

Thanks for introducing me, and thank you all for coming to my talk.

Today I would like to introduce ...

And here I would also like to thank all my collaborators' great help.

Introduction: 3

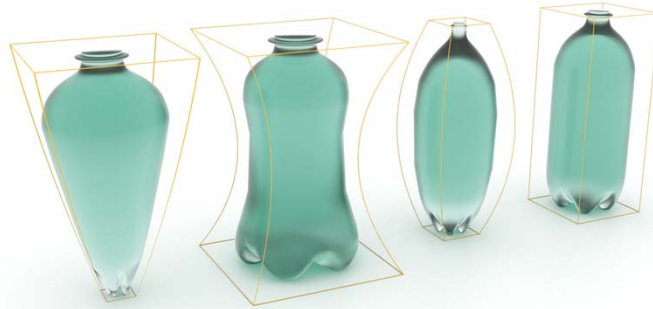
Sampling: 4

Basis: 2

Laplace: 4

Results: 3

Content creation



Maximize *variance*, while keeping *shape structure*.

Our work is **targeted** at the problem of content creation, which is an important step for providing **wide variety of artistic data**.

For creating new data, people normally start from a **exist template shape**. Such as **these bottles**, they look different, but all of the left three are **derived** from the right most **neutral** shape.

That relates to human's ability of **abstracting structures**, and infer new variance in the mean while.

Symmetric man-made world



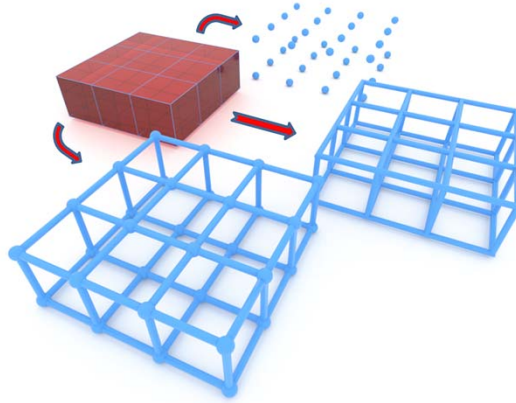
Symmetry: invariant to transformation.

Structural organization is a very natural way for people to understand the shape. So it is no coincidence that man-made shapes tend to have some prominent structure.

One **rigorous mathematical study** of shape structure is symmetry group, which in 3D space basically means invariance to 3D transformations.

We adopt this notion in our work, and use symmetries as our structure representation.

Symmetry-aware shape modeling

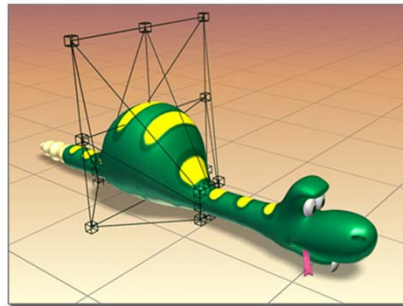


We need a modeling tool that understands the target shape.

So back to our content creation problem, we need a modeling tool that **can understand the symmetry structure of target shape.**

And make use of this important information.

Fast, and easy



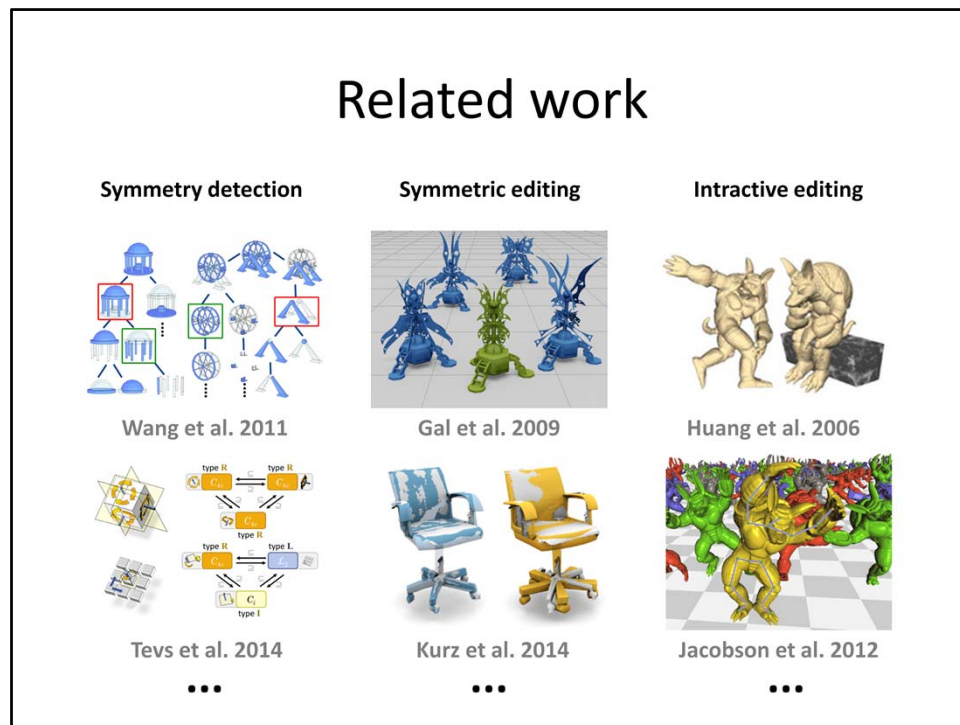
3ds Max tutorial

Minimal interaction – maximal intended effect.
But how to formulate *symmetric editing*?

Modern modelling software has greatly improved productivity. For example, in this FFD application, a selected segment can be easily manipulated by very few control points.

But for models with **complex symmetry structure**, how can we formulate symmetric editing?

