Efficient Computation of Constrained Parameterizations on Parallel Platforms

Th. Athanasiadis, G. Zioupos and I. Fudos

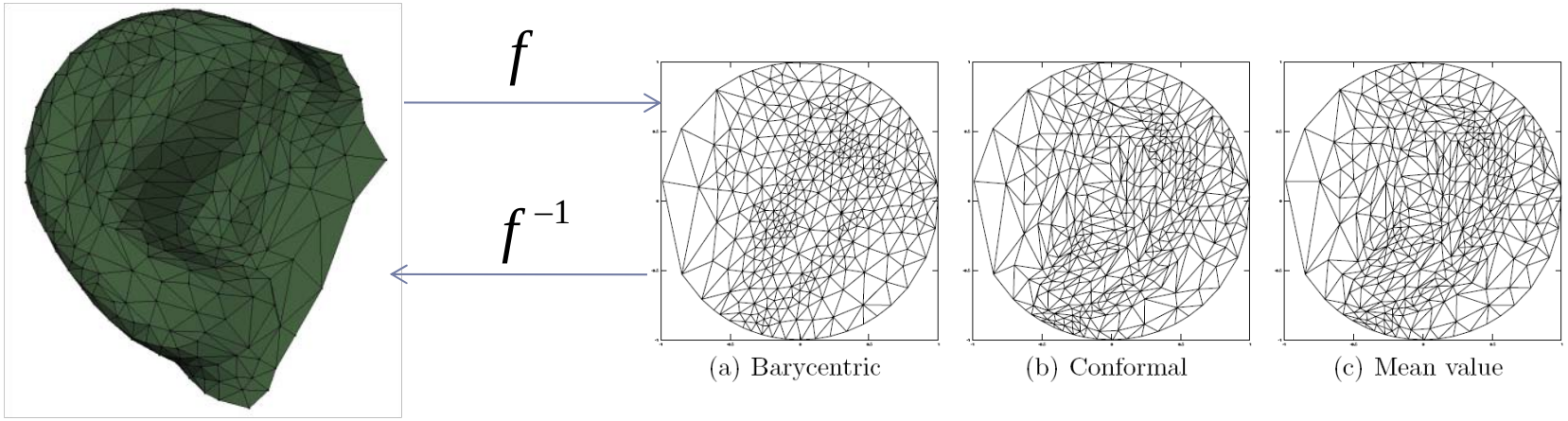
Motivation

- Mesh parameterization is a powerful geometry processing tool

- Applications
  - Remeshing
  - Texture mapping
  - Segmentation
  - Shape search
  - Morphing
  - Other …

- **Conclusion**: fast and robust mesh parameterization is central to many applications

Parameterization Definitions



- Surface $S \in \mathbb{R}^3$
- Parameter domain $\Omega \in \mathbb{R}^2$
- Bijective Mapping $f : S \to \Omega$ and $f^{-1} : \Omega \to S$

(one-to-one correspondence between Ω and S)

Properties of Parameterization

- A parameterization may introduce distortion
- Maps are characterized by their distortion:
  - Minimizing angle distortion : **conformal**
  - Minimizing area distortion : **authalic**
  - Minimizing distance distortion : **isometric**

Research Highlights – Contributions

- A simplified formulation for the isometric parameterization problem
- An efficient non linear solver using parallel platforms (GPU, Multicore cpus)
- An iterative untangling algorithm for computing initial solution for the non linear solver
- A cut-and-paste application using the parallel solver

