Sketch-based Template Creation for Early Automotive Styling

Levent Burak Kara & Kenji Shimada

Carnegie Mellon University

December 2007

Introduction

- Support early automotive styling
  - Designers sketch early in the design
  - Cannot realize the concept in 3D
- Create 3D shape from 2D sketches
  - Rapidly convert sketches into 3D shape
  - No need for advanced modeling skills
- Approach
  - User marks points on a sketch
  - Modify a 3D template to match the sketch
  - 3D Draw on the new template

Major Advances since July 2007

- ✓ Renewed car templates
- ✓ Improved camera calibration algorithm
- ✓ Optimization-based template deformation
- ✓ Edge design using pen strokes
- ✓ Simplified vertex/tangent manipulation
- ✓ Post-Dimensioning
- ✓ Curve creation and styling on template
- ✓ Case examples for sedan/hatchback/minivan
- ✓ Paper submitted to CAD 08 conference

System Overview

User marks fiducial points

↓

Template alignment

↓

Template Deformation

↓

Impose continuity and shape constraints

↓

Edge manipulation

↓

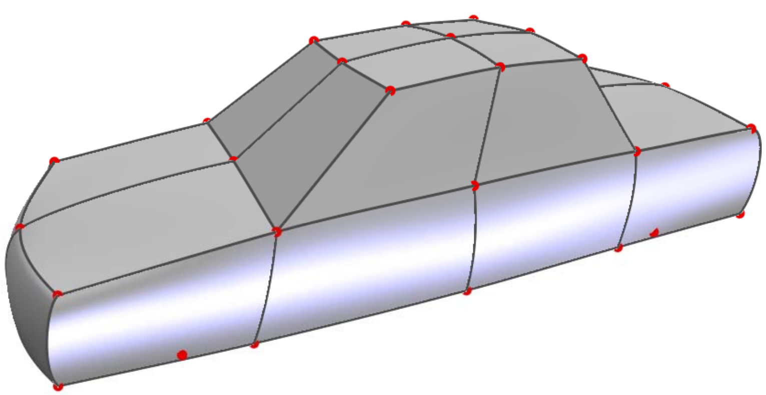
Dimensioning

↓

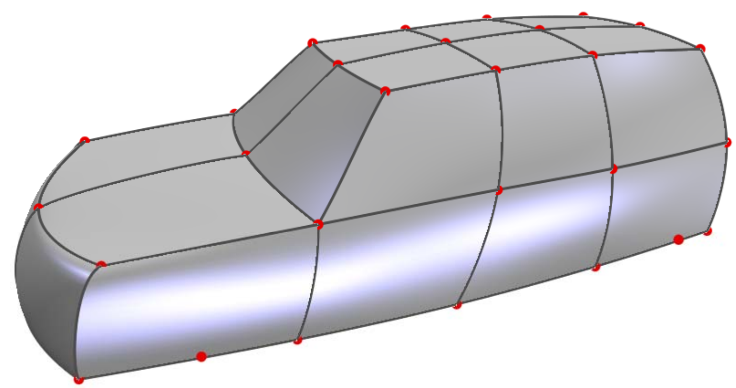
Styling

Automotive Templates

- Generic car shapes without details
  - Fiducial nodes, including wheel centers (red)
  - Cubic edges (black)
  - Bi-cubic surface patches (gray)



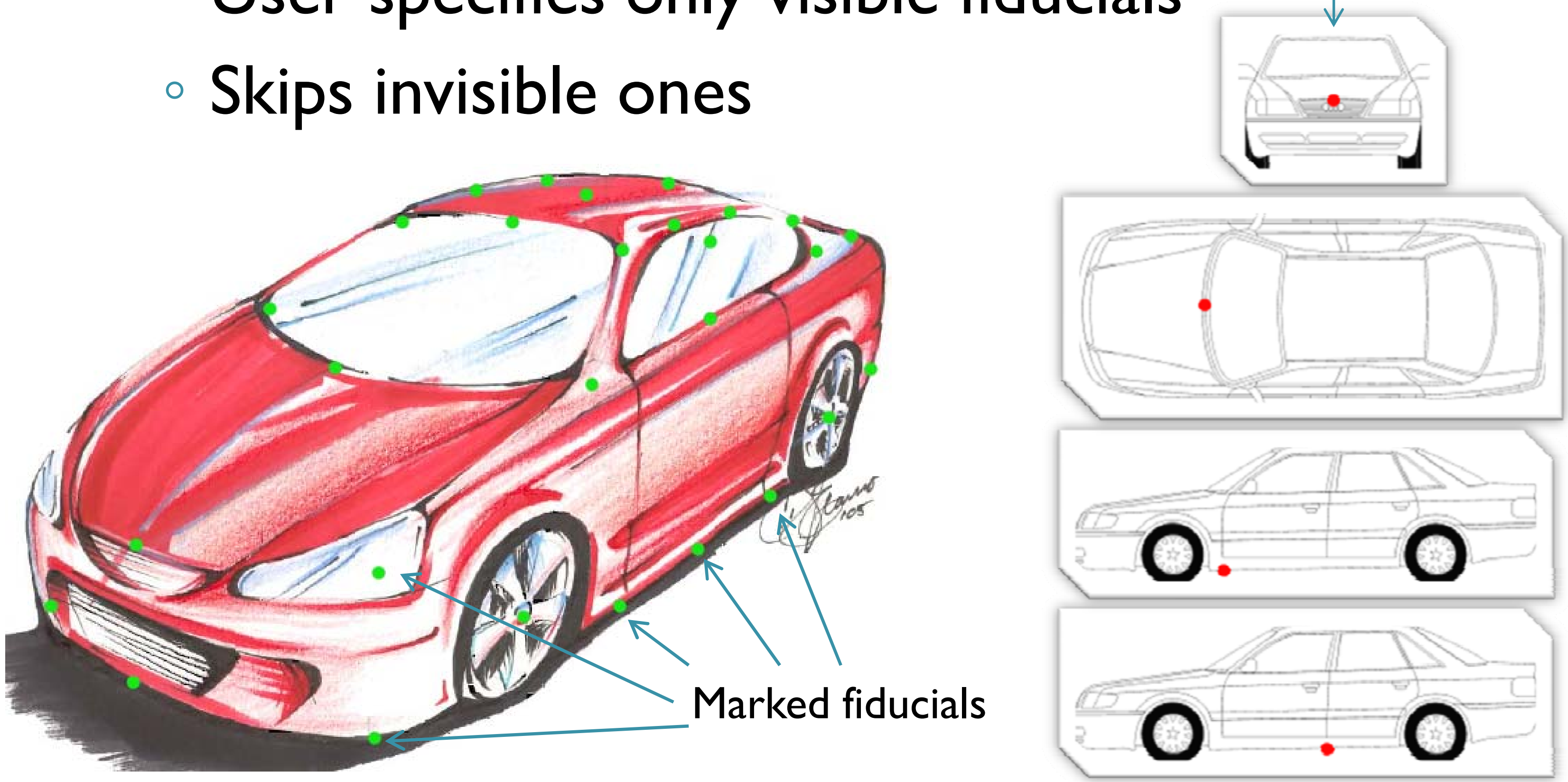
39 fiducial nodes
62 edges
28 surface patches



41 fiducial nodes
66 edges
30 surface patches

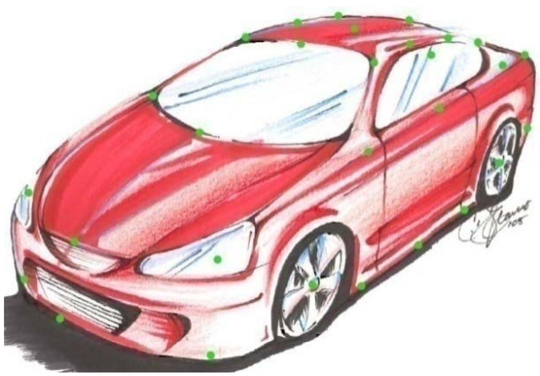
User Input

- User marks fiducial points on the sketch
  - UI widget guides the user
  - User specifies only visible fiducials
  - Skips invisible ones

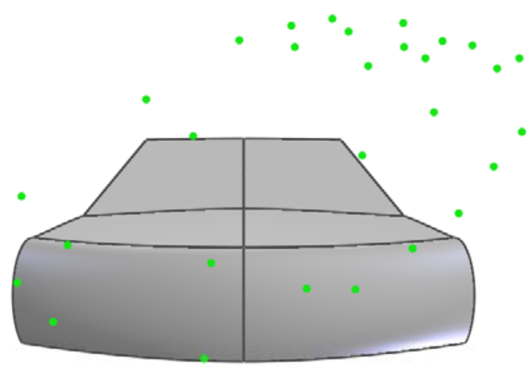


Template Alignment

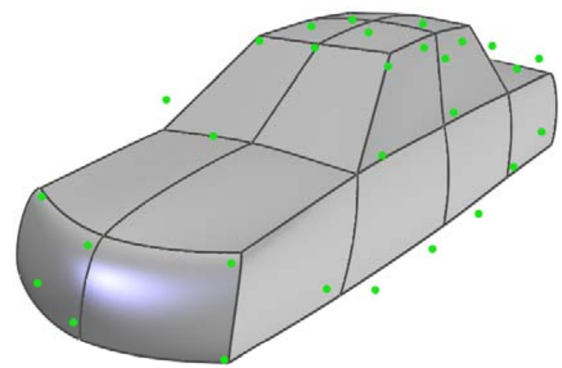
- Align the template with the sketch
  - Match template fiducials with users markers
  - Template is <u>not</u> deformed
- Uses *N-point* camera calibration
  - Best camera in the "*Least Squares*" sense



Input sketch and markers



Template with uncalibrated camera



Template with calibrated camera

Template Alignment

- Formulation [1]
  - User marks *N* fiducial points in the sketch (**p**)
  - We know the corresponding 3D nodes (**P**)
  - Estimate camera properties

Camera Model $\mathbf{p}_{3\mathrm{x}N} = \frac{1}{s} \cdot \mathbf{K}_{3\mathrm{x}3} \cdot [\mathbf{R}_{3\mathrm{x}3}\mathbf{T}_{3\mathrm{x}1}] \cdot \mathbf{P}_{4\mathrm{x}N}$

$$\mathbf{p}_{3\mathrm{x}N} = \begin{bmatrix} u_1 & u_2 & . & u_N \\ v_1 & v_2 & . & v_N \\ 1 & 1 & . & 1 \end{bmatrix}$$

2D fiducial points

$$\mathbf{K}_{3\mathrm{x}3} = \begin{bmatrix} \alpha & -\alpha \cdot \cot\theta & u_0 \\ 0 & \dfrac{\beta}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsic camera properties (unknown)

$$\mathbf{P}_{4\mathrm{x}N} = \begin{bmatrix} x_1 & x_2 & . & x_N \\ y_1 & y_2 & . & y_N \\ z_1 & z_2 & . & z_N \\ 1 & 1 & . & 1 \end{bmatrix}$$

3D fiducial points

$\mathbf{R}, \mathbf{T}$ rotation/translation matrices (unknown)

s scale factor (unknown)

α, β scale factors in u and v directions (unknown)

u_0, v_0 camera center (unknown)

θ skew (radians) between u and v axes (unknown)

Template Alignment

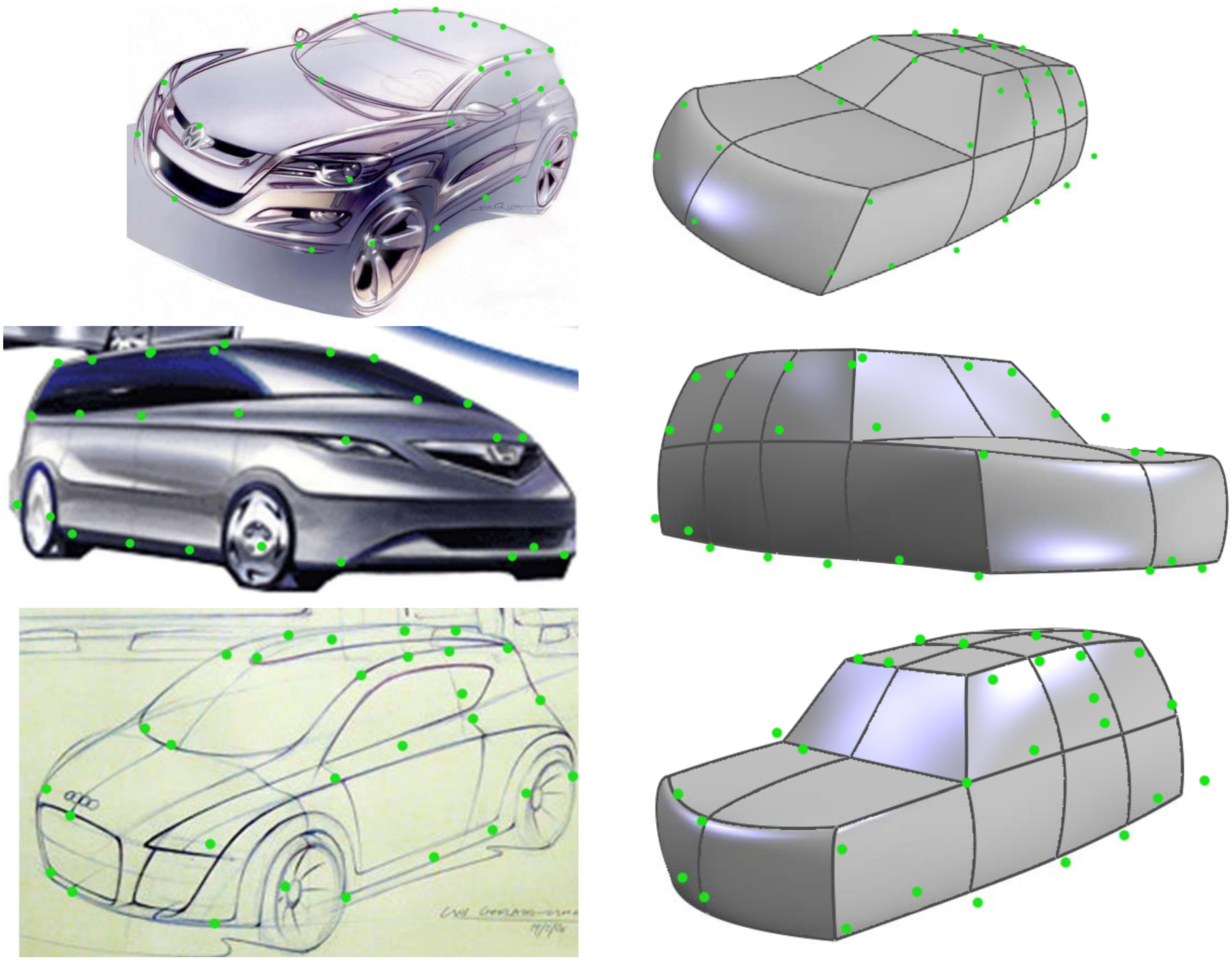
- Introduce matrix $\mathbf{M}$: $\mathbf{M}_{3\text{x}4} = \frac{1}{s} \cdot \mathbf{K}_{3\text{x}3} \cdot [\mathbf{R}_{3\text{x}3} \mathbf{T}_{3\text{x}1}]$
- Compute $\mathbf{M}$ from $\mathbf{p}$ and $\mathbf{P}$ using LDT [2]
- Rewrite $\mathbf{M}$: $\mathbf{M}_{3\text{x}4} = [\mathbf{A}_{3\text{x}3} \mathbf{b}_{3\text{x}1}]$
- Identify unknowns from $\mathbf{A}$ and $\mathbf{b}$ [3]

$$
\begin{aligned}
s &= \pm 1/||\mathbf{a}_3||, \\
\mathbf{r}_3 &= s\,\mathbf{a}_3, \\
u_0 &= s^2(\mathbf{a}_1 \cdot \mathbf{a}_3), \\
v_0 &= s^2(\mathbf{a}_2 \cdot \mathbf{a}_3), \\
\cos(\theta) &= -\frac{(\mathbf{a}_1 \times \mathbf{a}_3) \cdot (\mathbf{a}_2 \times \mathbf{a}_3)}{||\mathbf{a}_1 \times \mathbf{a}_3|| \cdot ||\mathbf{a}_2 \times \mathbf{a}_3||}, \\
\alpha &= s^2 ||\mathbf{a}_1 \times \mathbf{a}_3|| \cdot \sin(\theta), \\
\beta &= s^2 ||\mathbf{a}_2 \times \mathbf{a}_3|| \cdot \sin(\theta), \\
\mathbf{r}_1 &= \frac{\mathbf{a}_2 \times \mathbf{a}_3}{||\mathbf{a}_2 \times \mathbf{a}_3||}, \\
\mathbf{r}_2 &= \mathbf{r}_3 \times \mathbf{r}_1, \\
\mathbf{t} &= s \cdot \mathbf{K}^{-1}\mathbf{b}
\end{aligned}
$$

Where $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$ are the column vectors of **A**.