Related work



Before going into the details of our construction, we would like to list some closely related work in recent years.

The first element is symmetry detection, which **provides the input to our pipeline**.

And then there are also some seminal works in the direction of editing while **keeping certain geometric constraints**.

The last is about speed, **especially those techniques for achieving interactive editing**.

Our goal is to **combine all these ingredients**, and propose a **generic editing framework**.

The first element is ..., which provides the input to our pipeline. Organized in a hierarchical way for better understanding very complex shape composition. Symmetry groups are studied in a very detailed classification philosophy. Our work uses the results provided by ...

Seminal work from ... uses a feature based description of shape structure, and can keep certain Euclidean invariance through optimization. Another recent work by ... Builds symmetric mapping from a template shape to the target scan data.

For achieving interactive editing, one line of research resort to sub-space method, which restrict the deformation on a pre-determined sub-set of variables, then propagates the motion to the entire mesh. For example, ... Builds a coarse control mesh around the original mesh, and ... Control points can be disconnected.

Our goal is to combine all these ingredients, and propose a generic editing framework.

IWIRES: preserving characteristic features and global structure. Bounded flat surface assumption, specific to constant curvature surfaces. Limited by the feature descriptions, example: twisted box, rotational symmetry, sample on the surface. Euclidean invariance instead of systematic symmetry structure analysis, so have to resort to greedy propagation algorithm to resolve conflictions.

Kurz: focused on matching, soft constraints.

Wang: symmetry detection and hierarchical organization.

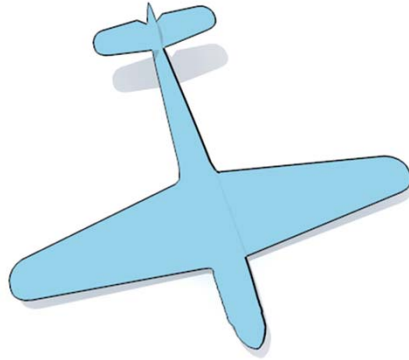
Bokeloh: translational symmetries, least-squares with user chosen constraints; dense bases produced by SVD.

Huang: builds a coarse control mesh around the original mesh.

Jacobson: solve the optimization problem on a low-frequency subspace and subsequently transfer the result to a high-resolution mesh.

Lipman: use spectral method to determine the space spanned by symmetry-invariant functions.

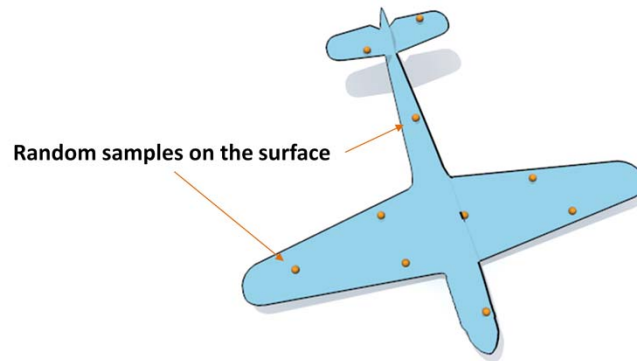
Subspace method



One of the most important technique for **reducing computation cost** is the subspace method.

Taking this airplane model for example,

Subspace method

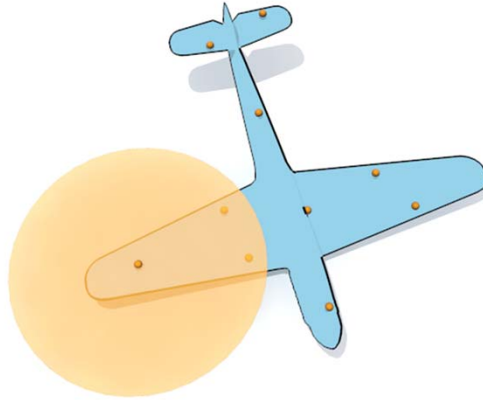


Subspace spanned by *basis constructed on samples*.

These methods construct basis functions on low frequency samples, and **restrict computations onto a subspace spanned by those basis**.

Here we show a simple random sampling on the surface,

Propagate computation

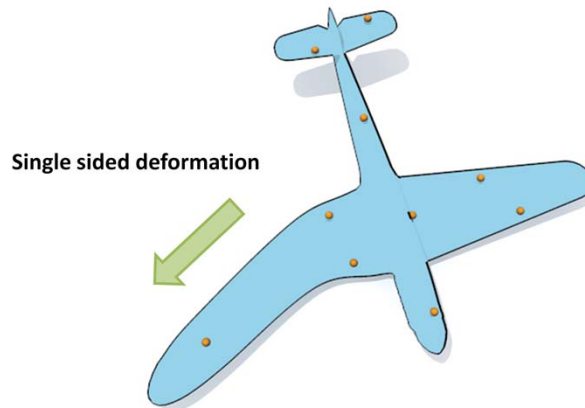


Each sample is associated with a *compact support* weighting function.

For evaluation on the original mesh, those methods just **propagate local computations through a convex combination**.

So each sample is associated with a compact support weighting function. we use Gaussian kernel in our implimentation.

Random sampling - problem



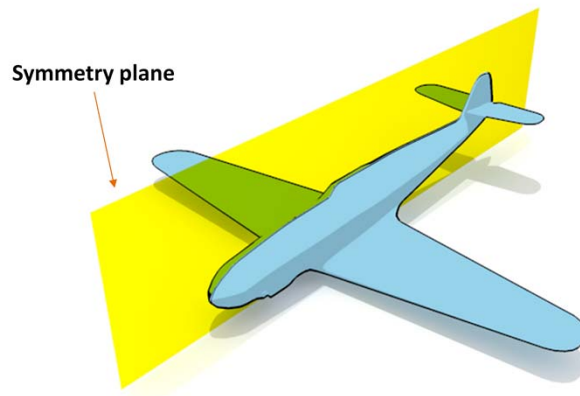
Sampling is not aware of *symmetry structure*.