Parameterization Definitions

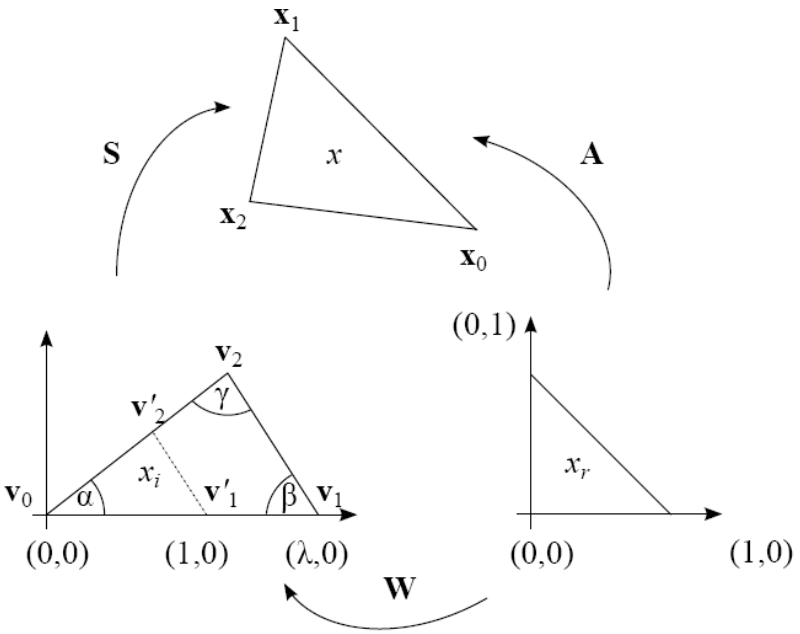
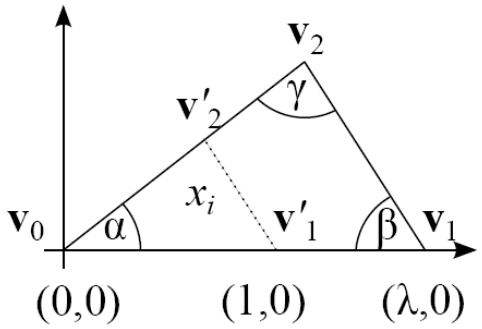


Figure 1: Ideal triangle $\triangle v_0 v_1 v_2$ and its similar triangle $\triangle v'_0 v'_1 v'_2$ on $\mathbb{R}^2$ along with the corresponding mappings.

- **W** : Edge matrix (Jacobian) of ideal triangle
- **A** : Edge matrix of the physical triangle
- $\mathbf{S} = \mathbf{A}\mathbf{W}^{-1}$

Isometric Parameterization Formulation



$$\eta_{shape^1} = \frac{\|\mathbf{S}\|_{\mathbf{F}}^2}{\det(\mathbf{S})}$$

$$\eta_{area} = \det(\mathbf{S}) + \frac{1}{\det \mathbf{S}}$$

$$\eta_{isometric} = \eta_{shape} \cdot \eta_{area} = \|\mathbf{S}\| + \frac{\|\mathbf{S}\|}{\det(\mathbf{S})^2}$$

Notes

(Equivalent to MIPS[2])

$$\eta_{shape} = \frac{1}{2} \frac{a^2 \cot\hat{\alpha} + b^2 \cot\hat{\beta} + c^2 \cot\hat{\gamma}}{\det(\mathbf{A})}$$

(Scale of ideal for η_{area})

$$\frac{1}{\lambda} = \sqrt{2E(\cot\hat{\alpha} + \cot\hat{\beta})}$$

(*E* is the area on the original mesh)

