Simple Formulas for Quasiconformal Plane Deformations

by Yaron Lipman, Vladimir Kim, and Thomas Funkhouser

ACM TOG 2012

Stephen Mann
Planar Shape Deformations

Used in
- Mesh parameterization
- Animation shape interpolation
- others
Planar Shape Deformations

Used in
- Mesh parameterization
- Animation shape interpolation
- others

Issues
- Distort shape
- Bijective?
- Interpolation issues
Planar Shape Deformations

Used in
- Mesh parameterization
- Animation shape interpolation
- others

Issues
- Distort shape
- Bijective?
- Interpolation issues

Conformal maps
- Preserve angles everywhere
- Bounded distortion: *quasiconformal*
Main Ideas

- Map four points to four points (interpolation)

\[ f = m_z \circ A \circ m_{-1} \]
Main Ideas

- Map four points to four points (interpolation)
- Embed quads in circular grid
Main Ideas

- Map four points to four points (interpolation)
- Embed quads in circular grid
- Map circular grid to parallelograms
Main Ideas

- Map four points to four points (interpolation)
- Embed quads in circular grid
- Map circular grid to parallelograms
- Map parallelograms to each other

Map is \( f = m_z \circ A \circ m_w^{-1} \)
Circles Through Four Points

- Through 4 points, there exists unique 5th point $z_\infty$ such that
  - There exist four circles through consecutive points and $z_\infty$
  - “Opposite” circles osculate
  - $z_\infty$ is outside
Mobius Transformations

- Work in complex plane
- Construct a Mobius transformation to map “quad” to parallelogram

\[ m(z) = \frac{az + b}{cz + d}, \quad ad - bc \neq 0 \]

(basically, map circles to lines, \( z_\infty \) to \( \infty \))
Details

- Look for Mobius transform that maps to parallelogram and linear transformation from square to “same” parallelogram:

  \[ m_z(z_j) = L(g_j), \quad j = 1..4 \]

  where \( z_j \) are four points, \( g_j \) are corners of square
Look for Mobius transform that maps to parallelogram and linear transformation from square to “same” parallelogram:

\[ m_z(z_j) = L(g_j), \quad j = 1..4 \]

where \( z_j \) are four points, \( g_j \) are corners of square

Expand:

\[ \frac{az_j + b}{cz_j + d} = g_j + \ell \overline{g_j}, \quad j = 1..4 \]
Details

- Look for Mobius transform that maps to parallelogram and linear transformation from square to “same” parallelogram:

$$m_z(z_j) = L(g_j), \quad j = 1..4$$

where $z_j$ are four points, $g_j$ are corners of square

- Expand:

$$\frac{az_j + b}{cz_j + d} = g_j + \ell g_j, \quad j = 1..4$$

Multiply both sides by $cz_j + d$ gives 4 nonlinear equations in 5 unknowns $(a, b, c, d, \ell)$

$$[Z|1| - ZG| - G| - Z\overline{G}| - \overline{G}] (a, b, c, d, \ell c, \ell d)^T = 0$$
Look for Mobius transform that maps to parallelogram and linear transformation from square to “same” parallelogram:

\[ m_z(z_j) = L(g_j), \ j = 1..4 \]

where \( z_j \) are four points, \( g_j \) are corners of square

Expand:

\[
\frac{az_j + b}{cz_j + d} = g_j + \ell g_j, \ j = 1..4
\]

Multiply both sides by \( cz_j + d \) gives 4 nonlinear equations in 5 unknowns \((a, b, c, d, \ell)\)

\[
\begin{bmatrix}
Z|1| - ZG|1 - G| - Z\overline{G} - \overline{G}
\end{bmatrix}
(a, b, c, d, \ell c, \ell d)^T = 0
\]

Solve using SVD and roots of degree 2 equation
Implementation

- Implemented in Octave (SVD, complex numbers, bad documentation)
Implementation

- Implemented in Octave (SVD, complex numbers, bad documentation)
  About 45 lines of code (plus testing, application code)
Implementation

