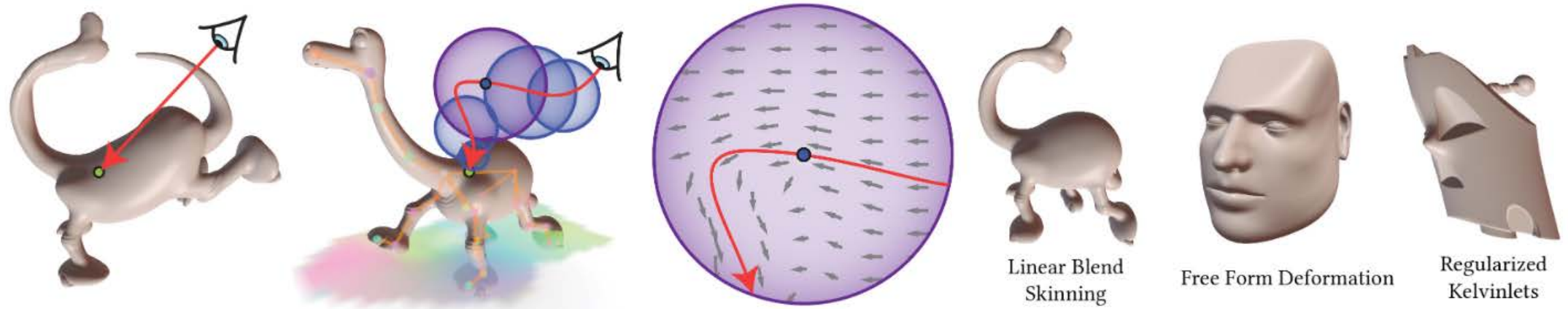
DGP Reading Group 2019-2020

Non-linear sphere tracing for rendering deformed signed distance fields

Dario Seyb[1] Alec Jacobson[2] Derek Nowrouzezahrai[3] Wojciech Jarosz[1]

[1]Dartmouth College [2]University of Toronto [3]McGill University

In *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 2019 *(in press)*

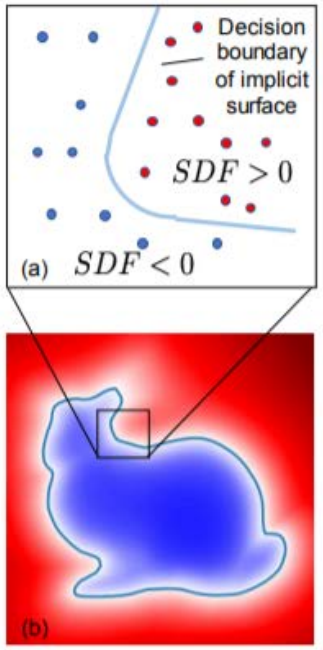


(a) Deformed Space (b) Undeformed Space (c) Initial Value Problem (d) Example Deformations

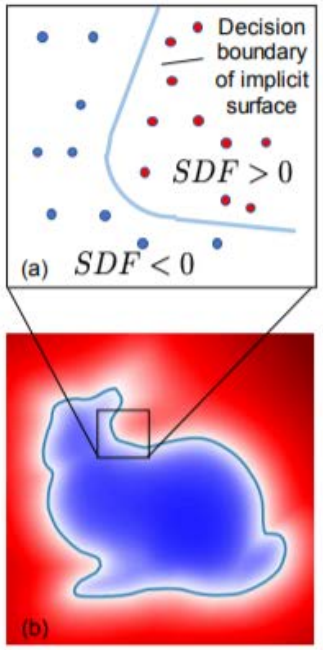
Presenter: Yuan Yao

2019.10.08

SDF(Signed Distance Function)



SDF(Signed Distance Function)

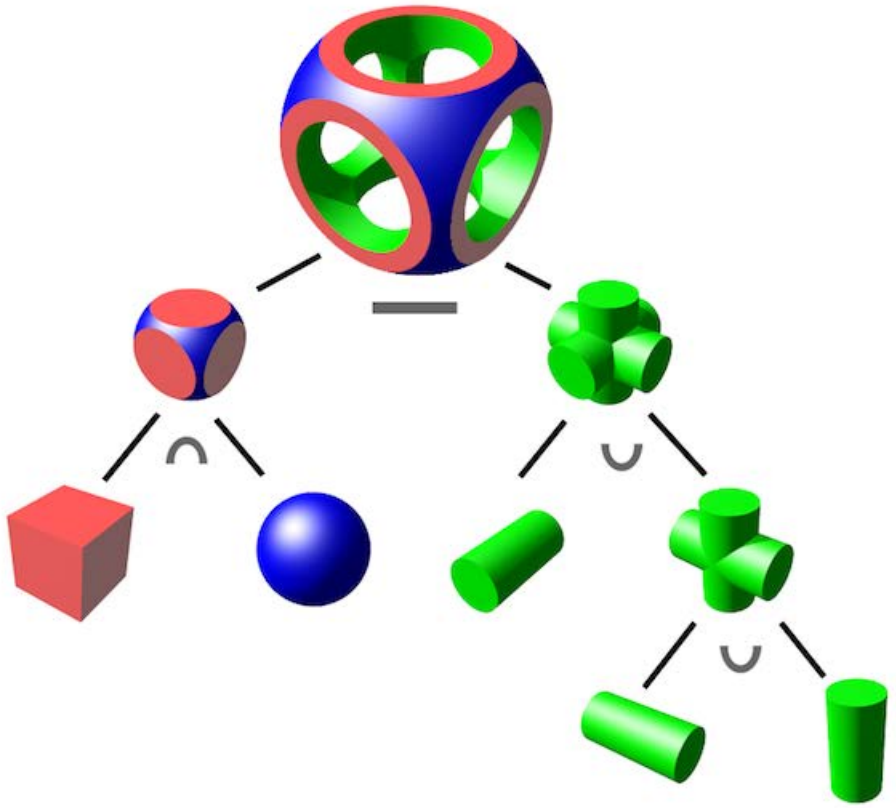


Sphere - exact

```
float sdSphere( vec3 p, float s )
{
  return length(p)-s;
}
```