The problem of this random sampling is, **for example**, if we want to **make the wing longer by dragging one sample** on one side, there will be only single sided deformation.

But this is clearly not what we expect: what we want is a symmetric deformation, which means both of the wings should look the same after the operations.

The reason is this sampling is not aware of symmetry structure, so there is **no correlations** between its symmetric parts.

Symmetry-preserving deformation



Forming **affine subspace** of all continuous spatial deformations.

By saying symmetry struction, while, **in this case** we mean the central symmetry plane, and both wings are reflective to each other.

Then how could we use this information to make sure these two parts move symmetrically?

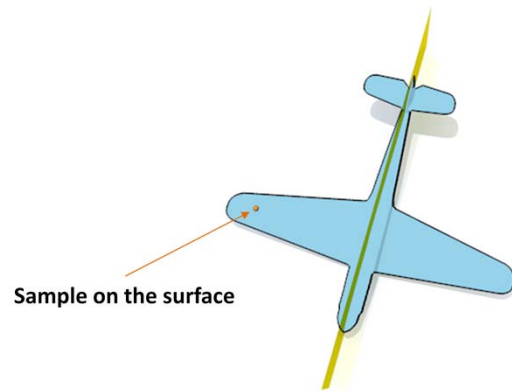
(click)

While, first of all, from theoretic aspect, in our paper we have prooved

Which means it is **possible to construct symmetric deformation subspace**, by putting basis functions on **symmetric samples**.

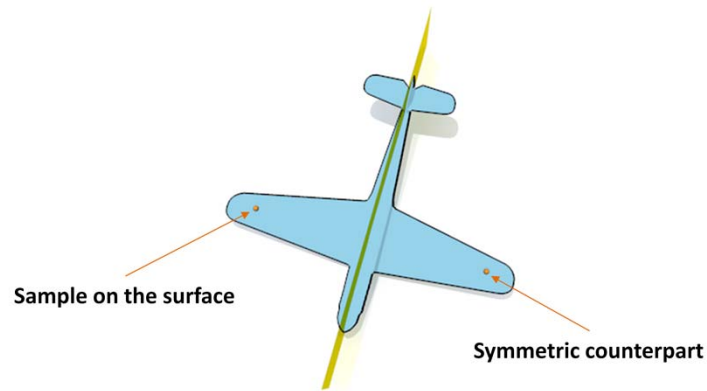
So there are two elements in our sampling pipeline: symmetric sampling and symmetric basis construction.

Symmetric sampling



We start with symmetric sampling: if we put a sample ...

Symmetric sampling

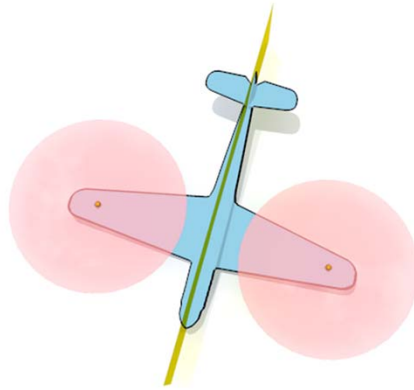


We consider *rigid discrete* transformations.

Then its symmetric counterpart is also added.

Here we consider rigid discrete transformations, and in this case it is the **reflection generated by this central symmetry plane**.

Invalidate neighbor candidates

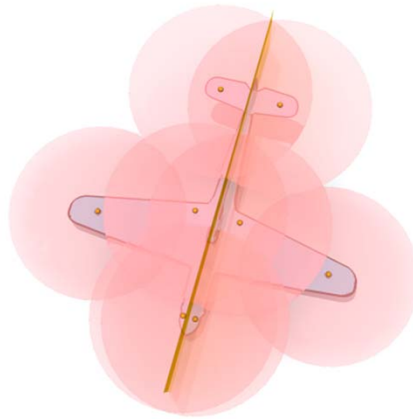


Density is controlled by a user provided *radius* parameter.

After a pair of samples are generated, we invalidate their **local neighbor candidates within a provided radius**.

In this way the **sampling density can be easily controlled**.

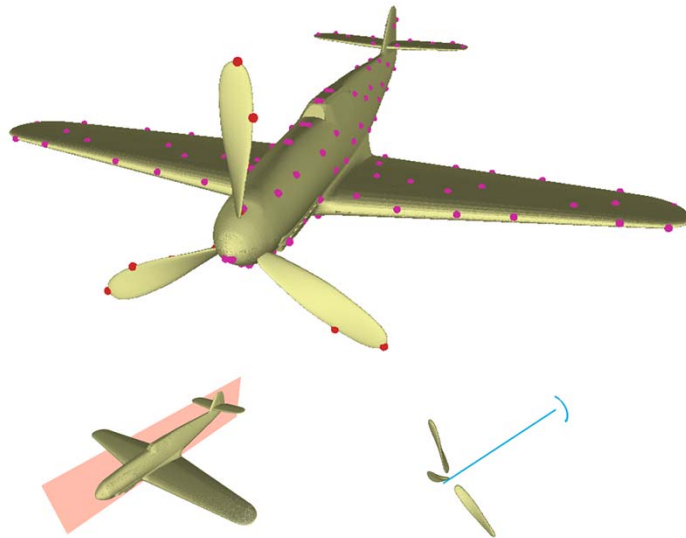
Repeat ...