- Implemented in Octave (SVD, complex numbers, bad documentation)
- About 45 lines of code (plus testing, application code)
- 4–8 hours work (errors, vagueness in paper, Octave struggles)
Implementation

- Implemented in Octave (SVD, complex numbers, bad documentation)
  About 45 lines of code (plus testing, application code)
  4–8 hours work (errors, vagueness in paper, Octave struggles)
- Simple examples seem okay...
Implementation

- Implemented in Octave (SVD, complex numbers, bad documentation)
  About 45 lines of code (plus testing, application code)
  4–8 hours work (errors, vagueness in paper, Octave struggles)

- Simple examples seem okay...
Implementation

- Implemented in Octave (SVD, complex numbers, bad documentation)
  About 45 lines of code (plus testing, application code)
  4–8 hours work (errors, vagueness in paper, Octave struggles)
- Simple examples seem okay...
Circle Examples

Circle examples “mixed” (red: invalid data)
Restrictions on Deformation

- Paper mentions two restrictions:
  - \( ad - bc \neq 0 \)
  - Counter-clockwise order

- Does not discuss boundary cases
  Does not discuss how restrictive conditions are

- Paper is big on "circle not crossing" using their method... but limits on where you can move points.
  Was circle a strawman example?
Restrictions on Deformation

- Paper mentions two restrictions:
  - \( ad - bc \neq 0 \)
  - Counter-clockwise order
- Does not discuss boundary cases
  Does not discuss how restrictive conditions are
- Invalid circle examples give some feel
Restrictions on Deformation

- Paper mentions two restrictions:
  - $ad - bc \neq 0$
  - Counter-clockwise order
- Does not discuss boundary cases
  - Does not discuss how restrictive conditions are
- Invalid circle examples give some feel

- Paper is big on “circle not crossing” using their method...
Restrictions on Deformation

- Paper mentions two restrictions:
  - $ad - bc \neq 0$
  - Counter-clockwise order

- Does not discuss boundary cases
  Does not discuss how restrictive conditions are

- Invalid circle examples give some feel

- Paper is big on “circle not crossing” using their method...
  ...but limits on where you can move points.
Restrictions on Deformation

- Paper mentions two restrictions:
  - $ad - bc \neq 0$
  - Counter-clockwise order
- Does not discuss boundary cases
  - Does not discuss how restrictive conditions are
- Invalid circle examples give some feel

- Paper is big on “circle not crossing” using their method...
  - ...but limits on where you can move points.
  - And is “not crossing” good? Overly restrictive?
  - Was circle a strawman example?
Local Version

- Deformation is global
  Very large distortion outside of quad
Local Version

- Deformation is global
  Very large distortion outside of quad
- Authors developed a local version
  - Restricted operations
    - Move opposite edges (i.e., to get a bend)
    - Move one point (two edges left unchanged)
Local Version

- Deformation is global
  - Very large distortion outside of quad
- Authors developed a local version
  - Restricted operations
    - Move opposite edges (i.e., to get a bend)
    - Move one point (two edges left unchanged)
- Used variation on method that leaves solid edges “unchanged”
  (in $f = m_z \circ A \circ m_w^{-1}$, use different ’A’; higher distortion)
- More engineering than math
Details

Working with images of Mobius functions (parallelograms):

- Sample points on fixed edges
  Also sample $\delta$ inside fixed edges
Details

Working with images of Mobius functions (parallelograms):

- Sample points on fixed edges
  Also sample $\delta$ inside fixed edges
- Map points to parallelograms
Details

Working with images of Mobius functions (parallelograms):