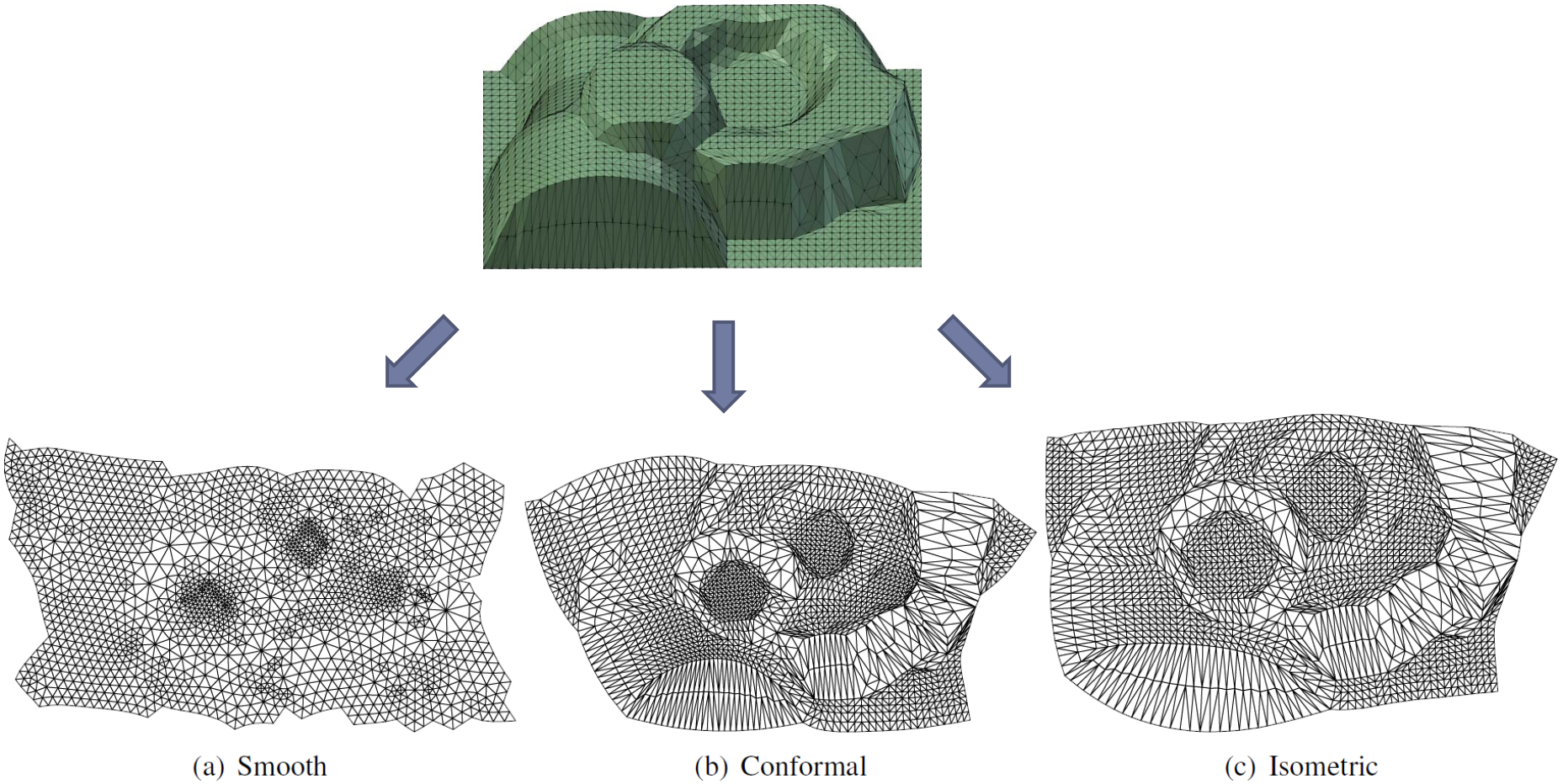
[1] P. M. Knupp, Algebraic mesh quality metrics, SIAM J. Sci. Comput. 23 (1) (2001) 193–218.
[2] K. Hormann, G. Greiner, MIPS: An efficient global parametrization method, in: P.-J. Laurent, P. Sablonniere, L. L. Schumaker(Eds.), Curve and Surface Design: Saint-Malo 1999, Innovations in Applied Mathematics, Vanderbilt University Press, Nashville, TN, 2000, pp. 153–162.

Non Linear Solver

- A system of non linear equations is formed

- By using the right non linear equations for each triangle we can control different properties of the final map
  - Conformal
  - Isometric
  - Smoothing (equal angles)

Defining Different Ideal Triangles



(a) Smooth (b) Conformal (c) Isometric

Constraints

- Hard constraints
  - Can be added in the system but an initial unfolded parameterization is required

- Soft constraints
  - Can be trivially supported, the metrics prevent folding due to the barrier form ($\eta_{shape} \rightarrow \infty$). Therefore, we can add a quadratic term in the non linear equations

Our approach

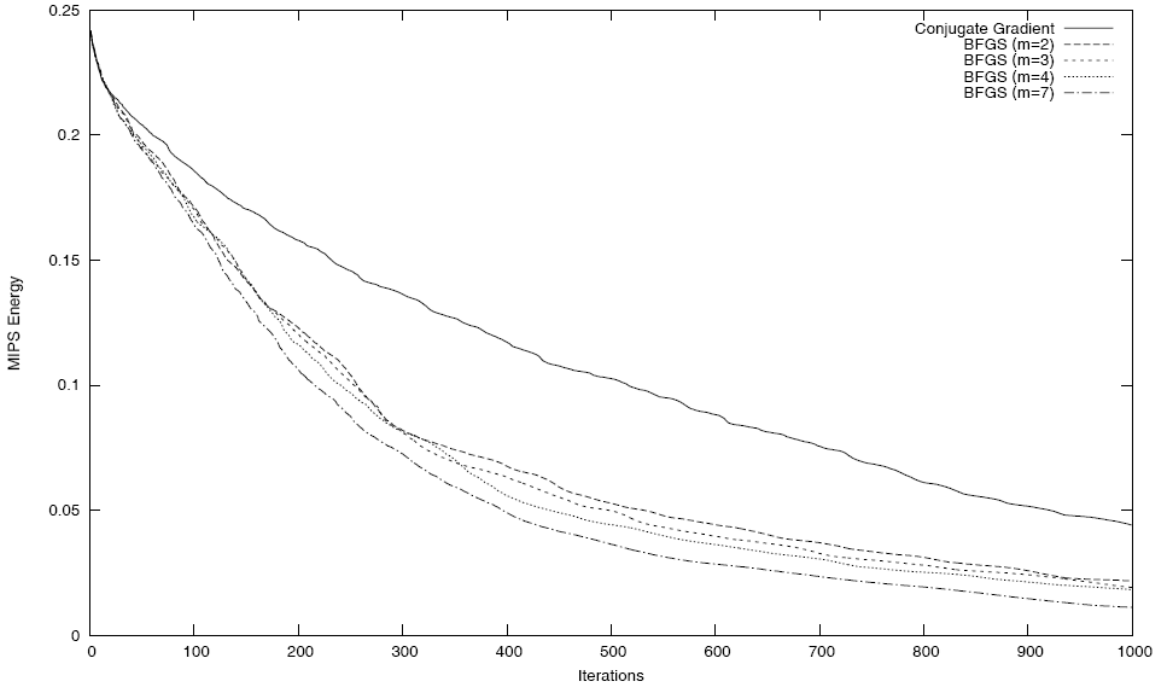
- Constrained parameterization
  - Compute an initial solution with a linear parameterization method
  - Fix the boundaries and the internal constraints
  - Untangle the mesh
  - Proceed as in the unconstrained case