If $\mathbf{t}_z$ <0 , switch s and reevaluate.

Template Alignment Examples



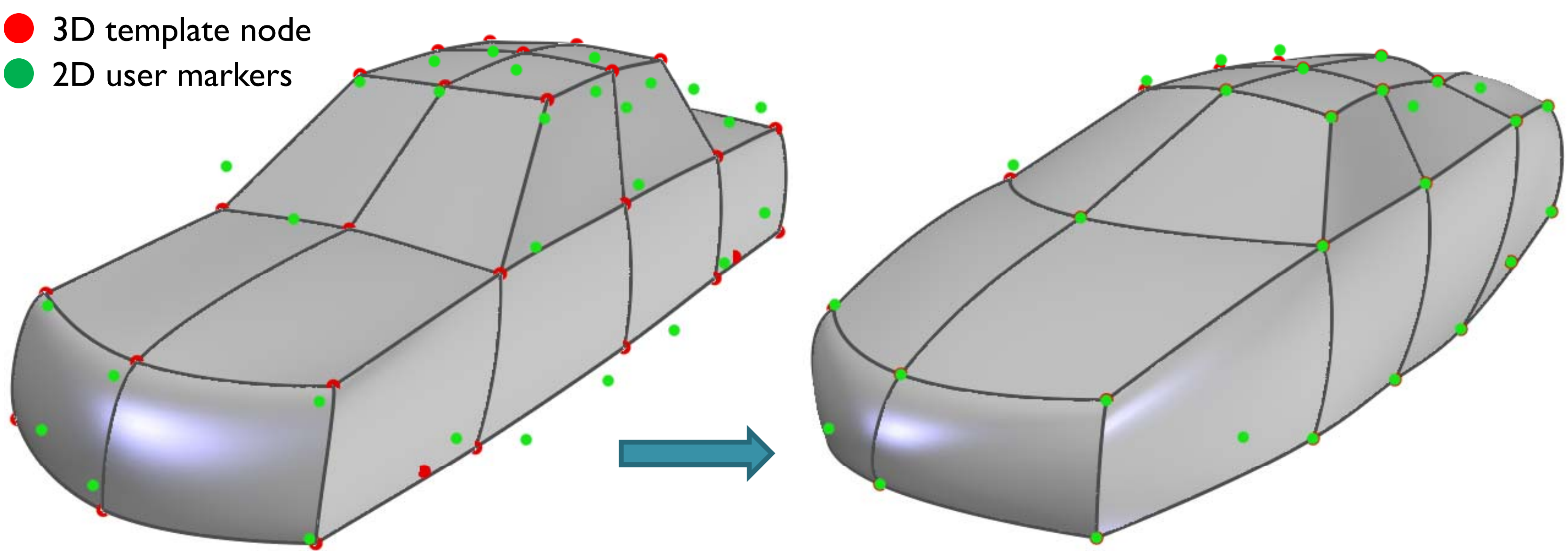
Template Alignment

- Our new template alignment approach provides <u>much improved results</u> compared to old, Bounding Box-based approach

- However, we are still looking ways to <u>further improve</u> this

Template Deformation

- Elastically deform the template
  - Adjust template nodes in 3D such that:
    (1) Template nodes match user's markers in 2D
    (2) The template has an acceptable 3D shape



Original, undeformed template

Deformed template: red points match well with green points

Template Deformation

- Cost function to minimize:

$$H = \alpha \sum_{i=1}^{m} \left\| \mathbf{F}_i - P(\mathbf{Vs'}_i) \right\| + \beta \sum_{j=1}^{n} \left\| \mathbf{V}_j - \mathbf{V'}_j \right\|$$

Minimizes mismatch in 2D

Minimizes deviation from undeformed template

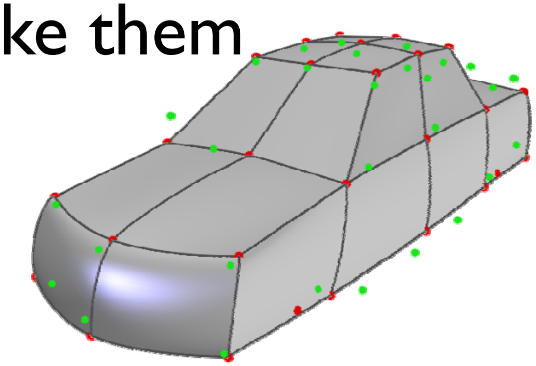
$\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \in R^3$	3D positions of original wireframe nodes
$\mathbf{V'} = \{\mathbf{v'}_1, \ldots, \mathbf{v'}_n\} \in R^3$	3D positions of deformed wireframe nodes. This is what we are trying to determine
$\mathbf{Vs'} = \{\mathbf{v'}_1, \ldots, \mathbf{v'}_m\} \subset \mathbf{V'}$	3D wireframe nodes whose fiducial points are marked by the user $m \le n$
$\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_m\} \in R^2$	2D screen coordinates of user's fiducial points
$P : R^3 \to R^2$	Function that projects 3D world coordinates to 2D image coordinates using the current projection matrix.

Template Deformation

- Cost function to minimize:

$$H = \alpha \sum_{i=1}^{m} \left\| \mathbf{F}_i - P(\mathbf{V}\mathbf{s}'_i) \right\| + \beta \sum_{j=1}^{n} \left\| \mathbf{V}_j - \mathbf{V}'_j \right\|$$

  - First term in H ensures that red dots match green dots in image plane.
  - Second term ensures that deformed template deviates minimally from undeformed template.
  - We scale the two terms to make them magnitude-wise comparable.
  - We take $\alpha=\beta=0.5$



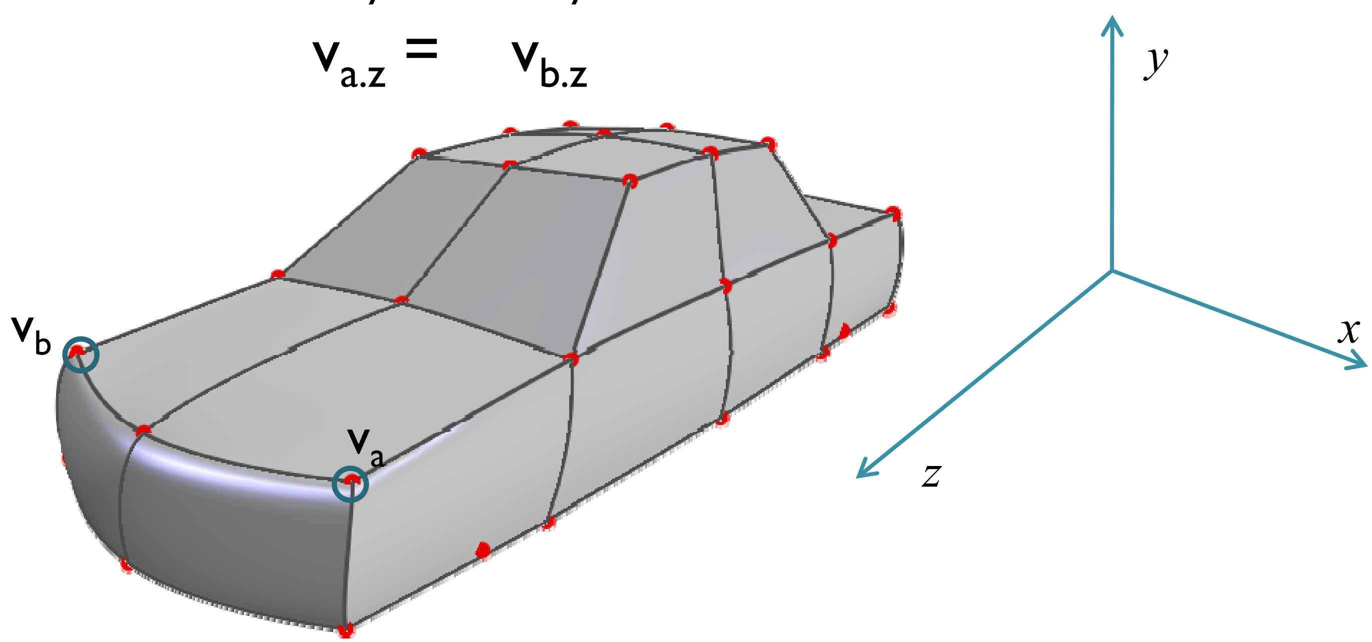
Template Deformation

- There are $3n$ optimization parameters (x,y,z for each node) where n is the number of template fiducials (i.e., red nodes)
- We use Sequential Quadratic Programming (SQP) technique to solve the optimization problem.

Optimization Constraints

- Symmetry constraints – Type I
  - For each symmetric node pair (15 pairs below), we have 3 linear equality constraints:

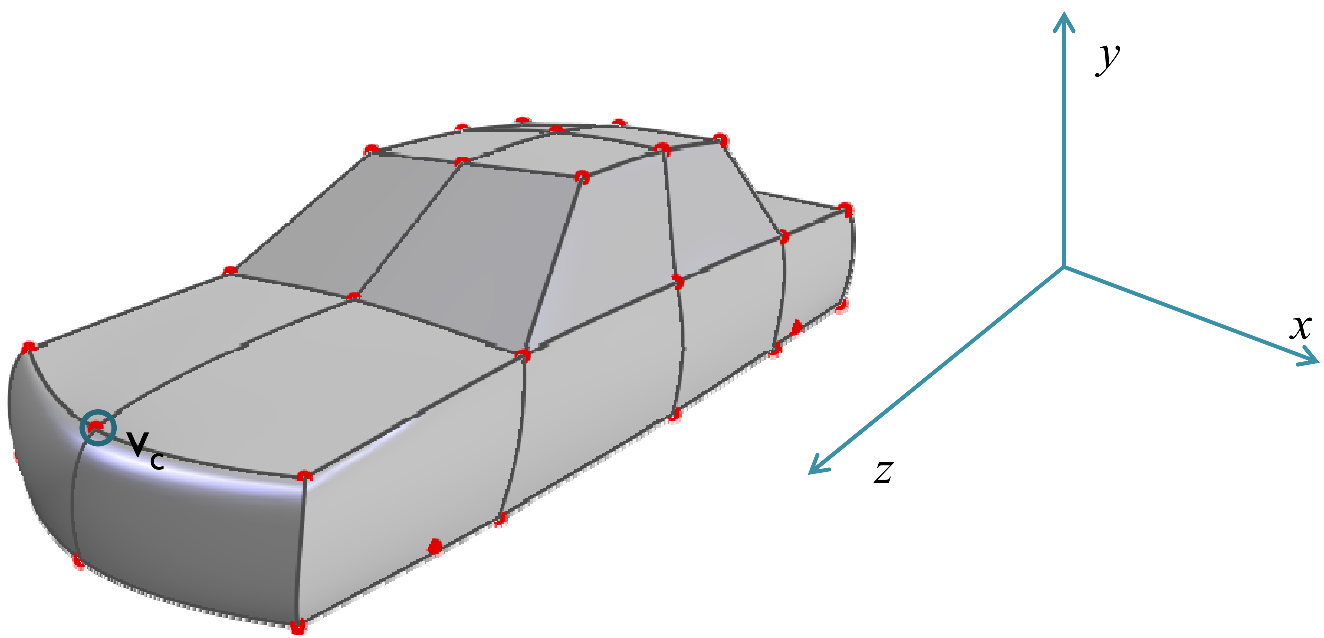
$$v_{a.x} = - v_{b.x}$$
$$v_{a.y} = v_{b.y}$$
$$v_{a.z} = v_{b.z}$$