Until the domain is fully covered.

This process is repeated until ...

Sampling example

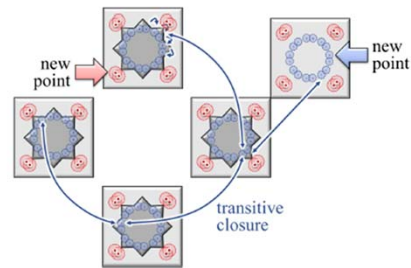
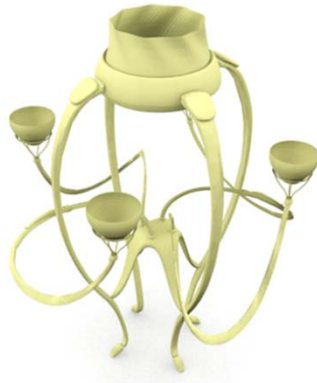


Here is a sampling result of the airplane model.

Which has **two separate symmetries**.

Of course this example is not hard to handle,

Hierarchical symmetric structure

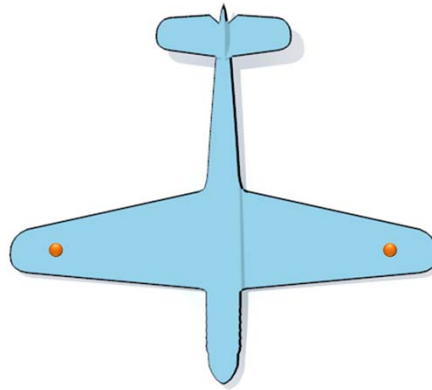


Connected through taking **transitive closure** of all transformations.

But our work can also handle the case of more complicated symmetry structure.

Details can be found in our paper.

Symmetric basis construction

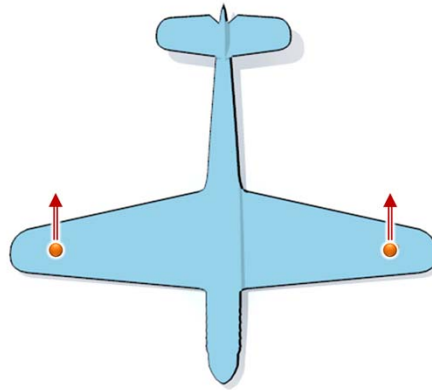


Now comes to our question: how to construct symmetric basis on those samples?

(click)

Let's start with one pair of symmetric samples,

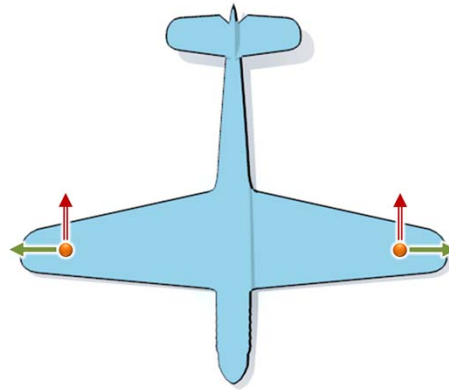
Symmetric basis construction



Fix one degree of freedom.

In our algorithm, first we Fix one degree of freedom, which means if move the left sample upwards, the right sample should also move upwards in the same amount.

Symmetric basis construction

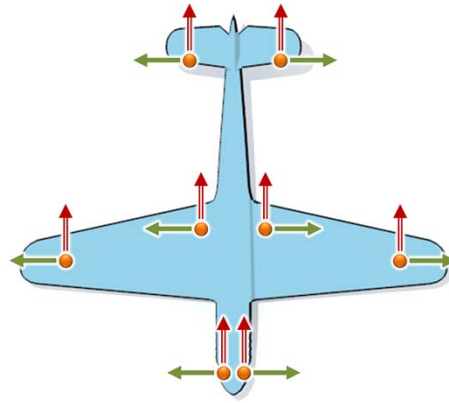


Symmetric local coordinate frame.

Then we add **all other orthogonal directions**, which **completes a local coordinate frame** for each sample.

Notice that these frames should be symmetric to the reflective plane.

Symmetric basis construction

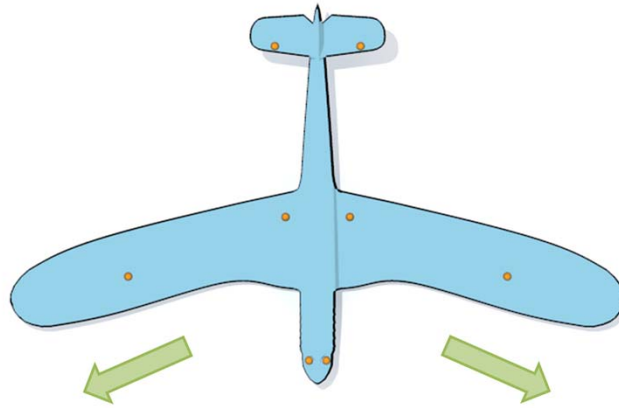


Same for all other samples.

We do the same for ...

This shows the complete set of our symmetric basis.

Symmetric movements



Dragging on one side

The other side move symmetrically

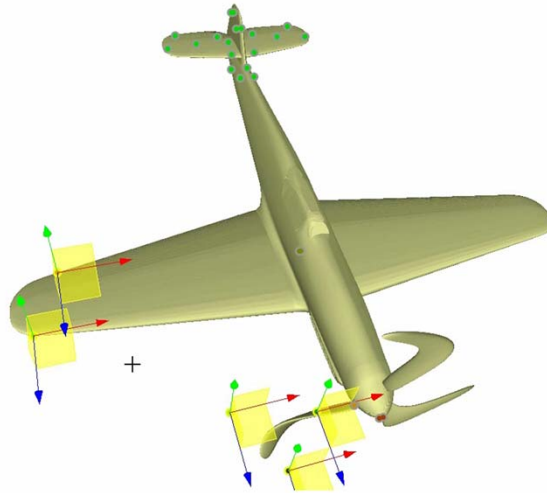
Symmetric editing on the samples is enforced inherently.