- Unconstrained case (free or fixed boundary):
  - Set the proper non linear equations equation for smoothing, conformal or isometric mapping
  - Solve the non linear problem with limited memory BFGS (L-BFGS)

L-BFGS Benefits

- Faster convergence than Conjugate Gradient
- Low memory requirement (Configurable)
- No hessian computation is required
- Simple implementation only blas-1 operations are required
- Highly parallel suitable for implementation on the GPU

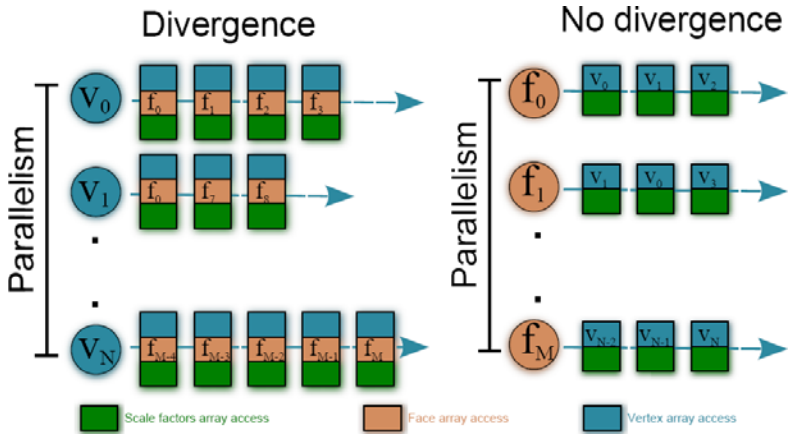
L-BFGS Convergence



- A few vectors of previous solutions (history) are hold for building a hessian approximation.
- The speed of convergence depends on the number of vectors of history retained
- A few vectors (3-7) are sufficient for good performance

GPU Implementation of the Solver

- OpenCL 1.1 used
- High accuracy and blas-1operations on GPU (important for BFGS)
- A lot of memory optimizations on derivative and function computation
  - Coalesced memory access
  - Minimized branch divergence