Optimization Constraints

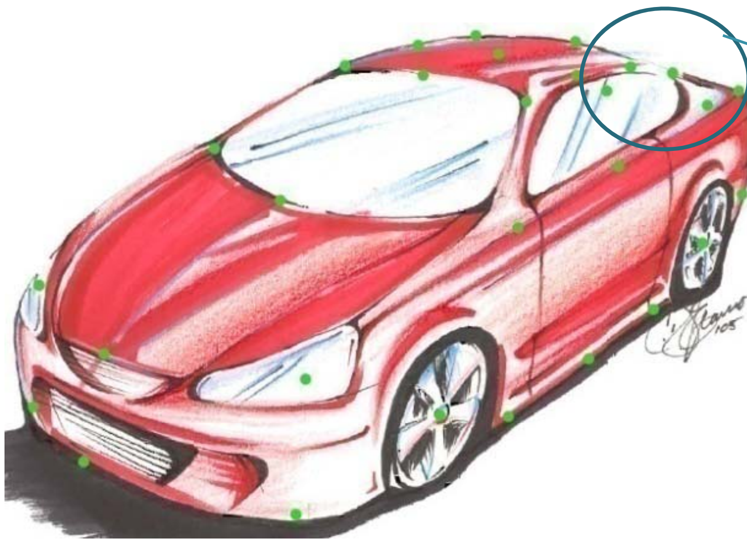
- Symmetry constraints – Type II
  - For each node on symmetry plane (9 nodes below), we have 1 linear equality constraint:

$$v_{c.x} = 0.00$$



Optimization Constraints

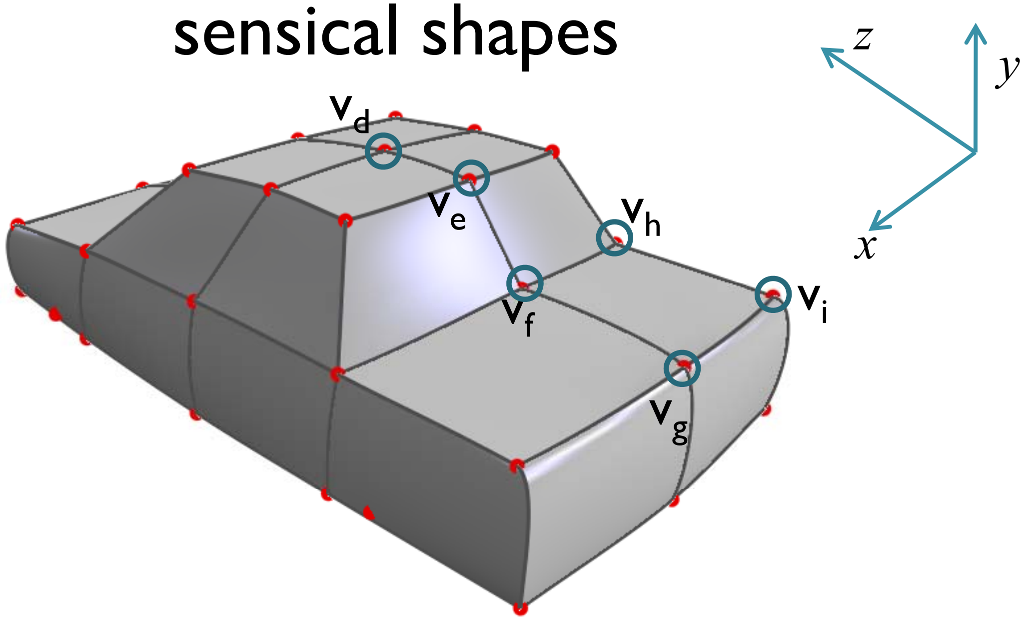
- Shape soundness constraints
  - Optimization-based deformation can break common-sense rules about car shape
    - Inaccurate, cluttered markers (e.g. markers far from the camera) can yield erroneous shapes



Potentially inaccurate / unreliable markers

Optimization Constraints

- Shape soundness constraints
  - Shape soundness constraints prevents non-sensical shapes

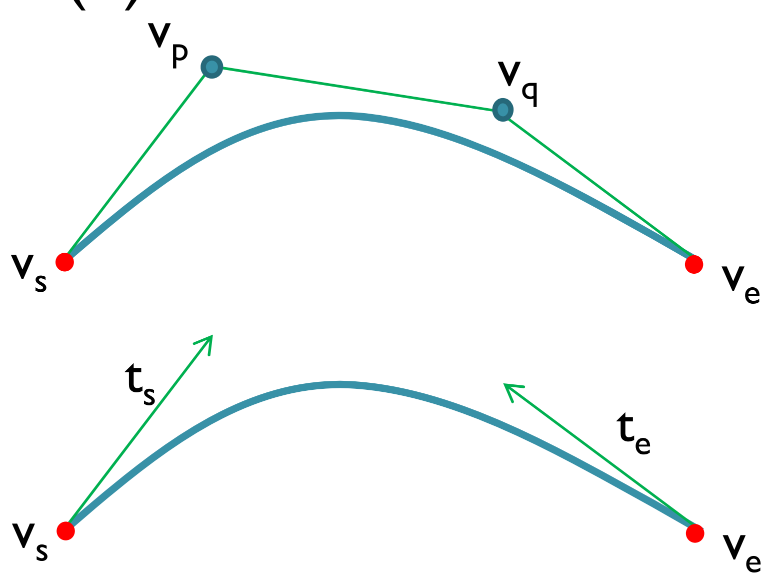


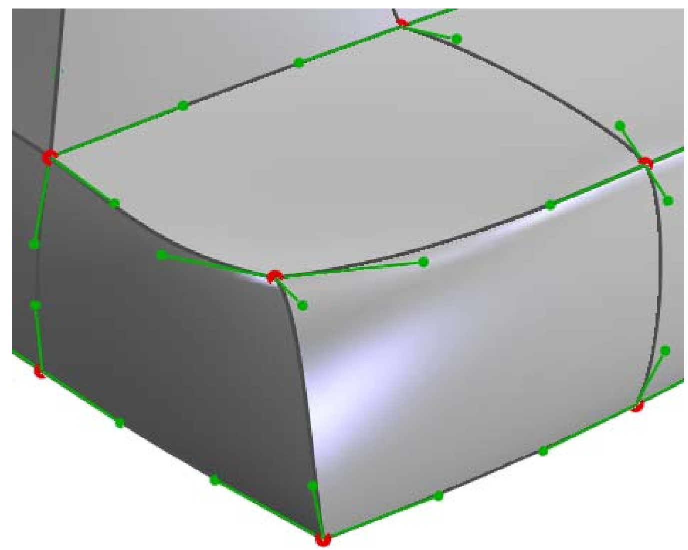
$v_{d.z} > v_{e.z}$
$v_{e.z} > v_{f.z}$
$v_{f.z} > v_{g.z}$
$v_{h.z} > v_{f.z}$
$v_{i.z} > v_{g.z}$

  - Similar constraints exist for rest of the car
  - These constraints can be abandoned if desired

Edge Representation

- Edges are represented as Cubic Beziers*
- We maintain two interchangeable forms:
  - (1) Four control polygon nodes, *or*
  - (2) Two end nodes + two end tangents

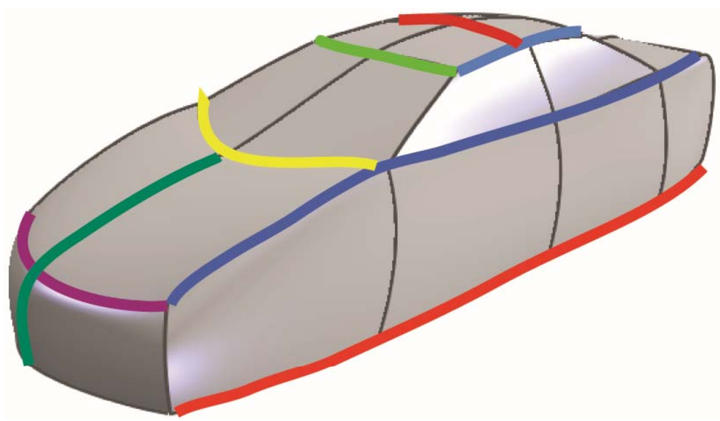




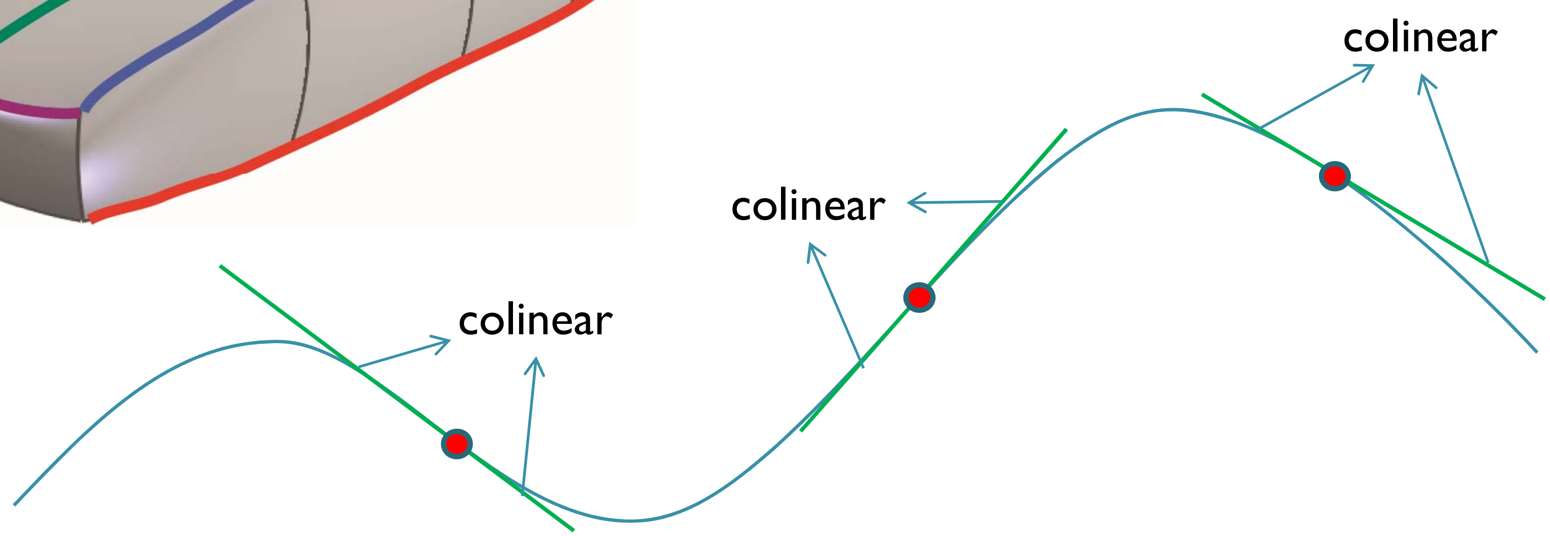
* Conventionally, Bezier curves are represented with 4 absolute points. In our case, the above interchangeable representation has been adopted for computational convenience.

G^1 Constraints

- Certain edges maintain G^1 continuity at all times
- This enables natural looking curves



Colored Curves consist of G^1 continuous edges



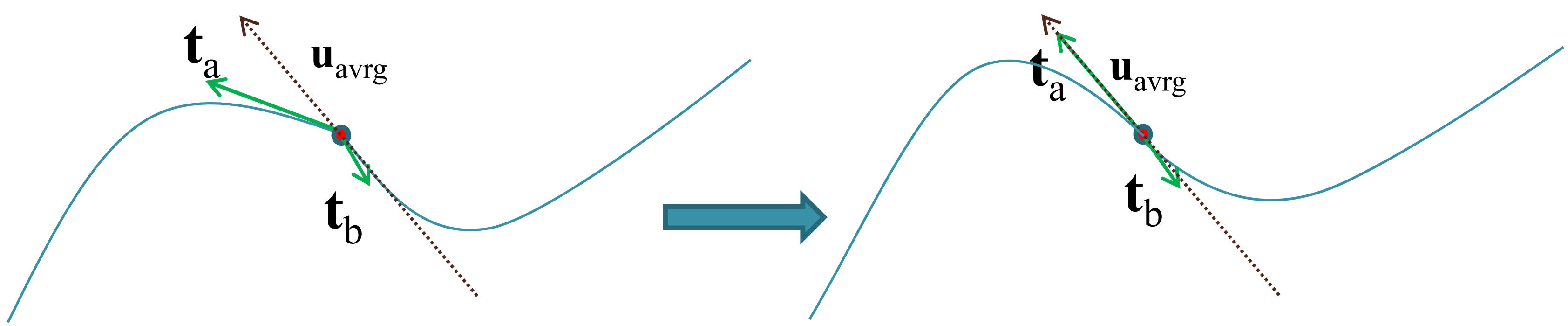
G^1 Constraints

- Algorithm for imposing G^1:
  - Compute a weighted average direction:

$$\mathbf{u}_{avrg} = \mathbf{t}_a + (-1)\mathbf{t}_b, \;\; normalize\,(\mathbf{u}_{avrg})$$

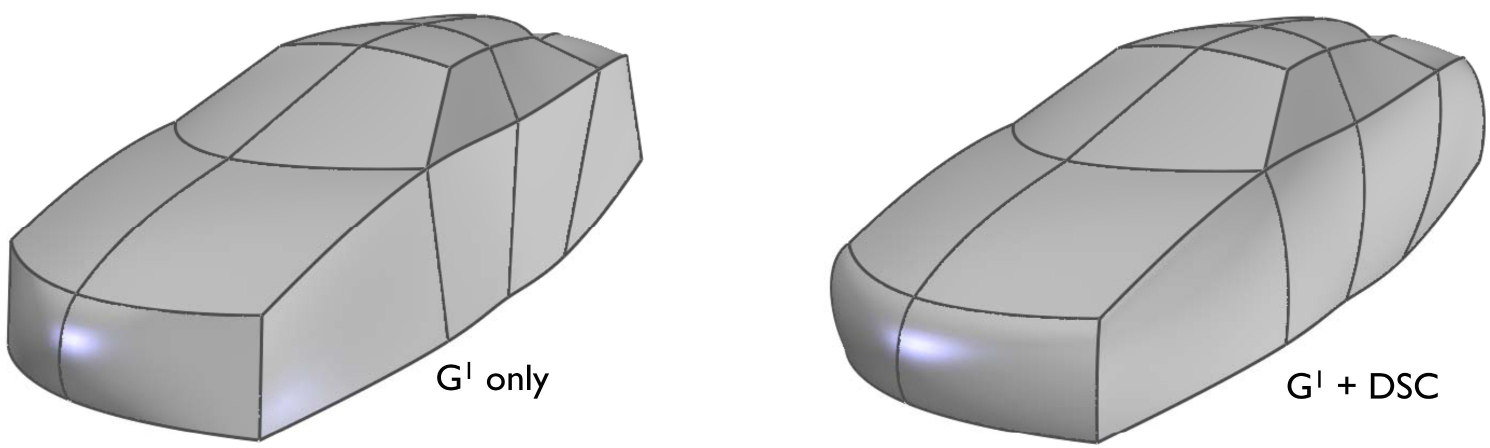
  - Adjust $\mathbf{t}_a$ and $\mathbf{t}_b$ to lie on $\mathbf{u}_{avrg}$, while maintaining their original magnitudes

$$\mathbf{t}_a \leftarrow mag(\mathbf{t}_a)\cdot \mathbf{u}_{avrg},$$
$$\mathbf{t}_b \leftarrow mag(\mathbf{t}_b)\cdot(-1)\cdot \mathbf{u}_{avrg},$$



Default Shape Constraints (DSC)

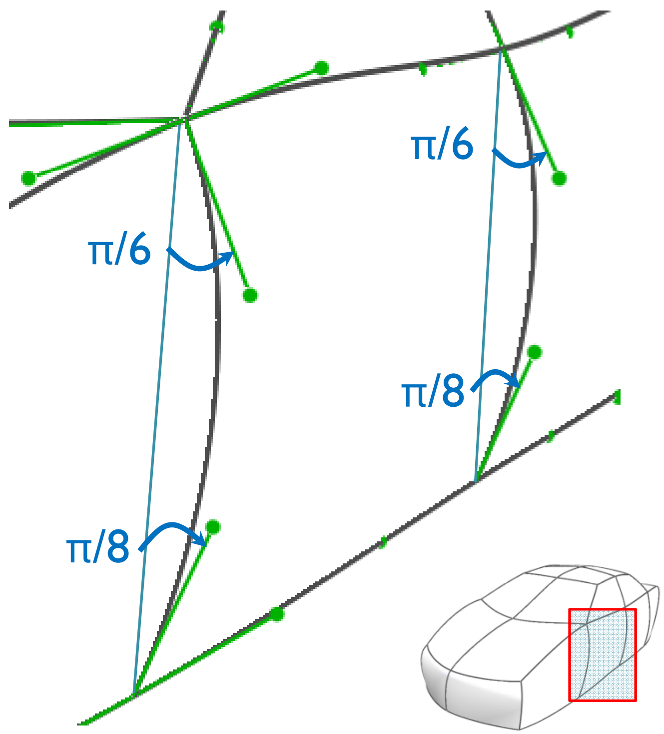
- We impose certain constraints immediately after template deformation
- These constraints yield better shapes