By this construction, we can see that **symmetric editing on the samples** is enforced inherently.

In this case, if we ...

Then ...

Example: symmetric editing



Here is an example, and you can see if we **only** edit one side, **the other side deforms accordingly**.

Laplace editing



Botsch and Sorkine2008

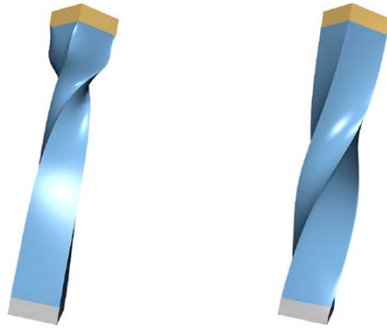
Fast and looking plausible.

Next I would like to **introduce the deformation part**.

For producing a **plausible** looking deformation, we adopt ... As the backend.

The benefit is **fast**, and **details of shape are maintained**.

Iterative rotation updating



Botsch and Sorkine2008

Natural looking comparable to non-linear method.

Also, for achieving **more natural looking**, we **iteratively update local rotations**.

Which looks comparable to expensive non-linear methods.

We re-estimate Laplace coordinates by iteratively (typically 5) applying a *rotation matrix* in the local frame of samples.

Laplace Editing

- Energy function:

$$E(\bar{u}) = E_L(\bar{u}) + \alpha E_C(\bar{u})$$

- E_L measures deviation from *Laplace coordinates*

$$E_L(\bar{u}) = \|L(\bar{x} + \bar{u}) - \delta\|^2$$

- E_C measures violation of *handle constraints*

$$E_C(\bar{u}) = \|A(\bar{u}) - a\|^2$$

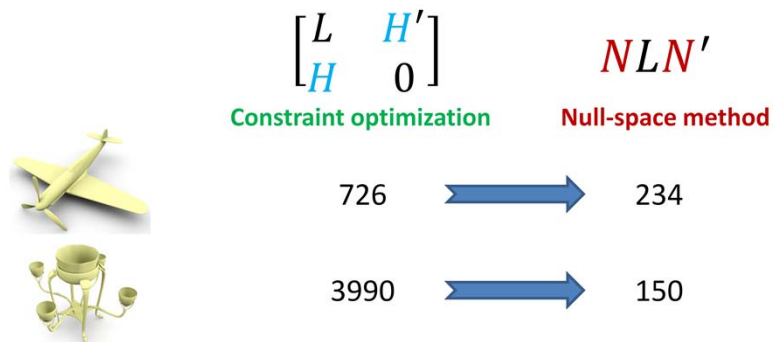
For laplace editin, basically we have this everygy function formulation,

Which has one part for measuring deviation from *Laplace coordinates*

And another part for measuring violation of *handle constraints*

Null-space projection

Apply *Lagrange multipliers* for **symmetry constraints** to give a linear system.
Project onto the **null-space** of **constraint matrix**.



Contrarily, more constraints will **reduce more degrees of freedom**, which means **less expensive computation**.

Then for symmetry constraints, we apply ... To ... **As in the ordinary constraint optimization process.**

One thing we want to mention, for faster computation, is the null-space method.

Which has the property that, if we **form another linear system** by projecting variables onto the ... Then these two systems **have the same optimal solution**. But the benefit is the **number of variables is greatly reduced**.

The problem of this formulation is, computation cost of the null-space is normally high. But in our paper we have proved the **symmetric basis constructed in our algorithm** span exactly the same null-space of constraints. So we can achieve this computation reducing **at no cost**.

(example click)

To show some concrete examples,

Here we measure the number of variables to solve. And show the difference by applying null-space projection.

Less than one third.

Another complex model with **more symmetry structure**, ... Reduced dramatically

NLN' – positive definite, L – full row rank

Null-space projection

Apply *Lagrange multipliers* for symmetry constraints to give a linear system.
Project variables onto the null-space of constraint matrix.

$$\begin{bmatrix} L & H' \\ H & 0 \end{bmatrix} \quad N L N'$$

Constraint optimization Null-space method

Then for symmetry constraints, we apply ... To ... **As in the ordinary constraint optimization process.**

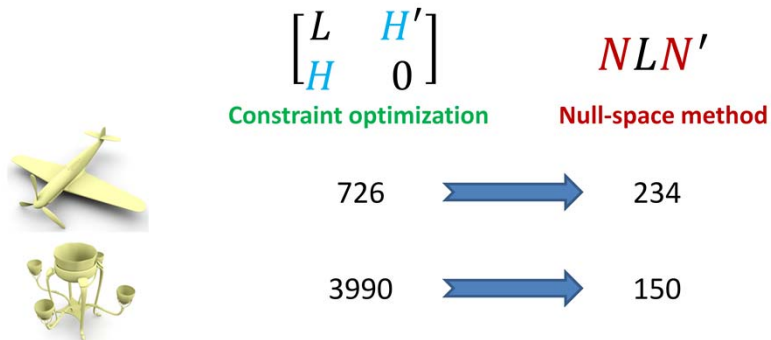
One thing we want to mention, for faster computation, is the null-space method.