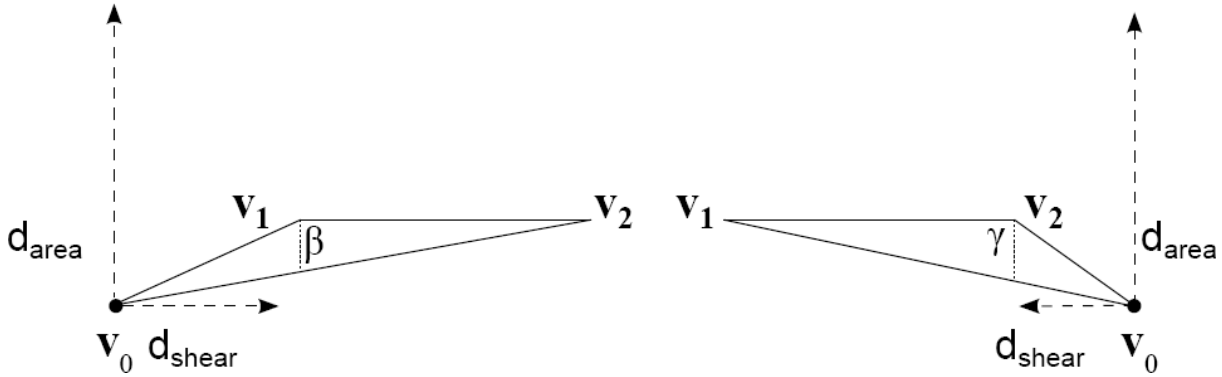


Untangling extensions

- Directly minimize the sum areas of all inverted elements

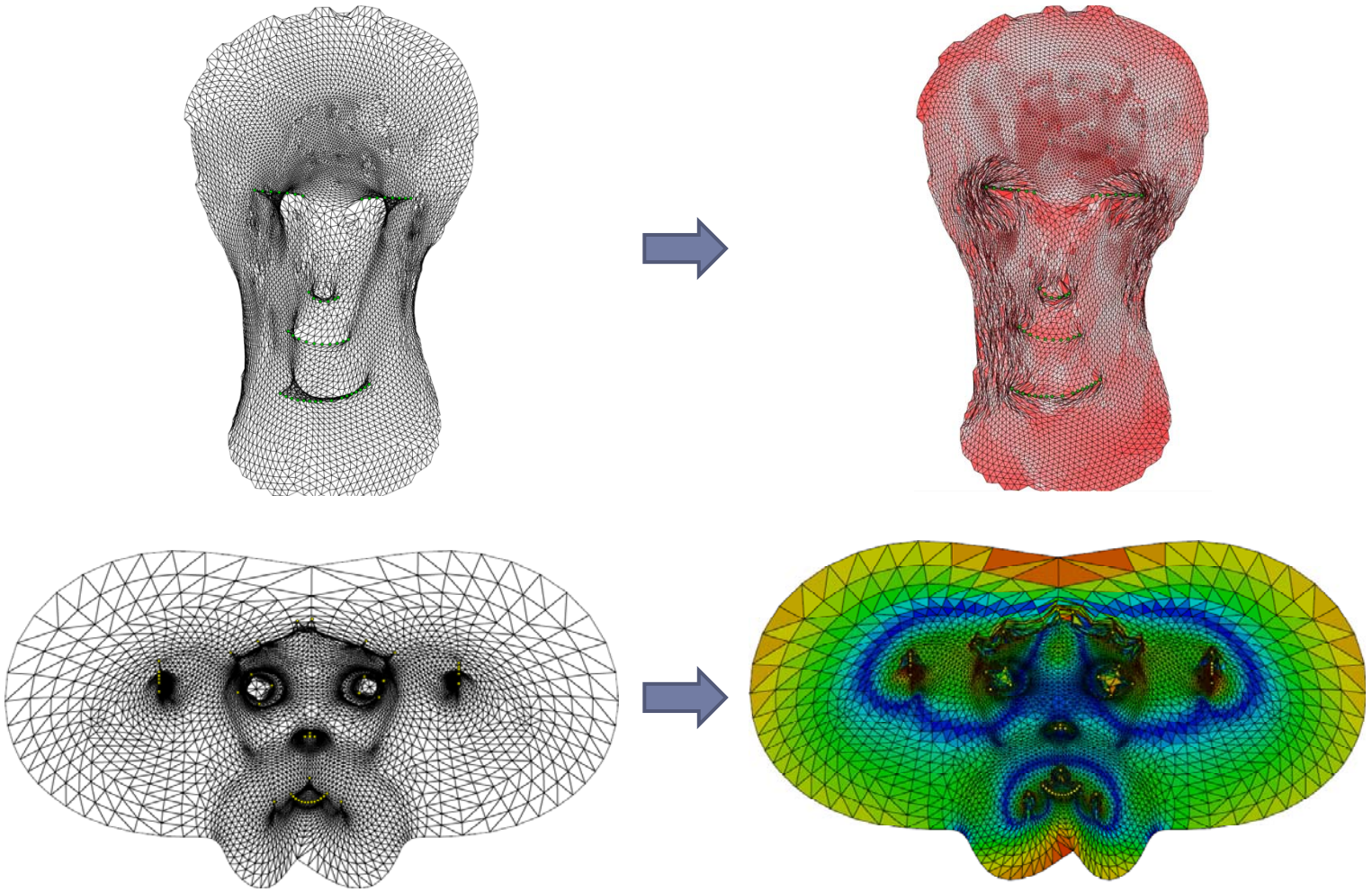
$$min f(x) = -\sum_{i}^{n} \det(\mathbf{A})$$

- Numerical Instability close to the solution due to invariance of the area to shearing operations