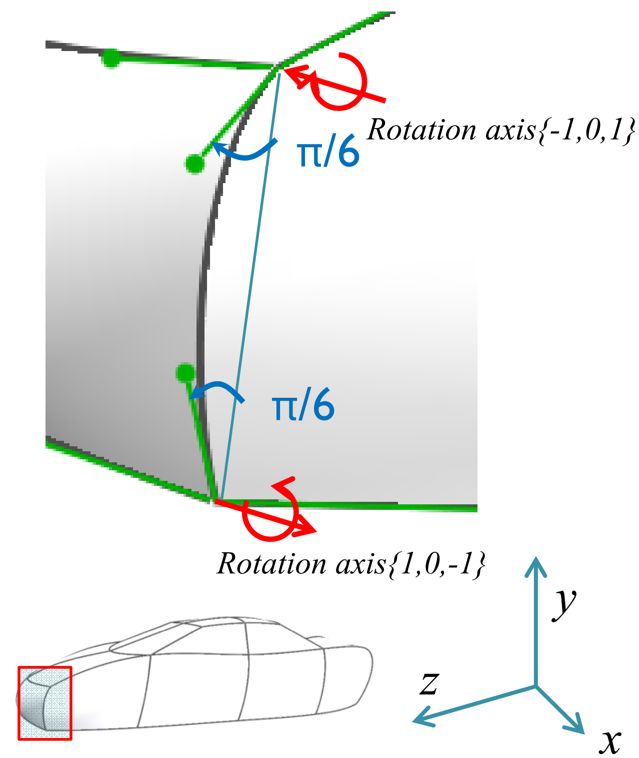
Shapes immediately after template deformation

Default Shape Constraints (DSC)

- Rounding DSC



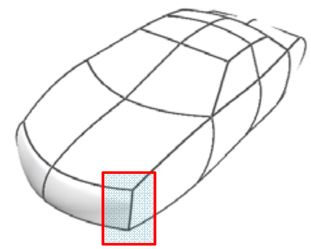
Side panel edges are pulled out

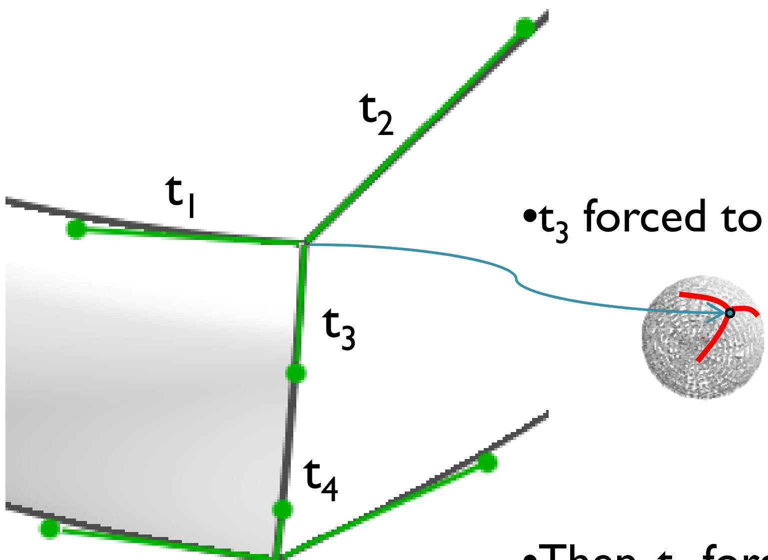


Front and rear corner edges are pulled out

Default Shape Constraints (DSC)

- Coplanarity DSC





•t_3 forced to lie on the plane defined by t_1 and t_2

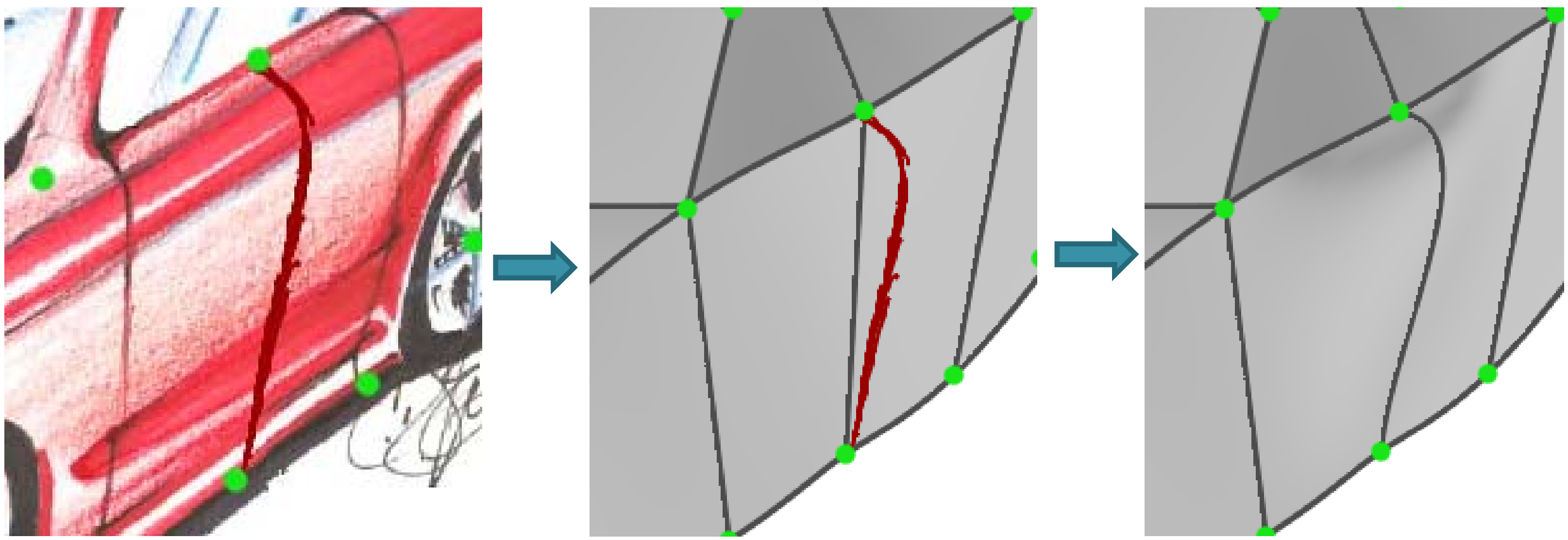
Simulates a locally spherical corner

•Then, t_4 forced to be coplanar with t_3

Applied to two front and two rear corner edges

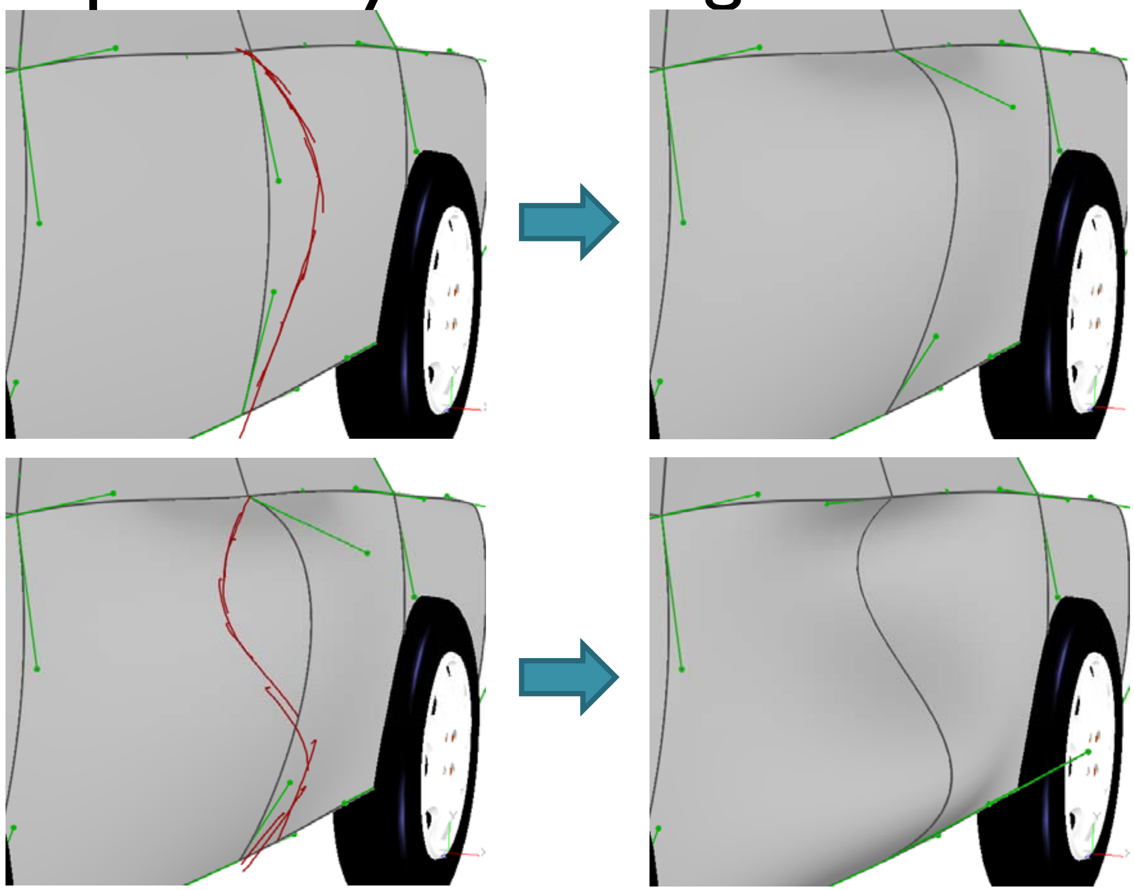
Edge Manipulation

- Edges can be modified from arbitrary viewpoints by sketching desired shape



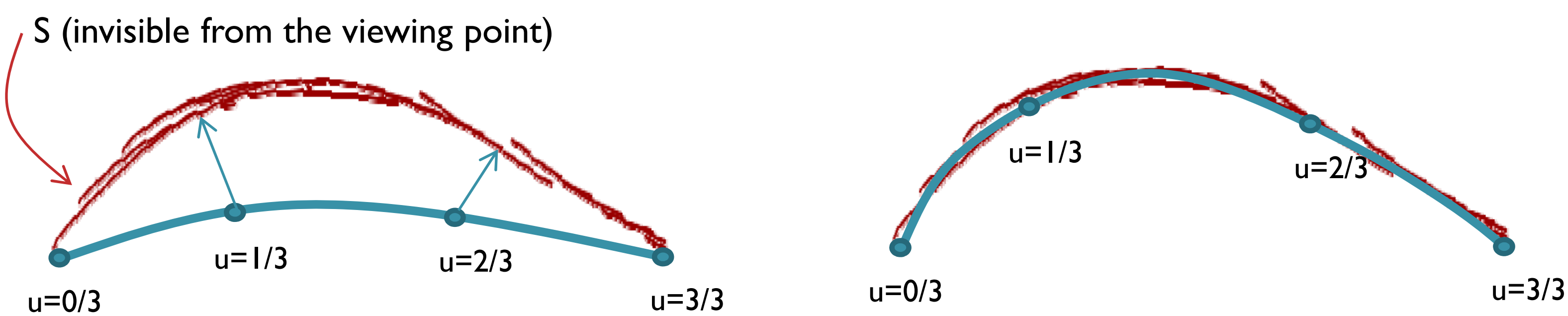
Edge Manipulation

- Edges can be modified from arbitrary viewpoints by sketching desired shape



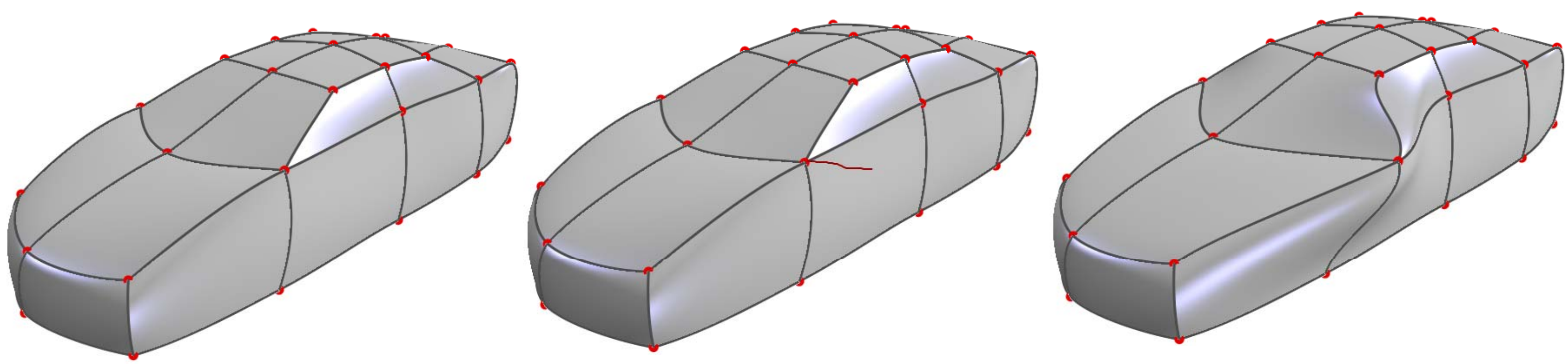
Edge Manipulation

- For edge modification, we use a minimum surprise method similar to [1]
- Algorithm:
  - Identify the intended curve to be modified
  - Create a 3D surface S that starts at eye, extends into screen passing through pen strokes.
  - Project points u=1/3 and u=2/3 of original 3D cubic curve onto S
  - Use a four-point Hermite interpolation [4] to reconstruct the new cubic curve



