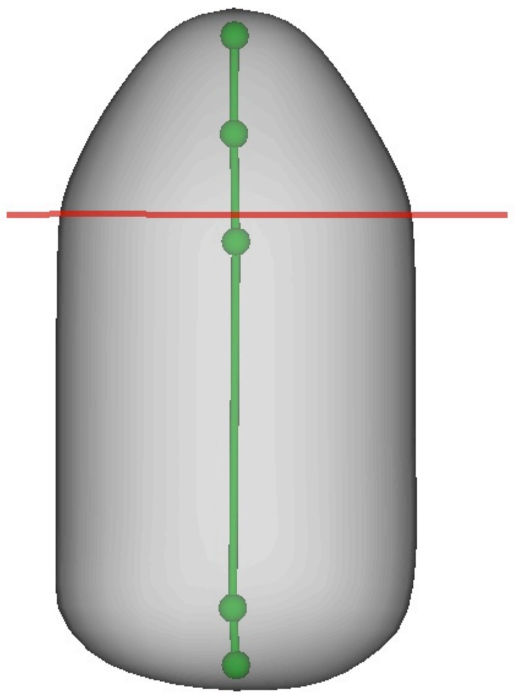
This concludes are talk,

Limitations

Only acyclic skeletons

Objects without strong, stable axial symmetry





Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

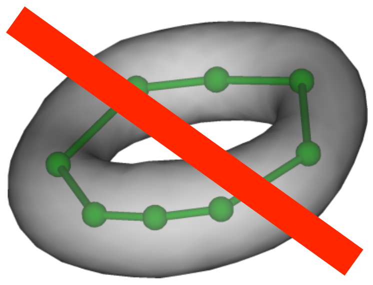
We are in fact currently working on it.

This concludes are talk,

Limitations

Only acyclic skeletons

Objects without strong, stable axial symmetry



Sponsored by ACM SIGGRAPH

Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

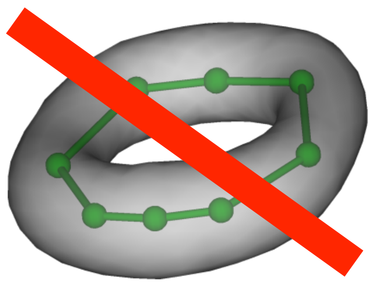
We are in fact currently working on it.

This concludes are talk,

Limitations & Future work

Only acyclic skeletons

Objects without strong, stable axial symmetry



Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

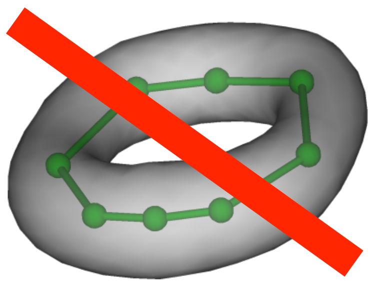
We are in fact currently working on it.

This concludes are talk,

Limitations & Future work

Only acyclic skeletons

Objects without strong, stable axial symmetry



Sponsored by ACM SIGGRAPH

Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

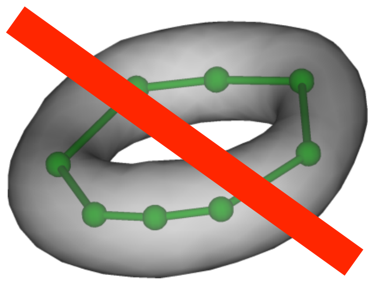
We are in fact currently working on it.

This concludes are talk,

Limitations & Future work

Only acyclic skeletons

Objects without strong, stable axial symmetry



Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

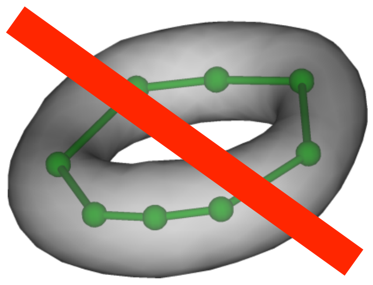
We are in fact currently working on it.

This concludes are talk,

Limitations & Future work

Only acyclic skeletons

Objects without strong, stable axial symmetry



Sponsored by ACM SIGGRAPH

Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

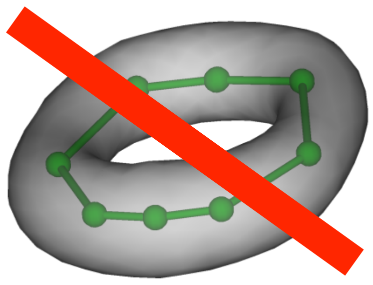
We are in fact currently working on it.

This concludes are talk,

Limitations & Future work

Only acyclic skeletons

Objects without strong, stable axial symmetry



Adapt for improved skinning methods

Sponsored by ACM SIGGRAPH

Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

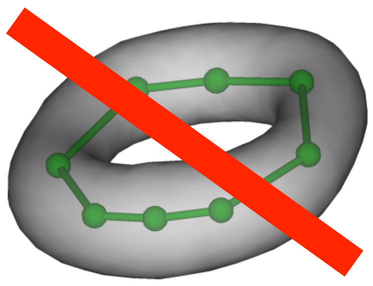
We are in fact currently working on it.

This concludes are talk,

Limitations & Future work

Only acyclic skeletons

Objects without strong, stable axial symmetry



Adapt for improved skinning methods

Include animation in the unified pipeline

Of course our system has limitations as well:

It can only handle acyclic skeletons, new shapes always have spherical genus, and are therefore created with tree like skeletons, and merging always occurs at a single joint.

Objects without strong, stable axial symmetry cannot be created with a single sketch. For example the head of the squid from the examples could not be created with a single sketch, but instead

we had to introduce a workaround: sketch a longer shape with a well defined symmetry axis and then cut off the extra material.

Avenues for future work include adapting our skin weight recomputation for more advanced skinning methods.

And an interesting yet non-trivial extension is including animation in the unified pipeline, such that changes to the model are instantaneously reflected in the animation.

We are in fact currently working on it.

This concludes are talk,

Thank you!

Thank you for your attention