- Therefore, we also apply a shearing minimization operator

Mesh Untangling Results



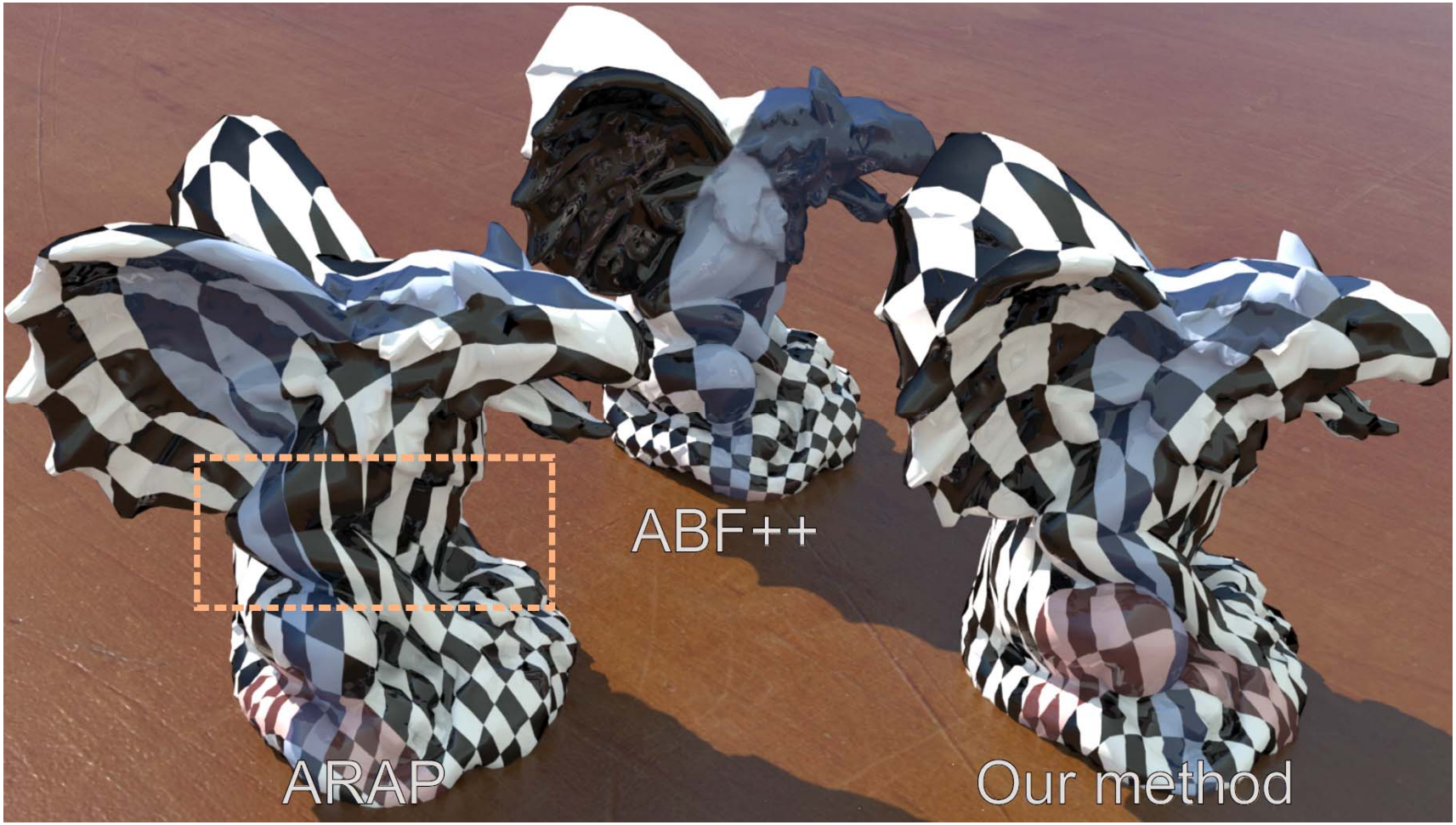
Results

method	model	# vertices	# faces	bijectivity	iters	# evals f	time(s)
ARAP	Blech	27993	55296	yes	7	-	1.95
Solver(1000)	Blech	27993	55296	yes	1000	2633	1.65
Solver(1500)	Blech	27993	55296	yes	1500	3658	2.38
ARAP	gargoyle	24406	48672	no	4	-	1.08
solver(5000)	gargoyle	24406	48672	yes	5000	15570	10.16
solver(10000)	gargoyle	24406	48672	yes	10000	30551	19.88
ARAP	julius	209083	416286	yes	28	-	59.02
solver(2500)	julius	209083	416286	yes	2500	5583	14.71
solver(4000)	julius	209083	416286	yes	4000	8943	23.92