Node/tangent Manipulation

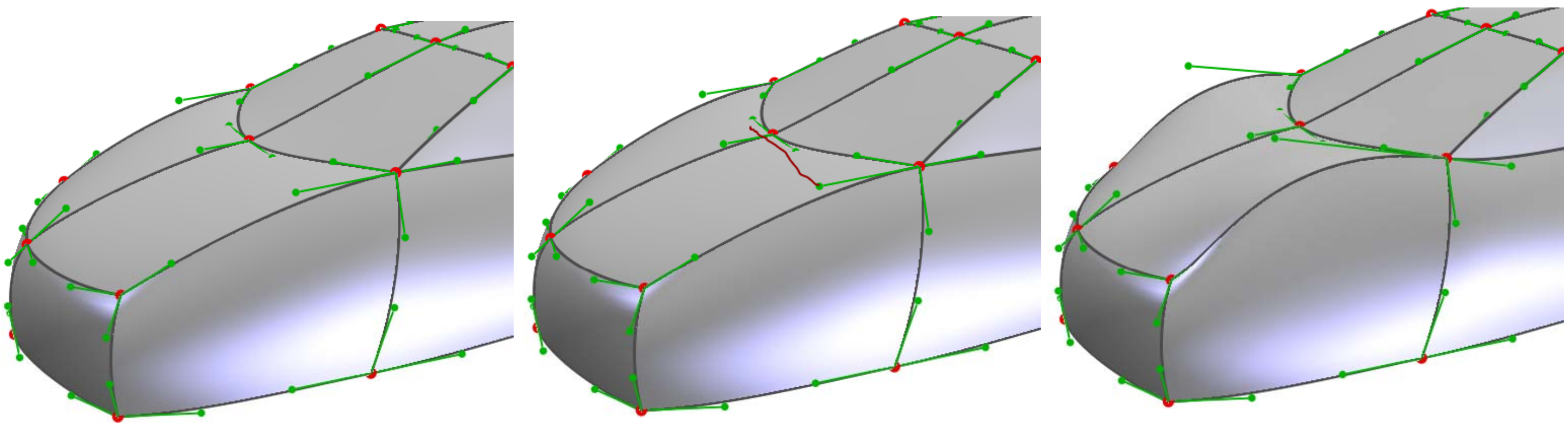
- Template nodes can be manually adjusted by simple point-and-drag



  - Nodes move parallel to current image plane
  - Edge tangents are kept unchanged (by design)
  - Symmetry automatically preserved

Node/tangent Manipulation

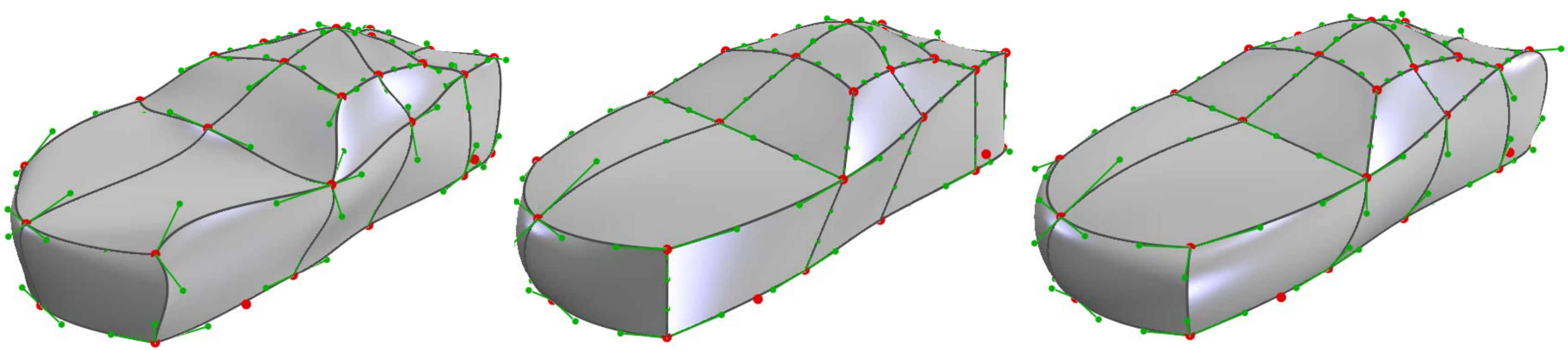
- Tangents can be manually adjusted by simple point-and-drag



  - Tangent tips move parallel to image plane
  - G1 preserved with neighboring edge
  - Symmetry automatically preserved

Edge Beautification

- Disfigured edges can be beautified by automatically applying:
  - (1) "annealing" (fit a simple, cubic chain to *n* nodes)
  - (2) default shape constraints (DSC)
  - Template node positions are not modified, only the tangent vectors.



Disfigured edges Annealed edges Annealed + DSC edges

Surface Representation

- Bicubic Coons patches [5]
  - H_i^3: Cubic Hermite interpolants

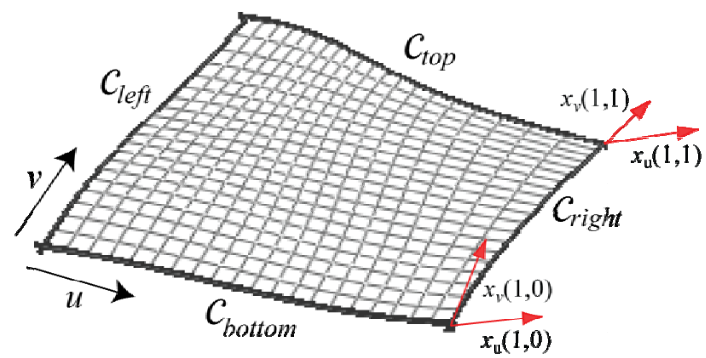
$\mathbf{x}(u,0)=\mathbf{c}_{bottom}(u) \quad \mathbf{x}(u,1)=\mathbf{c}_{top}(u) \quad \mathbf{x}(0,v)=\mathbf{c}_{left}(v) \quad \mathbf{x}(1,v)=\mathbf{c}_{right}(v) \quad u,v\in[0,1]$

$\boxed{\mathbf{p}(u,v)=\mathbf{h}_c(u,v)+\mathbf{h}_d(u,v)-\mathbf{h}_{cd}(u,v)}$ Points on surface

$$\mathbf{h}_c(u,v)=H_0^3(u)\mathbf{x}(0,v)+H_1^3(u)\mathbf{x}_u(0,v)+H_2^3(u)\mathbf{x}_u(1,v)+H_3^3(u)\mathbf{x}(1,v)$$

$$\mathbf{h}_d(u,v)=H_0^3(v)\mathbf{x}(u,0)+H_1^3(v)\mathbf{x}_u(u,0)+H_2^3(v)\mathbf{x}_u(u,1)+H_3^3(v)\mathbf{x}(u,1)$$

$$\mathbf{h}_{cd}(u,v)=\begin{bmatrix}H_0^3(u)\\H_1^3(u)\\H_2^3(u)\\H_3^3(u)\end{bmatrix}^T\begin{bmatrix}\mathbf{x}(0,0) & \mathbf{x}_v(0,0) & \mathbf{x}_v(0,1) & \mathbf{x}(0,1)\\ \mathbf{x}_u(0,0) & \mathbf{x}_{uv}(0,0) & \mathbf{x}_{uv}(0,1) & \mathbf{x}_u(0,1)\\ \mathbf{x}_u(1,0) & \mathbf{x}_{uv}(1,0) & \mathbf{x}_{uv}(1,1) & \mathbf{x}_u(1,1)\\ \mathbf{x}(1,0) & \mathbf{x}_v(1,0) & \mathbf{x}_v(1,1) & \mathbf{x}(1,1)\end{bmatrix}\begin{bmatrix}H_0^3(v)\\H_1^3(v)\\H_2^3(v)\\H_3^3(v)\end{bmatrix}$$



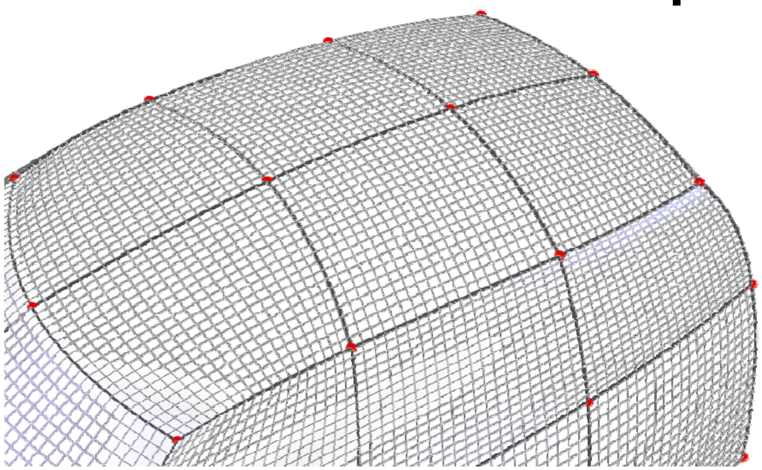
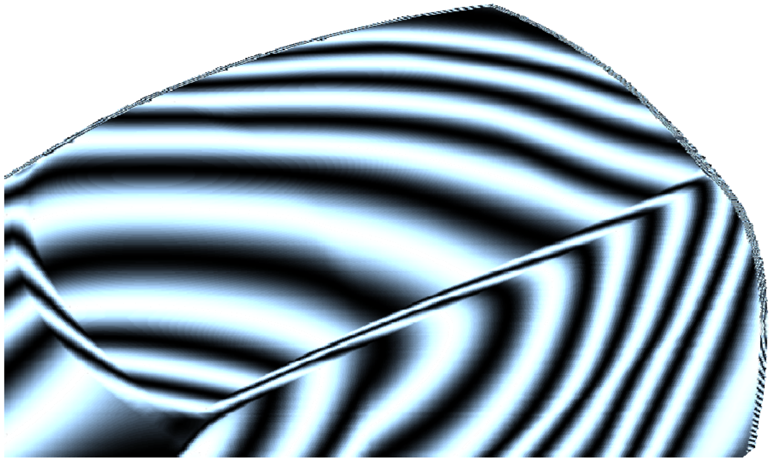
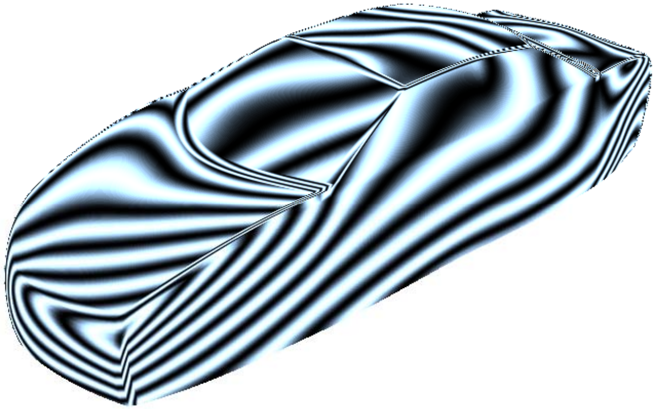
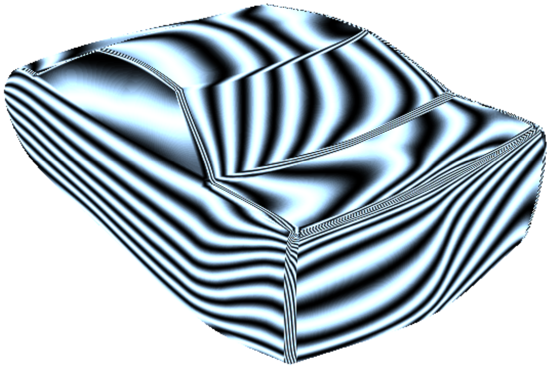
$$\mathbf{x}_v(u,0)=H_0^3(u)\mathbf{x}_v(0,0)+H_3^3(u)\mathbf{x}_v(1,0) \quad \mathbf{x}_v(u,1)=H_0^3(u)\mathbf{x}_v(0,1)+H_3^3(u)\mathbf{x}_v(1,1)$$

Blending function for cross-boundary derivatives. Similar functions for $x_u(.,.)$

We take twist vectors $x_{uv}(.,.) = 0$

Surface Representation

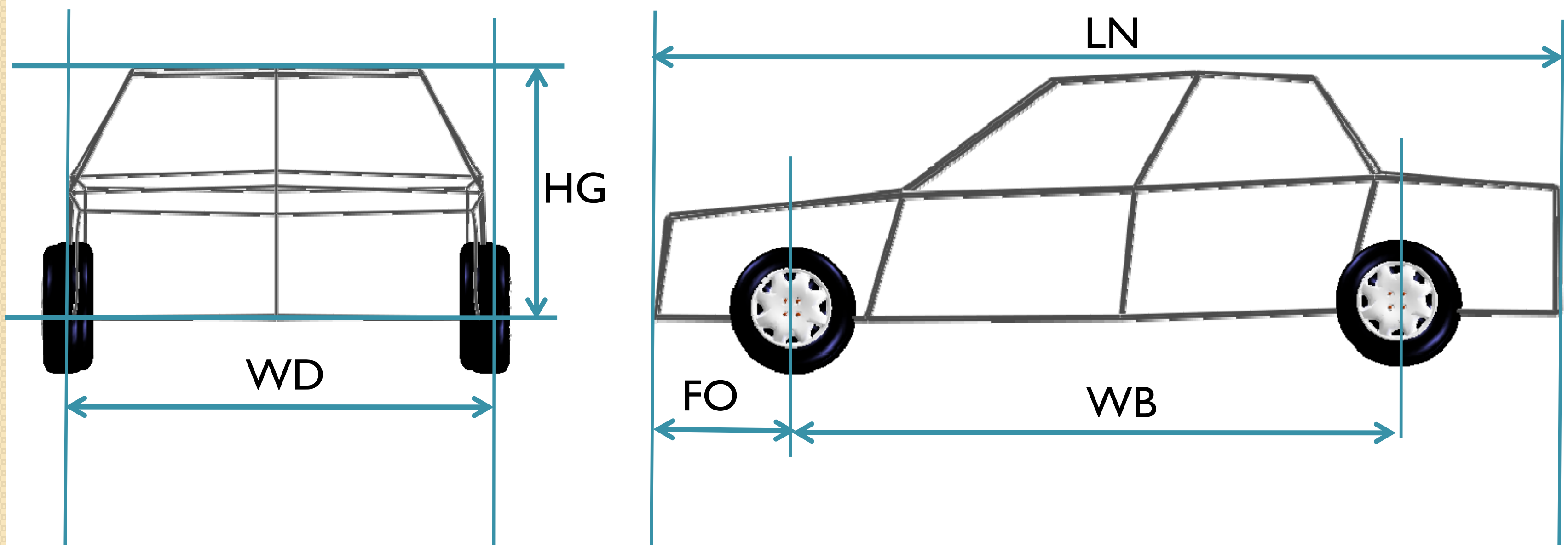
- G^1 edge continuity produces smoothly blended surface patches