- Sample points on fixed edges
  Also sample $\delta$ inside fixed edges
- Map points to parallelograms
- Use thin-plate splines to construct transformation to map “z” points onto “w” points

\[
\phi(z) = \sum_j b_j \phi(|z - c_j|) + A(z)
\]

\[
\phi(c_j) = d_j
\]
How To Solve \( \varphi(z) = \sum_j b_j \phi(|z - c_j|) + A(z) \)

- Paper says “done in a standard way” and cites a book
How To Solve $\varphi(z) = \sum_j b_j \phi(|z - c_j|) + A(z)$

- Paper says “done in a standard way” and cites a book
- I tried my own way (we have $A$, just solve $\sum$ part as linear system)
How To Solve \( \varphi(z) = \sum_j b_j \phi(|z - c_j|) + A(z) \)

- Paper says “done in a standard way” and cites a book
- I tried my own way (we have \( A \), just solve \( \sum \) part as linear system)
How To Solve \( \varphi(z) = \sum_j b_j \phi(|z - c_j|) + A(z) \)

- Paper says “done in a standard way” and cites a book
- I tried my own way (we have \( A \), just solve \( \sum \) part as linear system)
- Looked up map in book: it is standard. And brief:

\[
\begin{bmatrix}
\phi(|c_i - c_j|) & P(c_j) \\
P(c_j)^T & 0
\end{bmatrix}
\begin{bmatrix}
b_j \\
A
\end{bmatrix}
= 
\begin{bmatrix}
d_j \\
0
\end{bmatrix}
\]

(Last page of paper 1/3 blank)
How To Solve $\varphi(z) = \sum_j b_j \phi(|z - c_j|) + A(z)$

- Paper says “done in a standard way” and cites a book
- I tried my own way (we have $A$, just solve $\sum$ part as linear system)
- Looked up map in book: it is standard. And brief:

$$
\begin{bmatrix}
\phi(|c_i - c_j|) & P(c_j) \\
\left(P(c_j)^T\right)^{-1} & 0
\end{bmatrix}
\begin{bmatrix}
b_j \\
A
\end{bmatrix} =
\begin{bmatrix}
d_j \\
0
\end{bmatrix}
$$

(Last page of paper 1/3 blank)
Example
Example
Maps for Example

(upside down, reverse order, inorder)
Another Example
Another Example
Map for Example
Other Radial Basis Functions?

- Thin plate splines: minimum solution of distance to affine map?
- Tried $r^3$ and $1/(r^2 + \epsilon)$

Conformal

Thin Plate

$r^2 \log r$

$r^3$

Thin plate is less deformed in some places
Radial basis function: $1/(r^2 + \epsilon)$

Thin Plate
$r^2 \log r$

$1/(r^2 + 0.5)$

$1/(r^2 + 0.05)$

$\frac{1}{r^2 + 0.05}$ clearly bad; thin plate probably better than $\frac{1}{r^2 + 0.5}$
Effect of $n$

- Method to reproduce boundaries only approximates
  $\phi(z) = \sum_{j=1}^{n} b_j \phi(|z - c_j|) + A(z)$
- They use $n = 10$. What is effect of $n$?
- On caterpillar example:

\[ n = 3 \]

\[ n = 5 \]

\[ n = 10 \]
Normal Points

- Method uses normal points to interpolate cross boundary derivatives
- Ran caterpillar example with and without normal points:

![Graphs showing normal points, no normal points, and both](image-url)
Which Normal Points?

Paper uses

\[ \tilde{p} = p + n\delta \left[ |z_\alpha - z_{\beta+1}| |p - z_{\alpha+1}| + |z_\beta - z_{\alpha+1}| |p - z_\alpha| \right] \]

(which is incorrect: need to normalized by by \( |z_{\alpha+1} - z_\alpha| \))
Which Normal Points?

