A New Algorithm of Polar Decomposition

Hiroyasu Hamada
h-hamada@sasebo.ac.jp
National Institute of Technology, Sasebo College

Kenta Toyofuku
ma213030@math.Kyushu-u.ac.jp
Kyushu University

Shizuo Kaji
skaji@yamaguchi-u.ac.jp
Yamaguchi University

Abstract

The polar decomposition expresses a linear transformation as a composition of rotation and anisotropic scaling (or stretch). It serves as a fundamental tool [6] in animation and deformation techniques such as ARAP [1], Shape matching [4], Poisson mesh editing [6] and so on. In this poster we propose a new algorithm which computes the polar decomposition for 3x3 matrices with positive determinants. Our algorithm is faster than existing methods in some cases.

Polar Decomposition

- For a 3x3 matrix A, there is a unique factorization
  $A = RS$ (R: rotation, S: symmetric) .
  This is called the polar decomposition of A.
- R is shown to be the "closest" rotation to A.

Interpolation of linear transformation

A most typical scenario is :
Given linear transformations A_0 and A_1, compute their interpolation.
This is achieved by

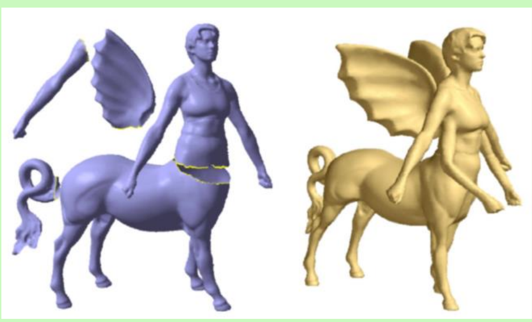
$$A_t = \mathrm{SLERP}(R_0, R_1, t)\,\{(1-t)\,S_0 + t\,S_1\}$$

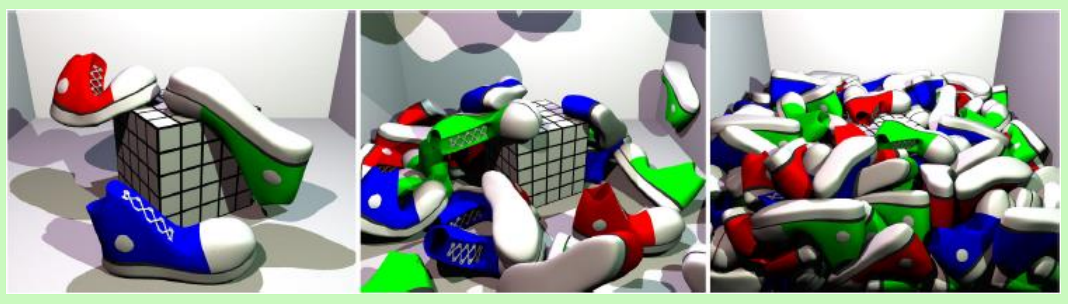
where $A_0 = R_0\,S_0$ and $A_1 = R_1\,S_1$.

This is used in ARAP [1].

In Poisson mesh editing [6], a slight generalisation of this is used to blend a number of transformations.

The polar decomposition is also used in Shape matching [4] to determine the rigid transformation which approximates a given deformation the best.





Existing algorithm

SVD (Singular Value Decomposition)

Suggested in [1] and others. SVD is expensive.

- Compute the singular value decomposition of A:
  $A = UDV$
- $R = UV$, $S = V^T DU$

Diagonalization

- $B = A^T A$
- Compute diagonalization of $B = UDU^T$.
- $S = U\sqrt{D}U^T$, $R = A\,S^{-1}$

Higham's algorithm [2]

Most popular in CG [5]. Involves only matrix multiplication and inverse but needs iteration.

Input: A, small number $\varepsilon > 0$ Output: R, S

```
C ← A
do {
    D ← (C^{-1})^T
    c ← ||C||_1 ||C||_∞
    d ← ||D||_1 ||D||_∞
    γ ← (d/c)^{1/4}
    P ← C
    C ← 0.5 γ C + (0.5/γ) D
} while ( ||C − P||_1 > ||P||_1 ε )
R ← C
S ← C^T A
```

Our Algorithm

- $B = A^T A$
  $C = \log B$
  $S = \exp(C/2)$, $R = A\exp(-C/2)$
- Note that there are quadratic formulae to compute log and exp using spectral decomposition. This involves computation of trigonometric functions, exponentials and logarithms of real numbers.
- An implementation is available in [3].

Discussion and Conclusion

We did timing comparison for the four different methods; SVD, diagonalization, Higham's, and ours in different environments.
In general, we observed that the performance was

diagonalization ≒ ours > Higham's > SVD .

Remarks

- Since our matrix is three dimensional, eigenvalues and eigenvectors can be calculated by a closed formula, which does not extend to higher dimensional matrices. If we use QR algorithm, the performance comparison becomes

  diagonalization ≒ ours ≒ Higham's > SVD .

  .
- Our algorithm involves computation of exponential and logarithm of real numbers. The computational cost depends heavily on the library for those functions.
- Both diagonalization and our method may have numerical problem with near-singular matrices.

Suggestion

- If there is a possibility of near-singular matrices in your problem, use Higham's.
- If your system does not provide a good maths library, use diagonalisation.
- Otherwise, try our method!

References

[1] M. Alexa, D. Cohen-Or and D. Levin, *As-rigid-as-possible shape interpolation*, In Proceedings of SIGGRAPH '00, pp. 157–164, 2000.
[2] N. J. Higham, *Computing the polar decomposition --- with application*, SIAM J. Sci. Stat. Comput., 7, pp. 1160-1174, 1986.
[3] S. Kaji, A C++ library for 3D affine transformation, https://github.com/shizuo-kaji/AffineLib, 2014.
[4] M. Müller, B. Heidelberger, M. Teschner and M. Gross, *Meshless deformations based on shape matching,* In Proceedings of SIGGRAPH '05, *pp. 471-478,* 2005.
[5] K. Shoemake, *Polar Matrix Decomposition*, Graphic Gems IV.
[6] Y. Yu, K. Zhou, X. Shi, H. Bao, B. Guo and H.-Y. Shum, *Mesh editing with Poisson-based gradient field manipulation,* ACM Trans. Graphics, 23, pp. 644-651, 2004.

Acknowledgements

This work was supported by Core Research for Evolutional Science and Technology (CREST) Program "Mathematics for Computer Graphics" of Japan Science and Technology Agency (JST) . The authors are grateful to H. Ochiai, Y. Mizoguchi and K. Matsushita at Kyushu University for their valuable comments.