Post-Dimensioning

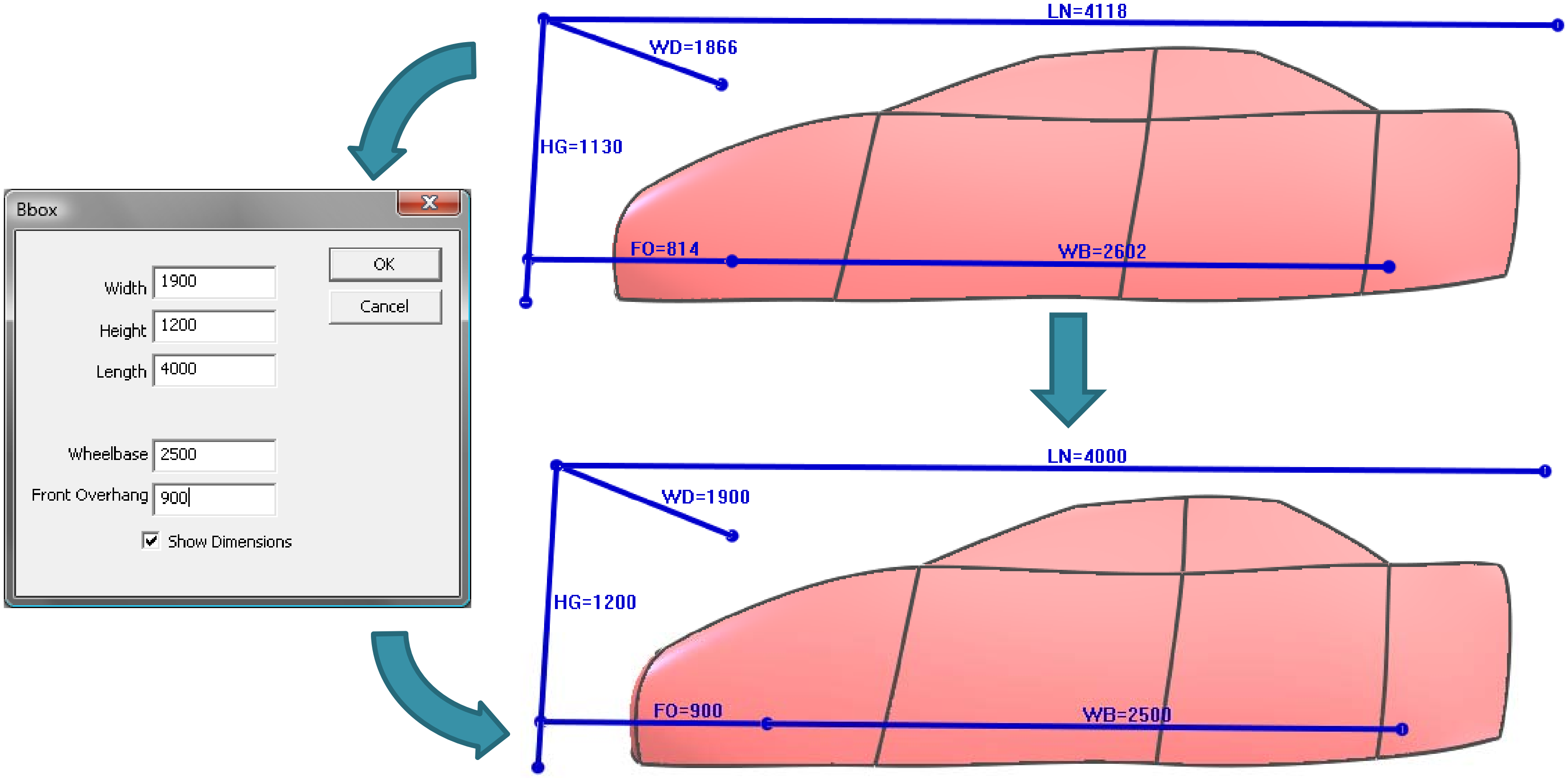
- Allows user to specify key dimensions
- Preserves shape as much as possible
- 5 dimensions can be set

WD: width
HG: Height
LN: length
FO: Front overhang
WB: Wheel base



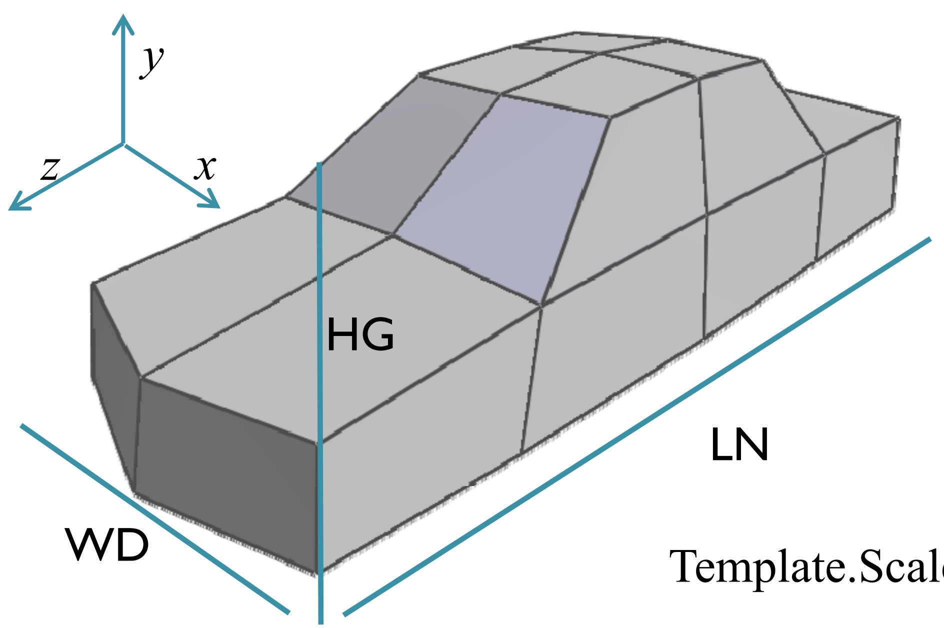
Post-Dimensioning

- User enters desired values
- Shape is automatically updated



Post-Dimensioning

- Performed in two steps:
  - (1) Non-uniform scaling with WD, HG, LN
    - All nodes, edges and surfaces are scaled taking (0,0,0) as the origin
    - Edges are converted to conventional Bezier form (absolute positions of 4 control points) to take advantage of affine invariance



$$s_x = \frac{\mathrm{WD}_{new}}{\mathrm{WD}_{old}}$$

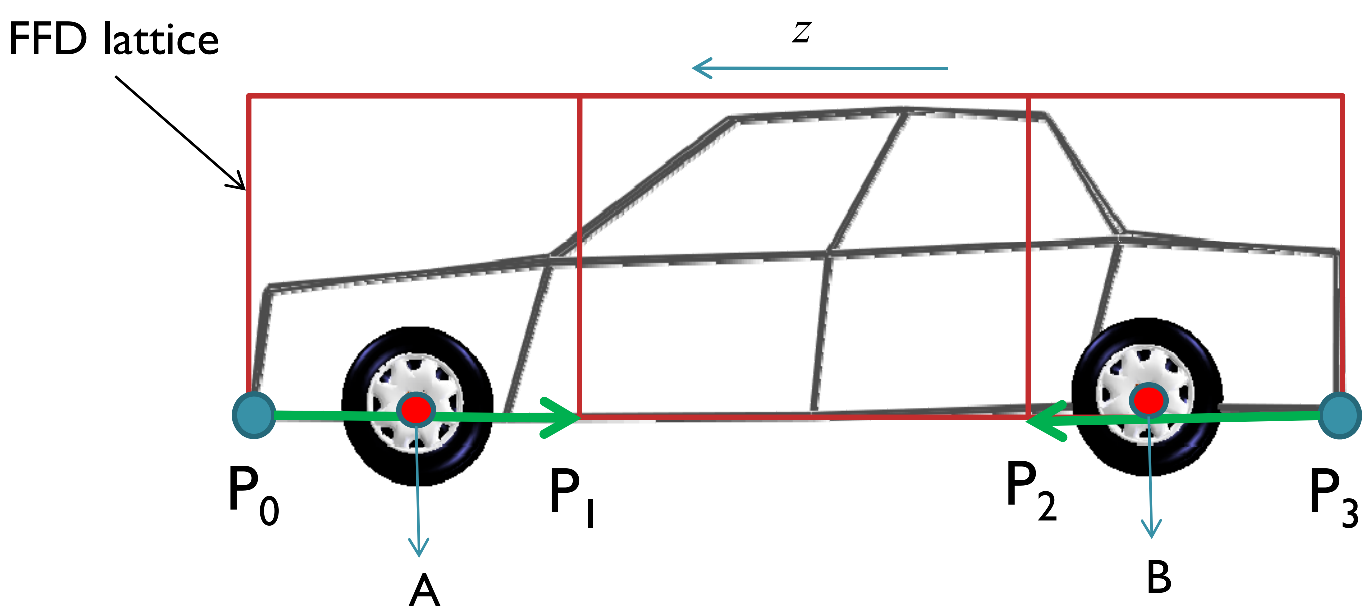
$$s_y = \frac{\mathrm{HG}_{new}}{\mathrm{HG}_{old}}$$

$$s_z = \frac{\mathrm{LN}_{new}}{\mathrm{LN}_{old}}$$

Template.Scale(double s_x, double s_y, double s_z)

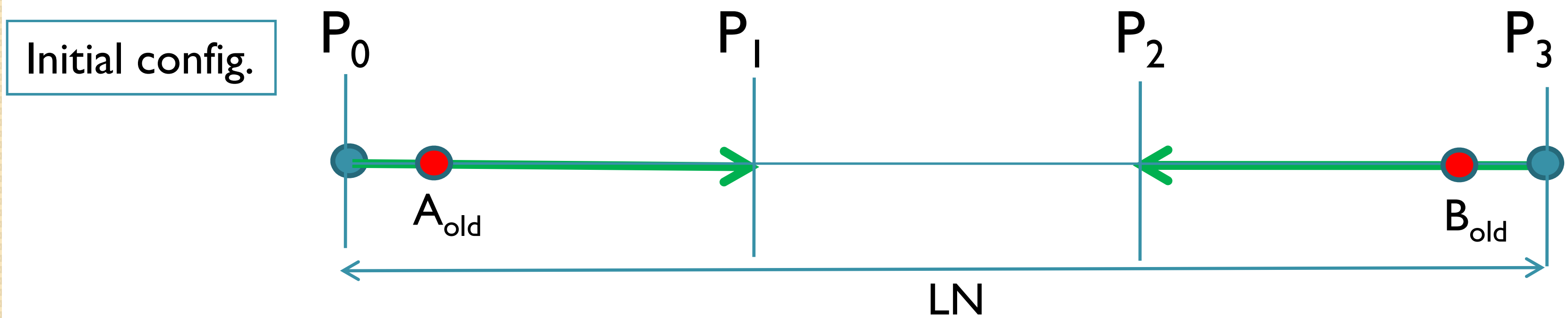
Post-Dimensioning

- Performed in two steps:
  - (2) Soft deformation with FO and WB
    - Idea: 1D, Cubic Free-Form Deformation
      - Move P_1 and P_2 parallel to z-axis to obtain FO and WB
      - Deform the volume together with P_1 and P_2



Post-Dimensioning

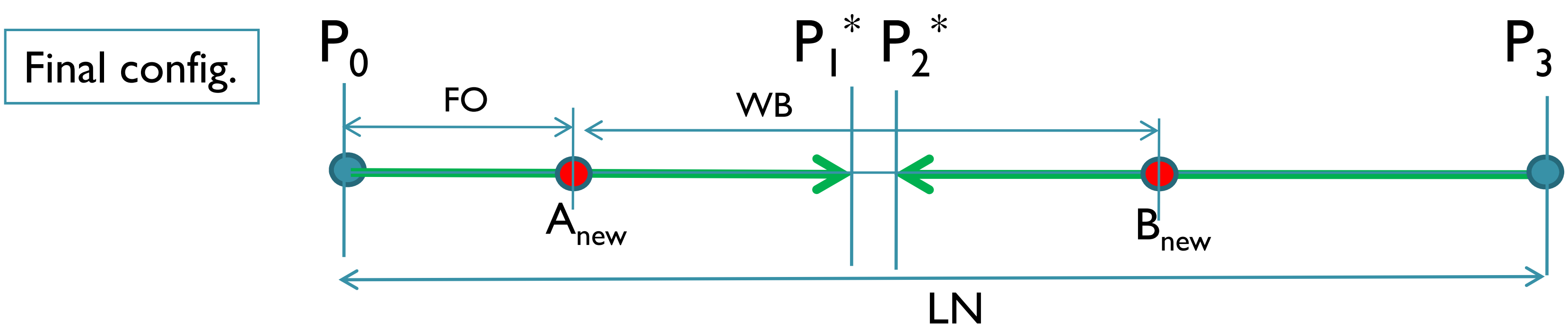
- Performed in two steps:
  - (2) Soft deformation with FO and WB
    - P0, P1,P2,P3 form a 1D cubic Bezier curve



$|P_0P_1| = |P_1P_2| = |P_2P_3| = LN/3$

Post-Dimensioning

- Performed in two steps:
  - (2) Soft deformation with FO and WB
    - P0, P1,P2,P3 form a 1D cubic Bezier curve



$FO = |P_0A_{new}|$, $WB = |A_{new}B_{new}|$

Post-Dimensioning

- Algorithm
  - Find parametric coordinates: $u_A = |A_{old}-P_0|/LN$, $u_B = |B_{old}-P_0|/LN$
  - Find points P_1^* and P_2^* such that $A_{old} \rightarrow A_{new}$, $B_{old} \rightarrow B_{new}$

$$A_{new} = B_0(u_A)P_0 + B_1(u_A)P_1^* + B_2(u_A)P_2^* + B_3(u_A)P_3$$

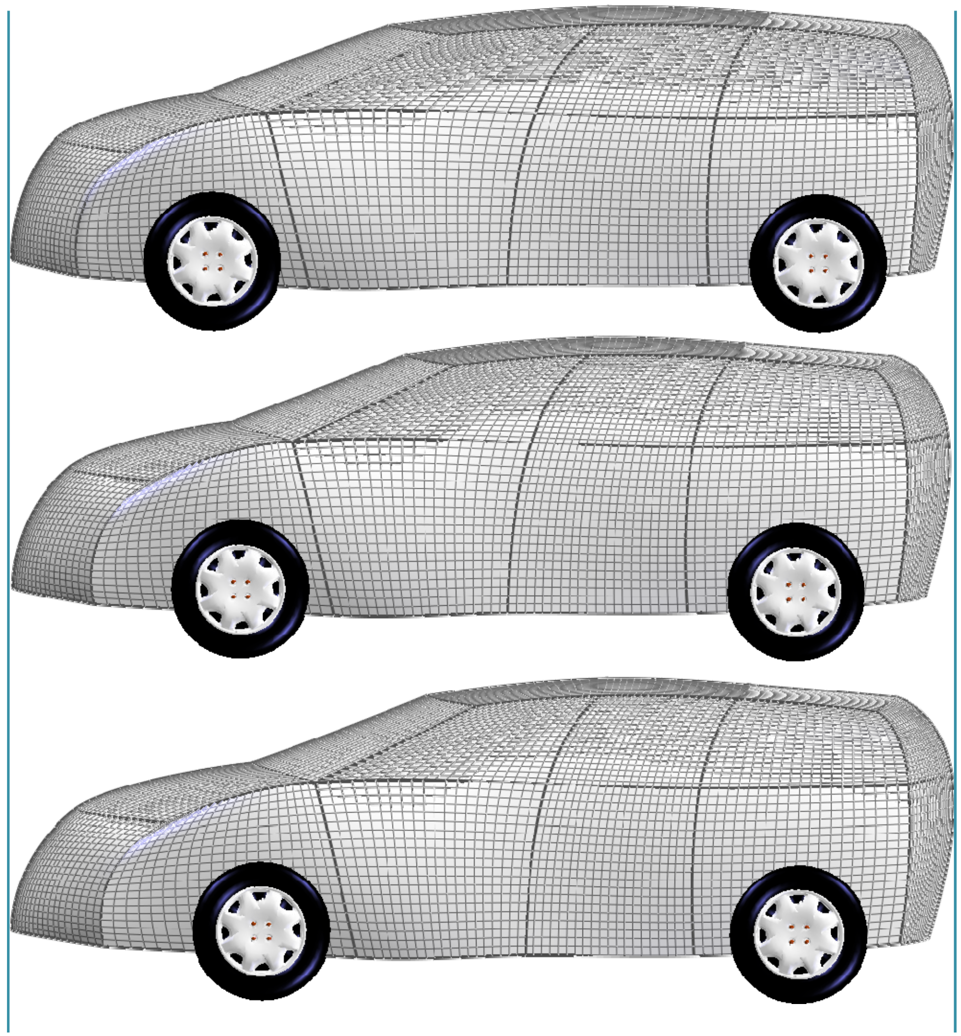
$$B_{new} = B_0(u_B)P_0 + B_1(u_B)P_1^* + B_2(u_B)P_2^* + B_3(u_B)P_3$$

$$\begin{bmatrix} B_1(u_A) & B_2(u_A) \\ B_1(u_B) & B_2(u_B) \end{bmatrix} \begin{bmatrix} P_1^* \\ P_2^* \end{bmatrix} = \begin{bmatrix} A_{new} - B_0(u_A)P_0 - B_3(u_A)P_3 \\ B_{new} - B_0(u_B)P_0 - B_3(u_B)P_3 \end{bmatrix}$$