- Paper uses

\[ \tilde{p} = p + n\delta \left[ |z_\alpha - z_{\beta+1}| |p - z_{\alpha+1}| + |z_\beta - z_{\alpha+1}| |p - z_{\alpha}| \right] \]

(which is incorrect: need to normalized by by \( |z_{\alpha+1} - z_{\alpha}| \))

- What if we use simpler

\[ \tilde{p} = (1 - \delta)p + \delta\tilde{p} \]
Perpendicular vs Affine
Perpendicular vs Affine Maps

Perpendicular

Affine
Delta

Paper chose $\delta = 0.01$

$\delta = 0.1$

$\delta = 0.01$

$\delta = 0.001$

Need to test with regular textures, etc., to decide what value best (but 0.1 looks bad)
Counter-clockwise

- Paper says “ordered in counter-clockwise fashion (different order will lead to a different map).” How different?
Counter-clockwise

- Paper says “ordered in counter-clockwise fashion (different order will lead to a different map).”

How different?
Counter-clockwise

- Paper says “ordered in counter-clockwise fashion (different order will lead to a different map).”

How different?
What matters?

Out of 3 stars:

*** Perpendicular vs Affine normal points†
*** Counter-clockwise vs clockwise
** Value of $n$†
* Thin plate vs other radial basis
* Normal points†
* Value of delta†
* Method to solve for thin plate coefficients

†: clearly needs more investigation
Code

- Line count
  - Global method: 45 lines of code
  - Local method: 142 (+25) lines of code (+320)
  - Testing, Applications: 751 lines of code (but...)

Really? It's Octave...
Code

- Line count
  - Global method: 45 lines of code
  - Local method: 142 (+25) lines of code (+320)
  - Testing, Applications: 751 lines of code (but...)

- Speed
Code

- Line count
  - Global method: 45 lines of code
  - Local method: 142 (+25) lines of code (+320)
  - Testing, Applications: 751 lines of code (but...)

- Speed
  Really? It’s Octave...
Extension: Scaling

In their method, distance between white points connected with solids lines is fixed.
Extension: Scaling

For perspective, might want scaling
No scaling
Extension: Scaling

For perspective, might want scaling

Scaling
Analysis

Method seems to do reasonable job. Why?

- Low deformation
- Lines map to circles \(\Rightarrow\) avoids kinks
Method seems to do reasonable job. Why?

- Low deformation
- Lines map to circles \(\Rightarrow\) avoids kinks
- Comparison to other methods unfair?
  Doesn’t map quads interiors to quad interiors
Can you break it?

Yes, but...
Can you break it?

Yes, but...

(pixel dropout due to...)
Can you break it?

Yes, but...

(pixel dropout due to...?)
Straight Sections

Found several cases where map is “straight” and bend concentrated
Conclusions

+ Slick global method
  Nice math, easy to implement

− Restrictions on four points not discussed
− Circle editing seems oversold
− Cherry picked examples?
− Geometric algebra reformulation?
Conclusions

+ Slick global method
  Nice math, easy to implement
+ Local method
  Doesn’t have guarantees of global method
  Not so simple

- Restrictions on four points not discussed
- Circle editing seems oversold
- Cherry picked examples?
- Geometric algebra reformulation?
Conclusions

+ Slick global method
  Nice math, easy to implement
+ Local method
  Doesn’t have guarantees of global method
  Not so simple
  But seems reasonable (straight sections?)
Conclusions

+ Slick global method
  Nice math, easy to implement
+ Local method
  Doesn’t have guarantees of global method
  Not so simple
  But seems reasonable (straight sections?)
  - Restrictions on four points not discussed
  - Circle editing seems oversold
  - Cherry picked examples?

Geometric algebra reformulation?
Conclusions

+ Slick global method
  Nice math, easy to implement
+ Local method
  Doesn’t have guarantees of global method
  Not so simple
  But seems reasonable (straight sections?)
− Restrictions on four points not discussed
− Circle editing seems oversold
− Cherry picked examples?

➤ Geometric algebra reformulation?
Several of the figures in this talk were based on figures in the Lipman, Kim, Funkhouser paper.