Which has the property that, if we **form another linear system** by projecting variables onto the ... Then these two systems **have the same optimal solution**. But the benefit is the **number of variables is greatly reduced**.

The problem of this formulation is, computation cost of the null-space is normally high. But the **symmetric basis constructed in our algorithm** span exactly the same null-space of constraints. So we can achieve this computation reducing **at no cost**.

Null-space projection

Apply *Lagrange multipliers* for **symmetry constraints** to give a linear system.
Project variables onto the **null-space** of **constraint matrix**.



Contrarily, more constraints will **reduce more degrees of freedom**, which means **less expensive computation**.

To show some concrete examples,

Here we measure the number of variables to solve. And show the difference by applying null-space projection.

Less than one third.

Another complex model with **more symmetry structure**, ... Reduced dramatically

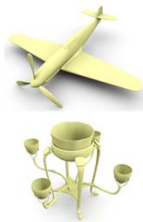
NLN' – positive definite, L – full row rank

Null-space projection

Apply *Lagrange multipliers* for **symmetry constraints**.
Project onto the **null-space** of **constraint matrix**.

Constraint optimization

Null-space method



726



234

3990



150

Contrarily, more constraints will **reduce more degrees of freedom**, which means **less expensive computation**.

Then for symmetry constraints, we apply ... To ...

One thing we want to mention, for faster computation, is the null-space method.

Which has the property that, if we **form another linear system** by projecting variables onto the ... Then these two systems **have the same optimal solution**. But the benefit is the **DoF is greatly reduced**.

The difficulty with this formulation is, null-space is normally hard to compute. But in our paper we have proved the **symmetric basis constructed in our algorithm** span exactly the null-space of constraints. So we can achieve this computation reduction **at no cost**.

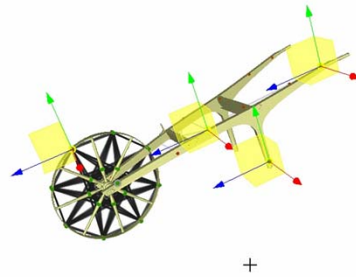
The number of variables to solve.

Less than one third.

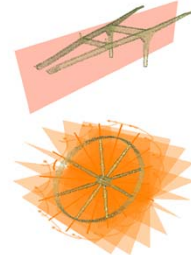
Another complex model with **more symmetry structure**, ...

NLN' – positive definite, L – full row rank

Result - I



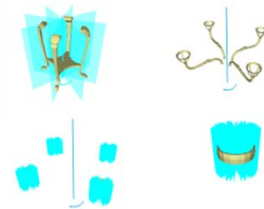
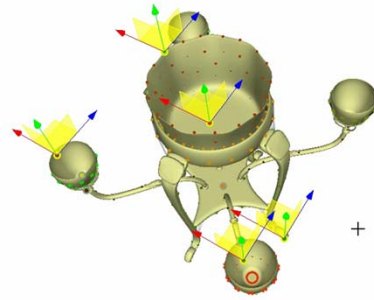
+



Triangles	97k
Symmetries	2
Samples	78
DoF	27
Frame rate	33
Prefactor	55 (ms)
Solving /5 Iter.	3 (ms)

We start with an example of less complexity.

Result - II



Triangles	165k
Symmetries	9
Samples	690
DoF	150
Frame rate	15
Prefactor	1300 (ms)
Solving /5 Iter.	21 (ms)

Much more complex symmetry structure, including this hierarchical symmetry links the four bowls.

You can see all the detail editing is very easy through our method.

The handles are put at user specified locations.

More examples could be found in our paper, and result video.

Comparison

Template



Target Scan



[KWW*14]

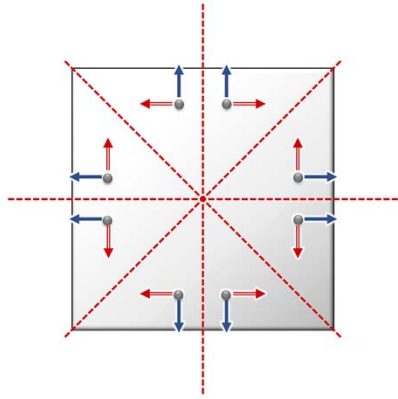


Ours



Thank you!

Hard constraints formulation

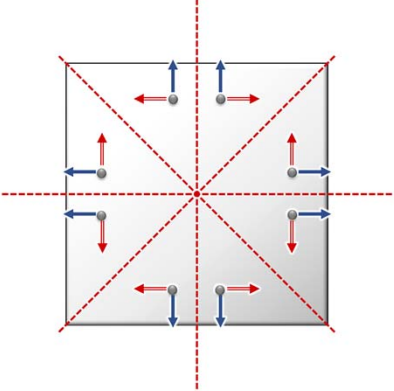


$$\begin{bmatrix} L & H' \\ H & 0 \end{bmatrix} \begin{pmatrix} \vec{P} \\ 0 \end{pmatrix}$$

$$H := \{\bar{p} = Tp\}$$

Constraint optimization, with **redundant** (e.g. transitive) constraint set.
Require expensive operations for removing redundancy.

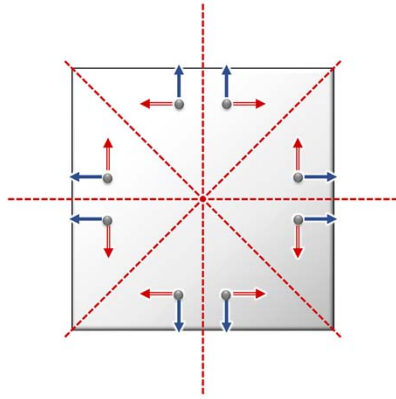
Simplest form



$$\begin{bmatrix} I_3 & & & \\ & T_2 & & \\ & & T_3 & \\ & & & T_4 \\ & & & & \ddots \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ \vdots \end{pmatrix} = \begin{pmatrix} \bar{p}_1 \\ \bar{p}_2 \\ \bar{p}_3 \\ \bar{p}_4 \\ \vdots \end{pmatrix}$$

Constraints are specified relative to a *reference element* in the orbit.