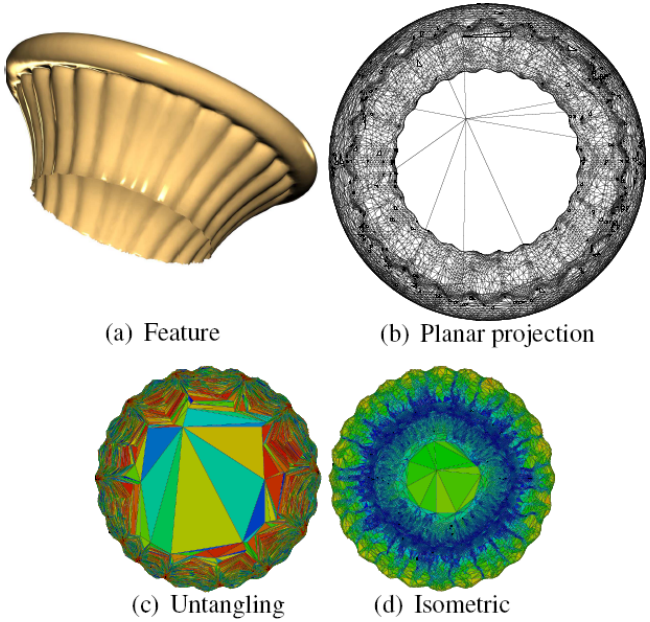
Method	Model	L2(min)	L2(max)	L2(rms)	Area(min)	Area(max)	Area(rms)	Angular
ARAP	Blech	0.719205	1.35083	0.0812426	0.565892	2.01864	0.15528	0.005267
Solver(1000)	Blech	0.721571	1.39323	0.0628883	0.595902	4.40744	0.13110	0.008731
Solver(1500)	Blech	0.708662	1.48147	0.0573159	0.574745	4.47457	0.12180	0.005351
ARAP	Gargoyle	0.408068	7465.28	74.4895	0.000164	8.2565	1.03866	0.421756
Solver(5000)	Gargoyle	0.18212	25.9687	3.81626	0.005277	31.9465	1.78979	0.108743
Solver(10000)	Gargoyle	0.226868	16.4789	2.36491	0.010048	22.0854	1.63601	0.121859
ARAP	Julius	0.63778	56.1057	0.203055	0.00486726	4.269	0.34954	0.043286
Solver(2500)	Julius	0.575291	2.03919	0.155464	0.299203	8.559	0.32586	0.041716
Solver(4000)	Julius	0.618229	2.00369	0.130395	0.315212	6.409	0.28318	0.045691

[1] L. Liu, L. Zhang, Y. Xu, C. Gotsman, S. J. Gortler, A local/global approach to mesh parameterization, in: Proceedings of the Symposium on Geometry Processing, SGP '08, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2008, pp. 1495–1504

Results



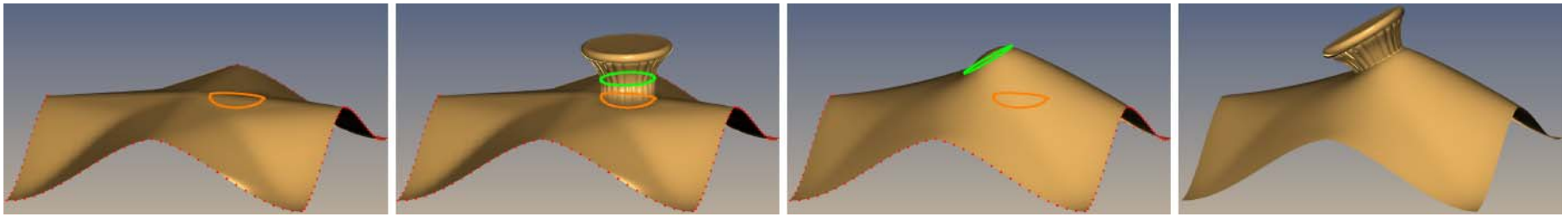
Cut and paste procedure



- Parameterize the base surface and the feature
  - Base surface can be parameterized with any method
  - Feature is parameterized with the OpenCL solver using a set of constrained vertices that match corresponding base feature vertices