  - We can analytically solve P_1^* and P_2^*
  - Using P_1^* and P_2^*, we apply a FFD [7] to the template on a 1x1x3 lattice structure

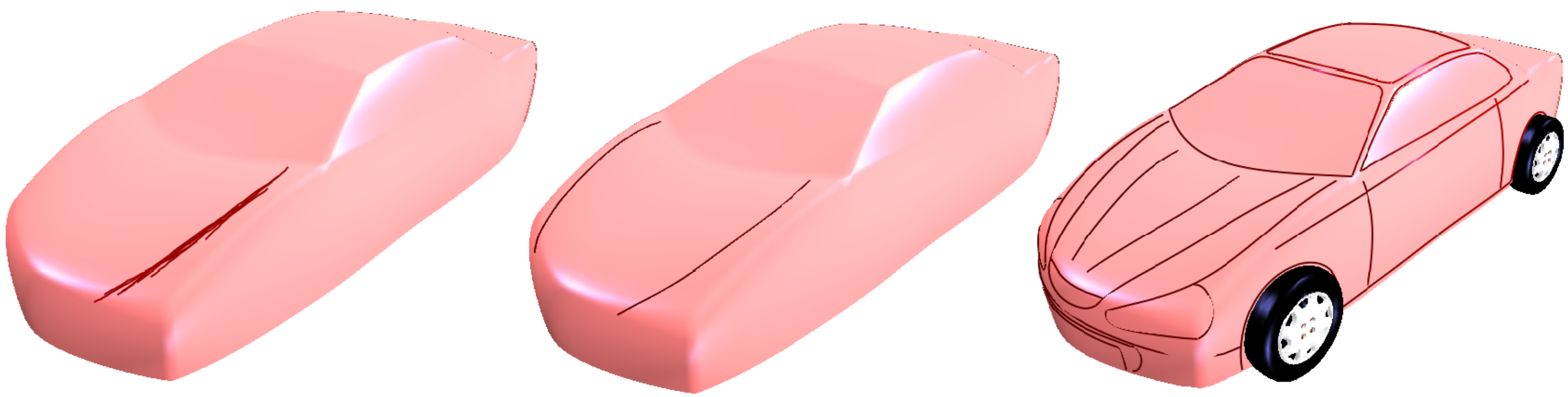
Post-Dimensioning

- Results in smooth deformations



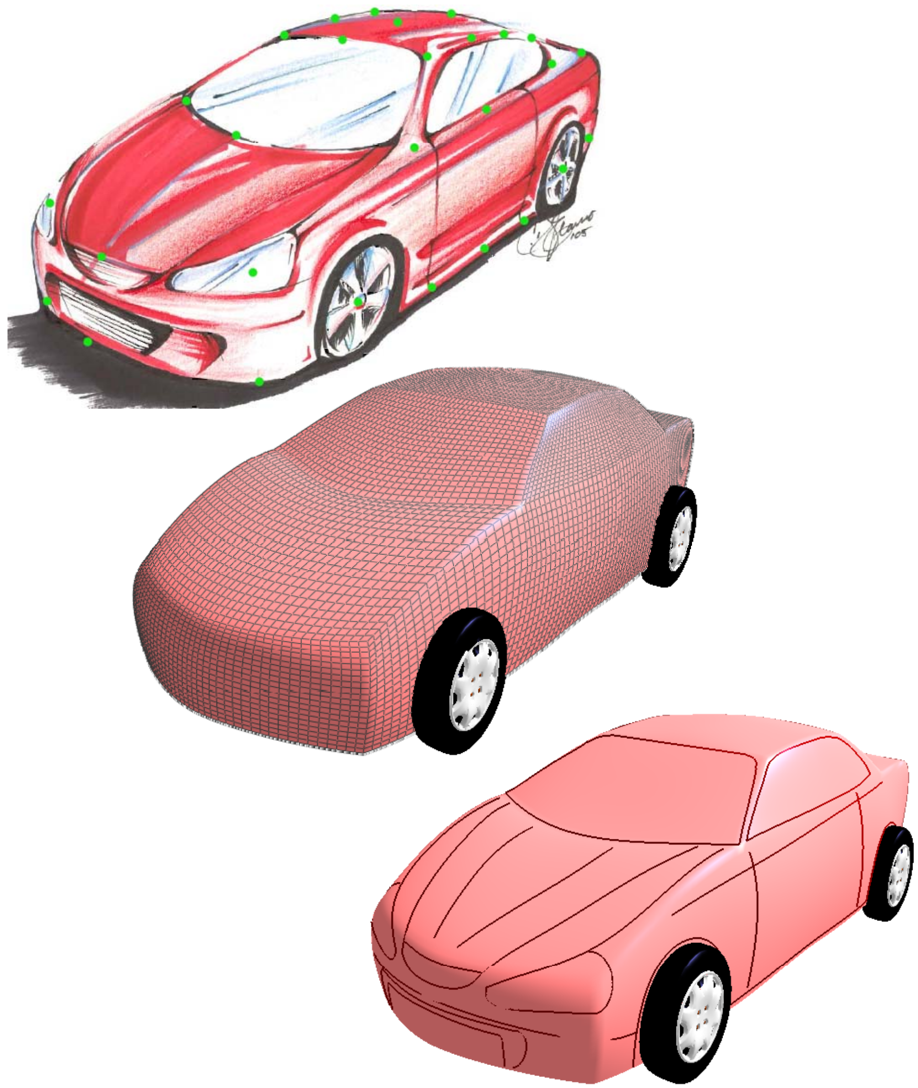
Styling

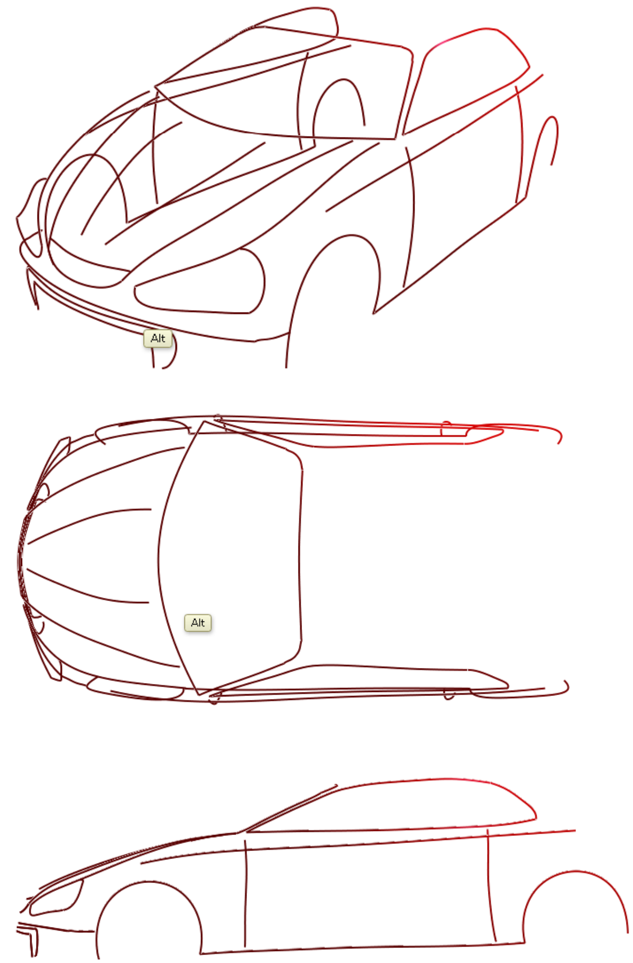
- After surface template is designed, the user can sketch on it



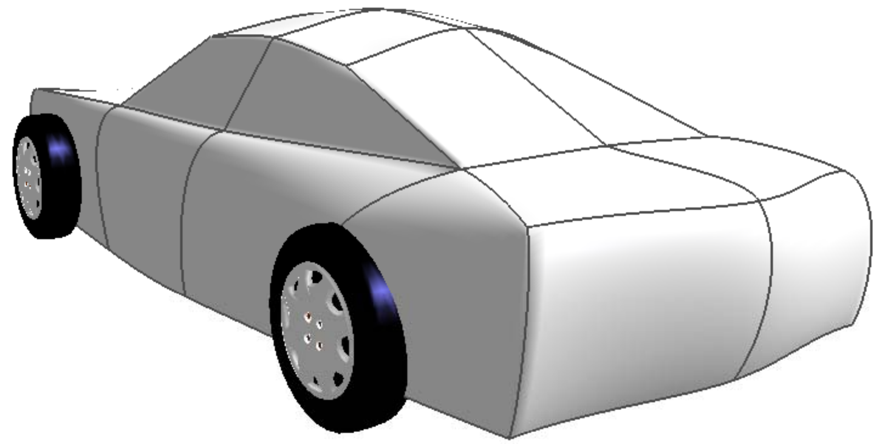
- Curves are first smoothed using Savitzky-Golay smoothing [6] in image plane, then projected onto template surface

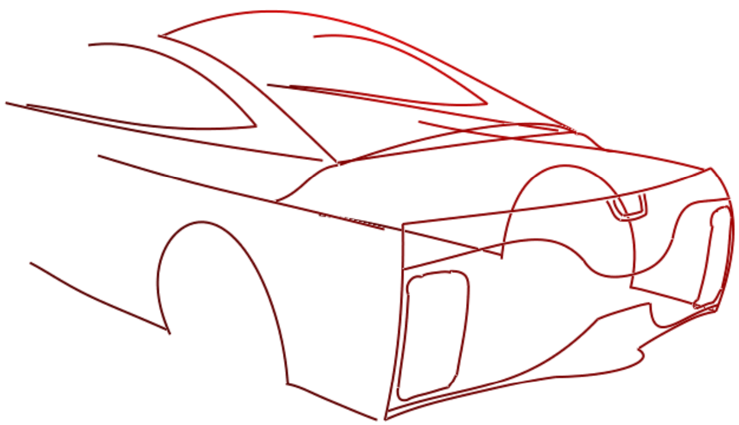
Results

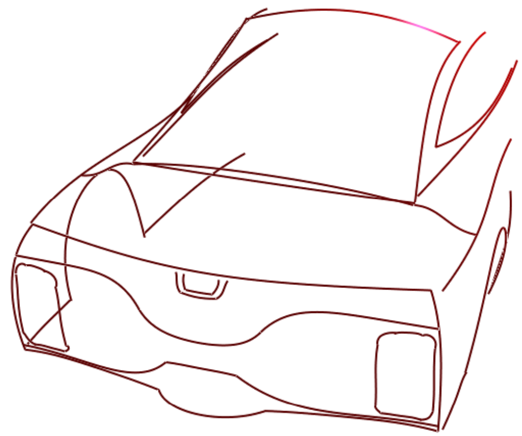




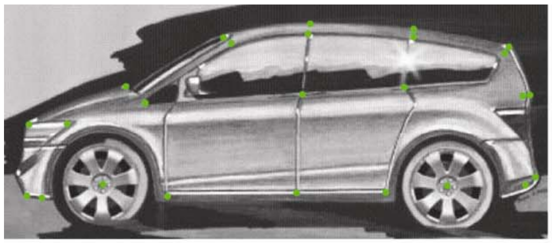
Results



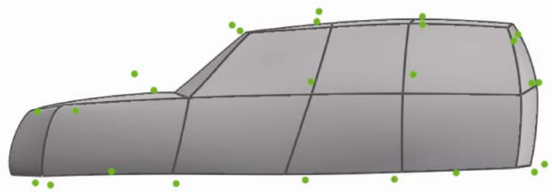




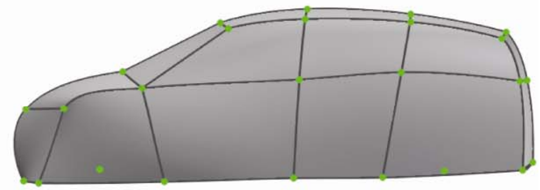
Results



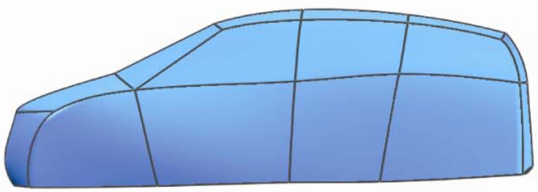
(a) Input sketch and marked fiducial points



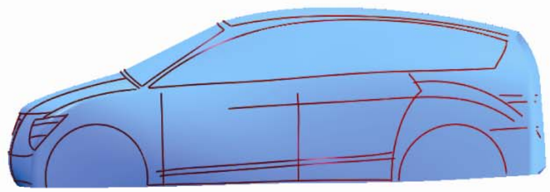
(b) Aligned template



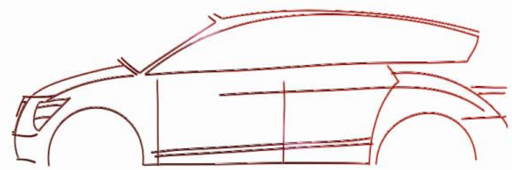
(c) Result immediately after deformation