Direct symmetric basis construction



$$\bar{p} = Tp = Rp + t$$

$$\Rightarrow \bar{\delta} = R\delta$$

$$N = [I_3 \quad R_2 \quad R_3 \quad R_4 \quad \cdots]$$

When displacing one vertex, all other vertices on the same symmetry orbit have respective **rotated motion**.

Null-space projection

The directly constructed basis set spans the **null-space** of constraint matrix.

$$\begin{bmatrix} L & H' \\ H & 0 \end{bmatrix}$$

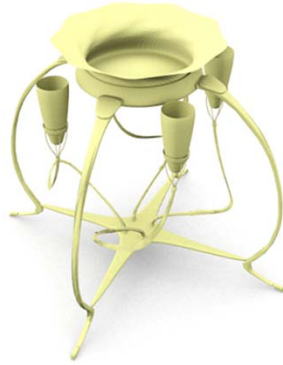
Dimension $\gg \dim(L)$

$$NLN'$$

Dimension $\ll \dim(L)$

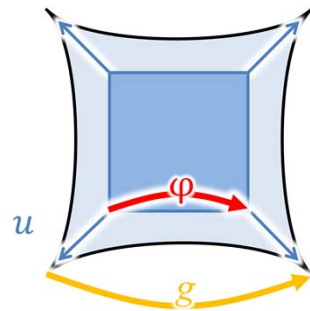
Contrarily, more constraints will **reduce more degrees of freedom**, thus less expensive computation.

Future work



Interactive updating symmetries: “pivoted” pre-factorization.

Symmetry-preserving deformation



$$u \circ \varphi = g \circ u$$

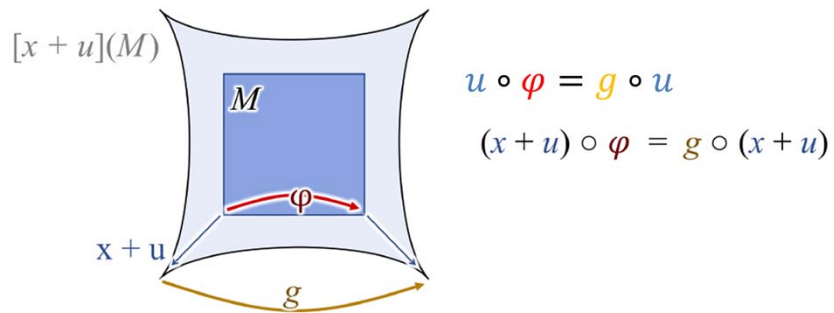
The set of all deformations that preserve the symmetries of a shape forms an **affine subspace** of the set of all continuous spatial deformations.

Rigid discrete transformations

In our work, we introduced the ..., which act simultaneously on some symmetric control points on the surface.

This is a very intuitive natural ...

Symmetry-preserving subspace basis



$$u \circ \varphi = g \circ u$$

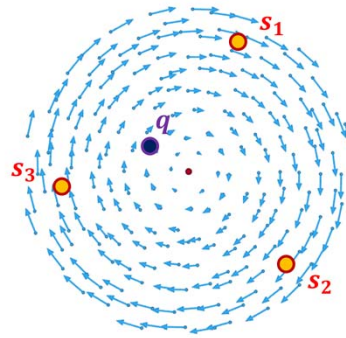
$$(x + u) \circ \varphi = g \circ (x + u)$$

The set of all deformations that preserve the symmetries of a shape forms an **affine subspace** of the set of all continuous spatial deformations.
 \Rightarrow we solve for *displacement vector* on each sample point.

Subspace method

- Describing the field by simple linearity
 - Discretize by “ankor points”
 - Map to arbitrary points

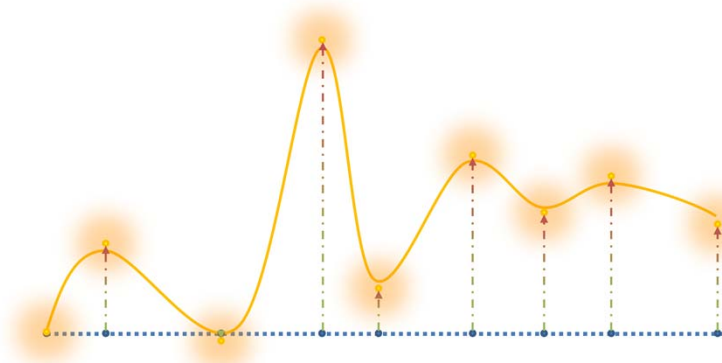
$$(s_1, s_2, s_3) \mapsto q$$



Subspace method

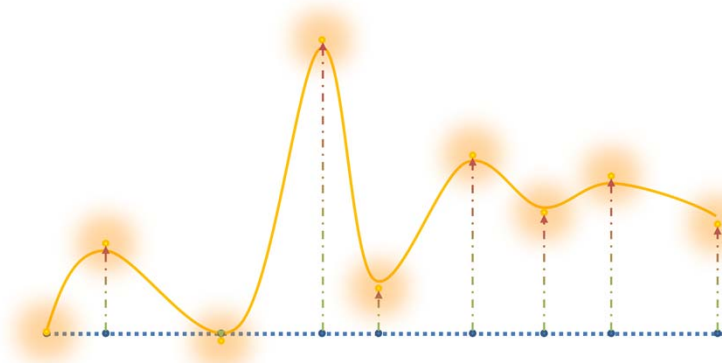
- Project deformation onto *symmetric subspace*
 - Sample symmetric “*basis points*” on surface
 - Basis are constructed on those points
 - Enforce *symmetric motion*
 - *Linear mapping* to arbitrary points

Lifting the displacements



Mesh vertices are lifted from their *compact supported* sample points.