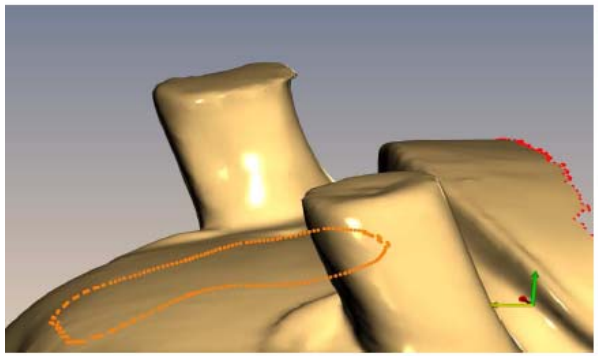
- Feature surface is stored in a floating point texture with the use of its parameterization

Cut and Paste Procedure

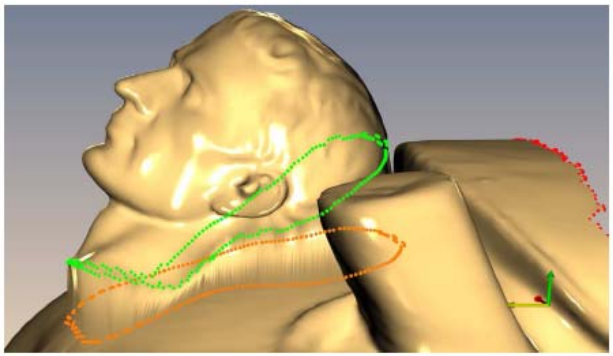


- Deform the base surface vertices with Radial Basis Functions (RBFs) to produce a suitable smooth surface for pasting
- Final surface is computed by tessellating the base surface with the use of the floating texture (OpenGL 4.0 tessellator unit)
- Rotation and translation of the pasted feature can be performed on the fly

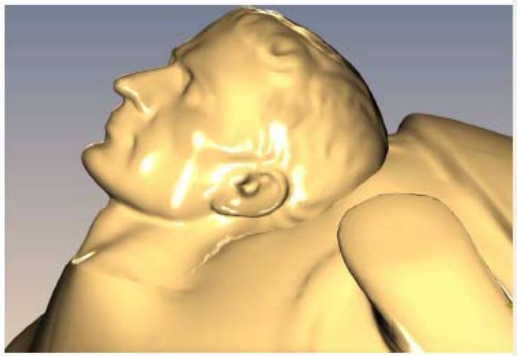
Cut and paste results



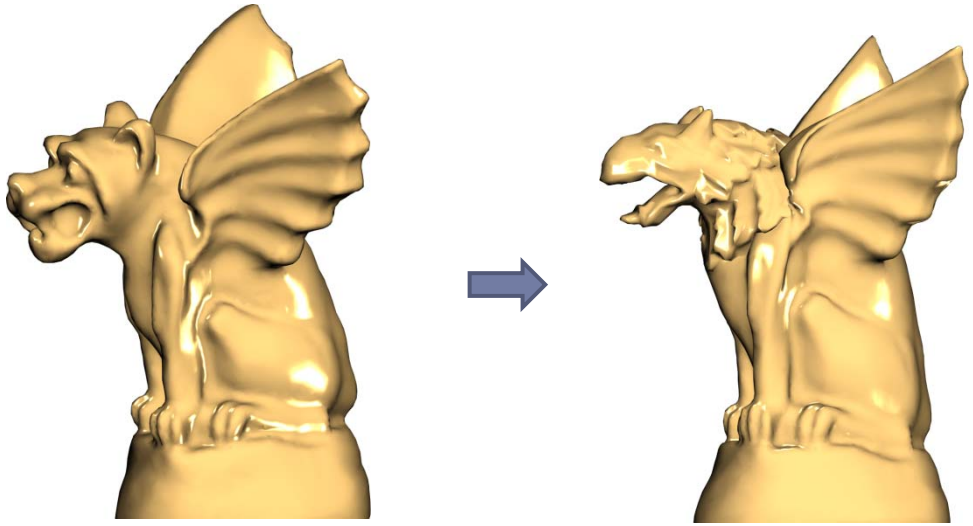
(a) Base surface



(b) Cut-and-paste without RBF



(c) RBF Interpolation



Conclusions

- A careful implementation of a non linear solver running on the GPUs is an order of magnitude faster than on the CPU

- Directly solving the non linear isometric parameterization problem without linear approximations or simplification is more robust and produce parameterizations with lower distortion. By using the power of GPUs the performance is comparable to the linear approximations

- Isometric parameterizations are a powerful tool when under sampling is an issue such as storing features for cut and paste operations

Thank you

Questions:

thathana@cs.uoi.gr

fudos@cs.uoi.gr

Software and source code:

www.cs.uoi.gr/~fudos/smi2013.html

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.