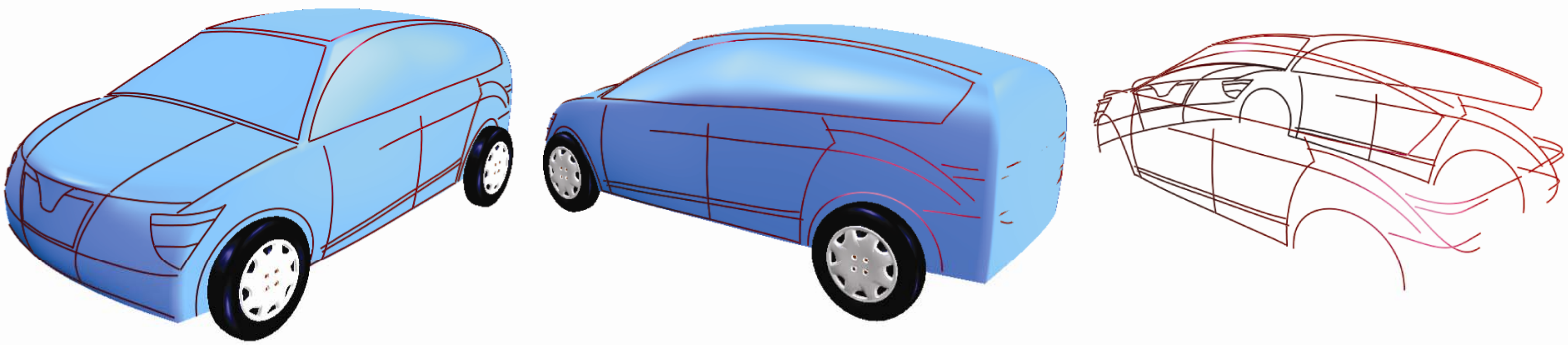
(d) Result after edge modification



(e) Styling curves drawn on the template

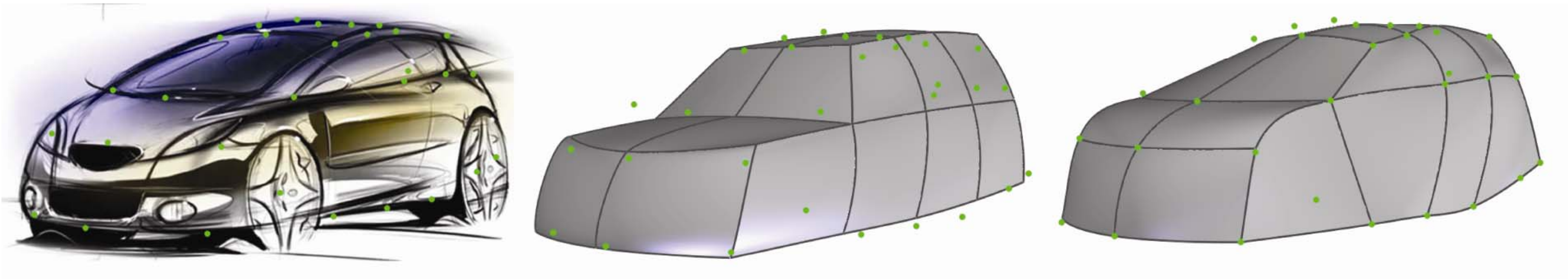


(f) Styling curves with template removed

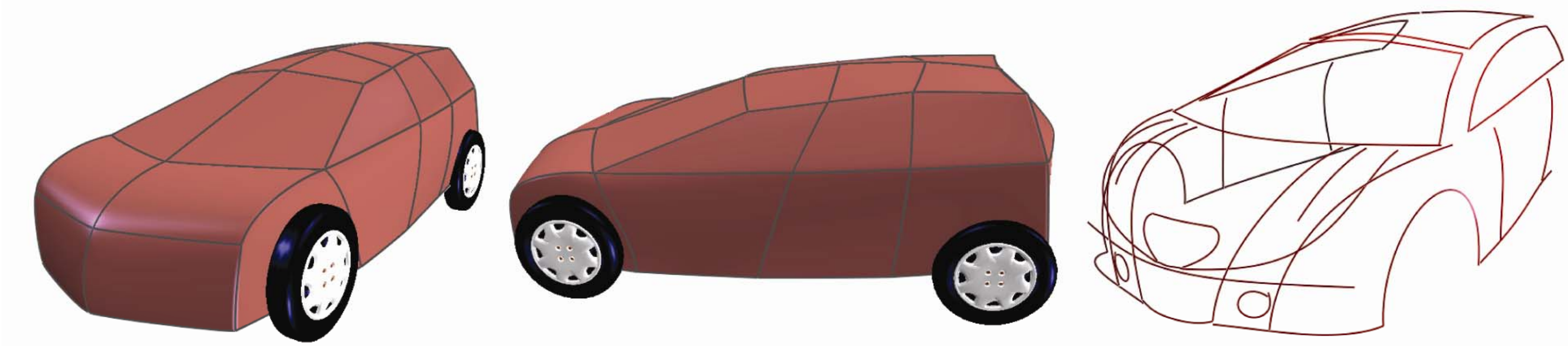


(g) Final results

Results

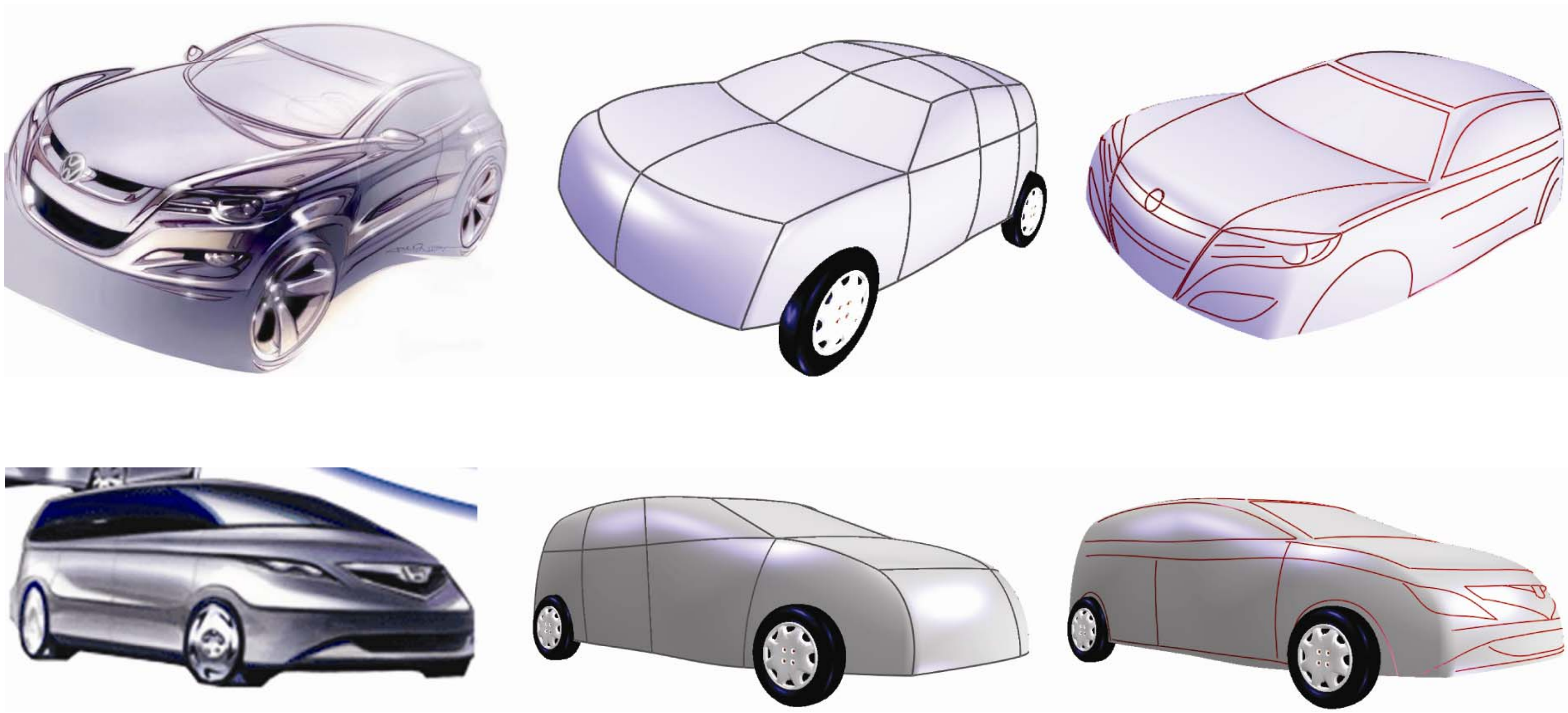


(a) Template alignment and deformation



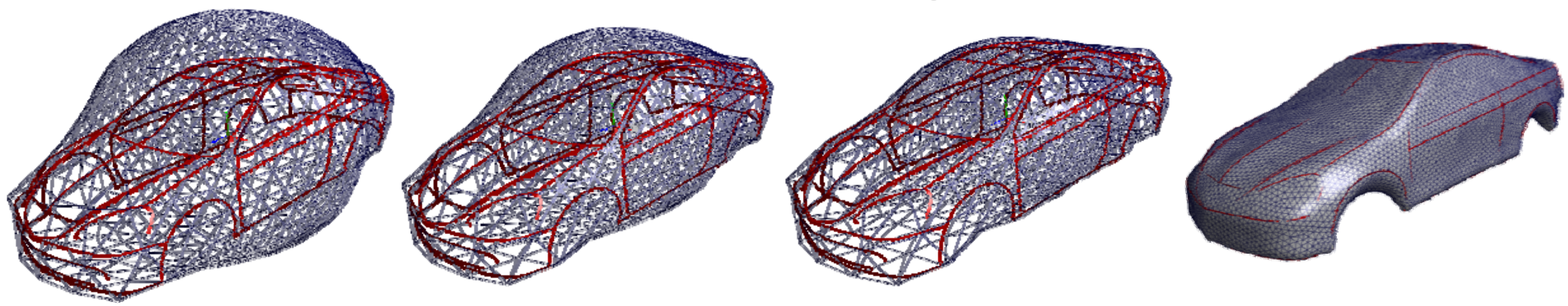
(b) Final results

Results

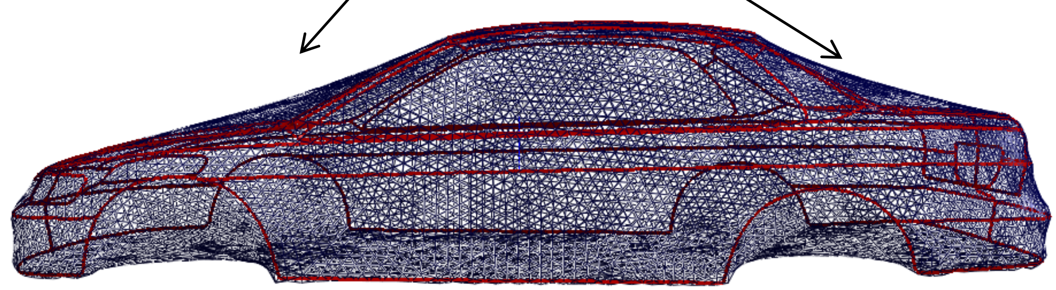


Ideas for Improving ShrinkWrap

- ShrinkWrap method wraps a sphere on a wireframe via shrinking and subdivision.

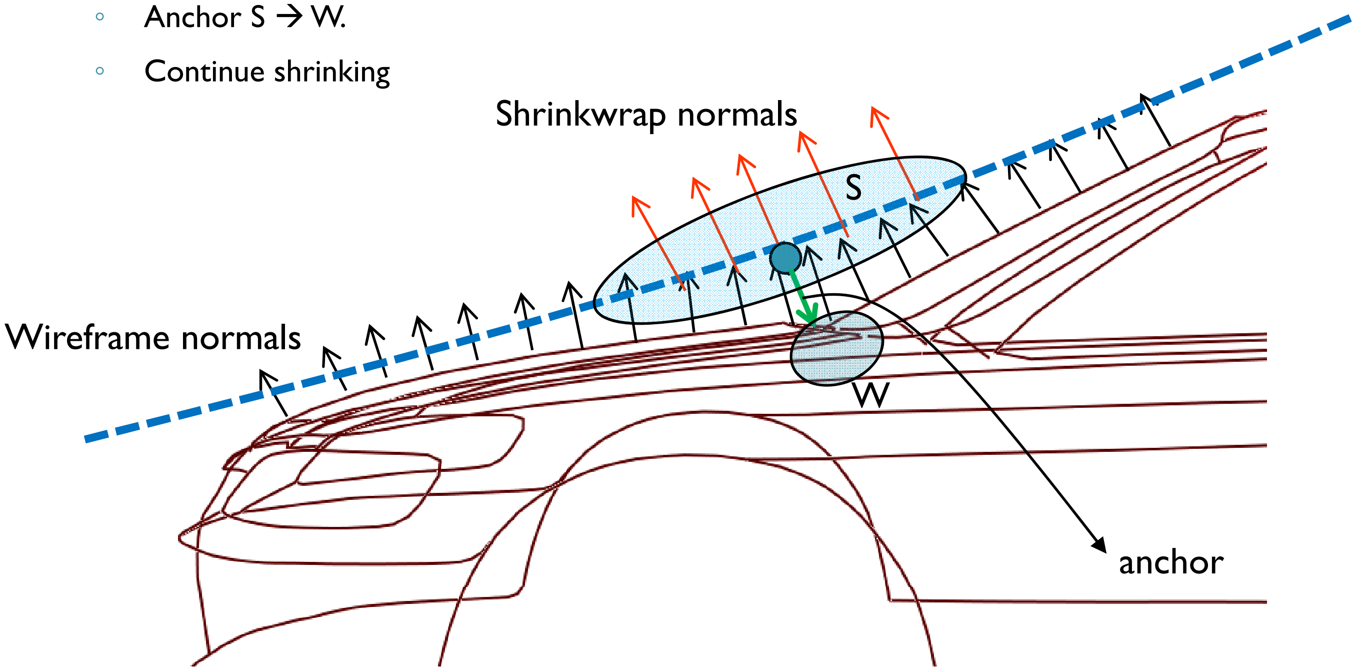


- Shrinkwrap has difficulty with concave regions.



Ideas for Improving ShrinkWrap

- Anchor shrinkwrap vertices to wireframe nodes
  - Detect shrinkwrap vertices S that are far from wireframe.
  - Detect wireframe vertices W that are far from S and have normal vectors similar to those in S.
  - Anchor S → W.
  - Continue shrinking



References

- [1] Levent Burak Kara, Chris D'Eramo, Kenji Shimada (2006) Pen-based Styling Design of 3D Geometry Using Concept Sketches and Template Models. ACM Solid and Physical Modeling Conference (SPM) 2006.
- [2] ABDEL-AZIZ, Y., AND KARARA, H. 1971. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In Symposium on Close-Range Photogrammetry, American Society of Photogrammetry, 1.18.
- [3] FORSYTH, D.A., AND PONCE, J. 2003. Computer Vision: a Modern Approach. Prentice Hall.
- [4] Mortenson, M. E. 1985 Geometric Modeling. John Wiley & Sons, Inc.
- [5] Farin, G. 2002 Curves and Surfaces for CAGD: a Practical Guide. 5th. Morgan Kaufmann Publishers Inc.
- [6] Levent Burak Kara, Kenji Shimada, Sarah D. Marmalefsky (2007). An Evaluation of User Experience with a Sketch-based 3D Modeling System. Computers & Graphics. Volume 31, Issue 4, August 2007, Pages 580-597
- [7] Sederberg, Thomas and Parry, Scott. "Free Form Deformation of Solid Geometric Models." SIGGRAPH, Association of Computing Machinery. Volume 20, Number 4, 1986. 151-159.