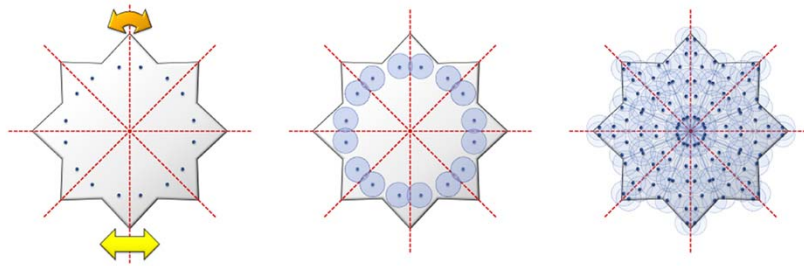
Lifting the displacements



Mesh vertices are lifted from their *compact supported* sample points.

Most expensive computation in our pipeline, gpu accelarated.

Symmetric sampling



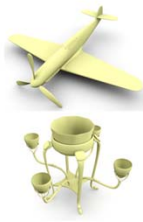
(a) random sample (b) excluded discs (c) final sampling

Density is controlled by a user provided *radius* parameter.

Null-space projection

Apply *Lagrange multipliers* for *symmetry constraints* to give a linear system.
Project onto the **null-space** of *constraint matrix*.

Null-space method $\begin{bmatrix} L & H' \\ H & 0 \end{bmatrix} \longleftrightarrow NLN'$

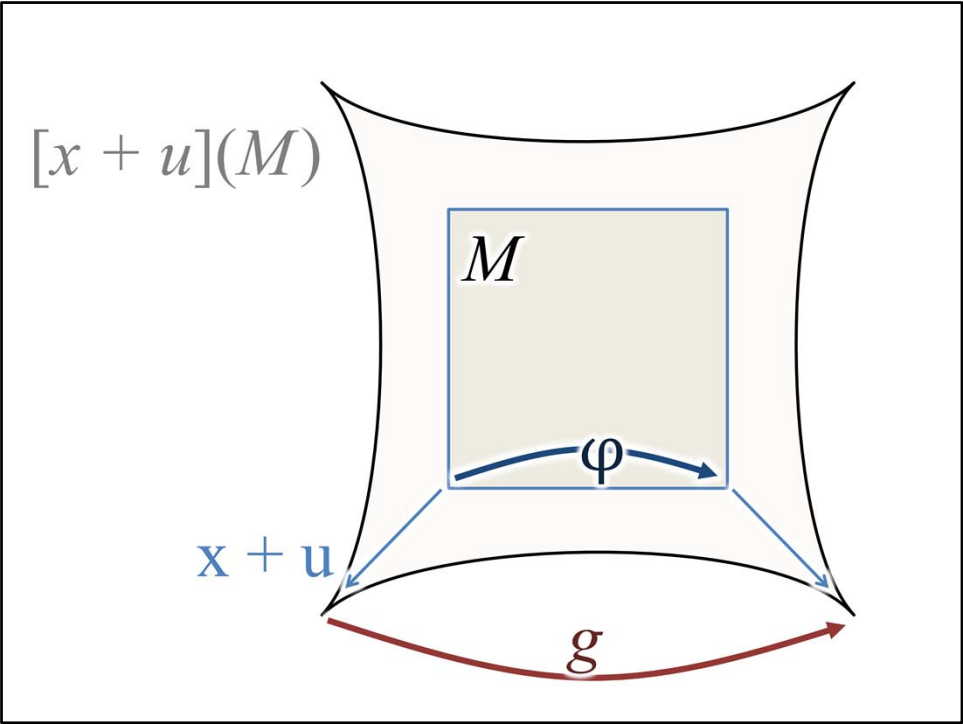


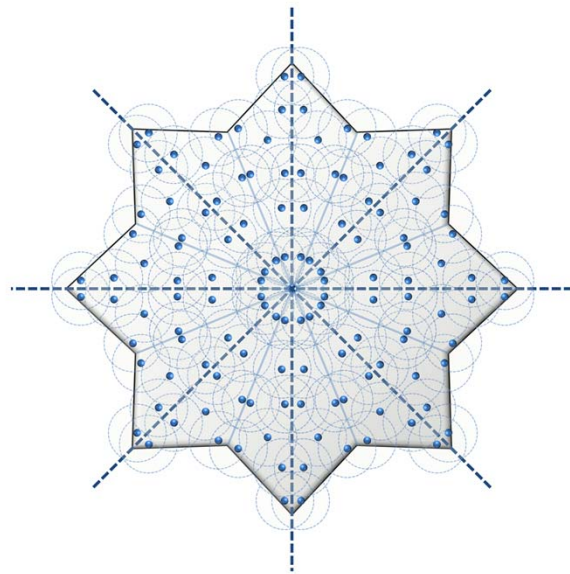
726 \longrightarrow 234

3990 \longrightarrow 150

Contrarily, more constraints will **reduce more degrees of freedom**, thus less expensive computation.

NLN' – positive definite, L – full row rank





sampling result