SDF

- Advantages
  - Infinite resolution
  - Controllable continuity
  - Robust constructive solid geometry
  - Domain repetition
  - Smooth blending



SDF

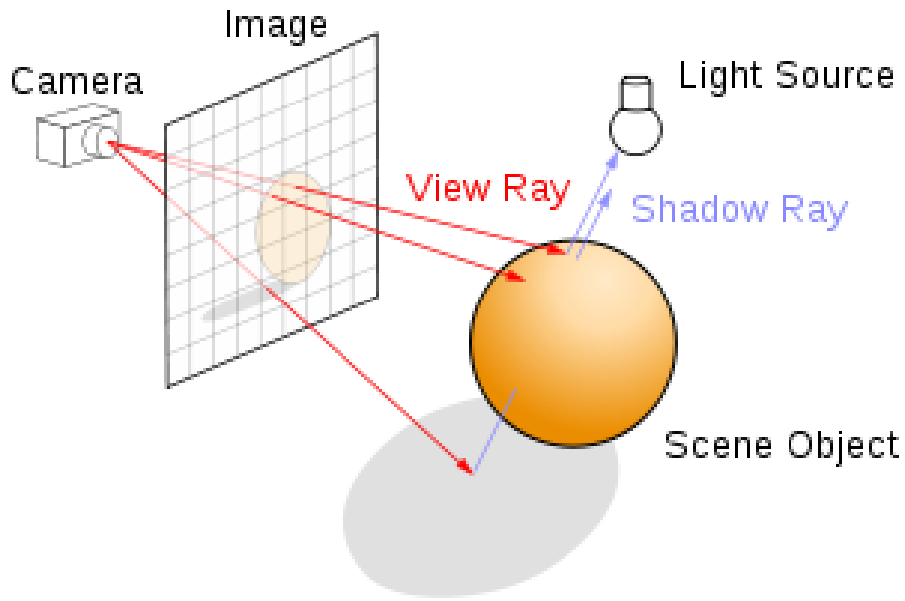
- Advantages
  - Infinite resolution
  - Controllable continuity
  - Robust constructive solid geometry
  - Domain repetition
  - Smooth blending
- Problem: not directly compatible with popular surface deformation developed for animating explicit surfaces.

Rendering SDF

- Inigo Quilez's website: http://iquilezles.org/

Rendering SDF

- Ray Tracing
  - Scene is defined in terms of explicit geometry.
  - Find intersection between ray and the scene.
  - Geometric intersection test.

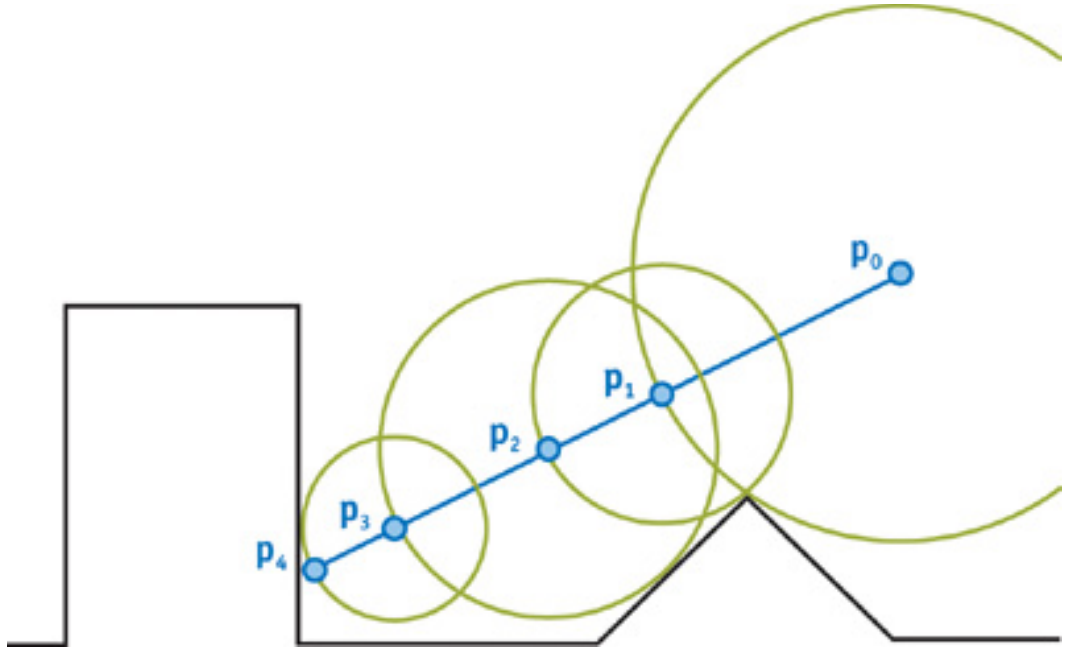


Rendering SDF

- Ray Marching
  - Scene is defined in terms of a signed distance function.
  - Find intersection between ray and the scene.
  - Start from camera, move a point along the view ray, bit by bit.
  - Check the value at the point, if negative(inside surface), done.

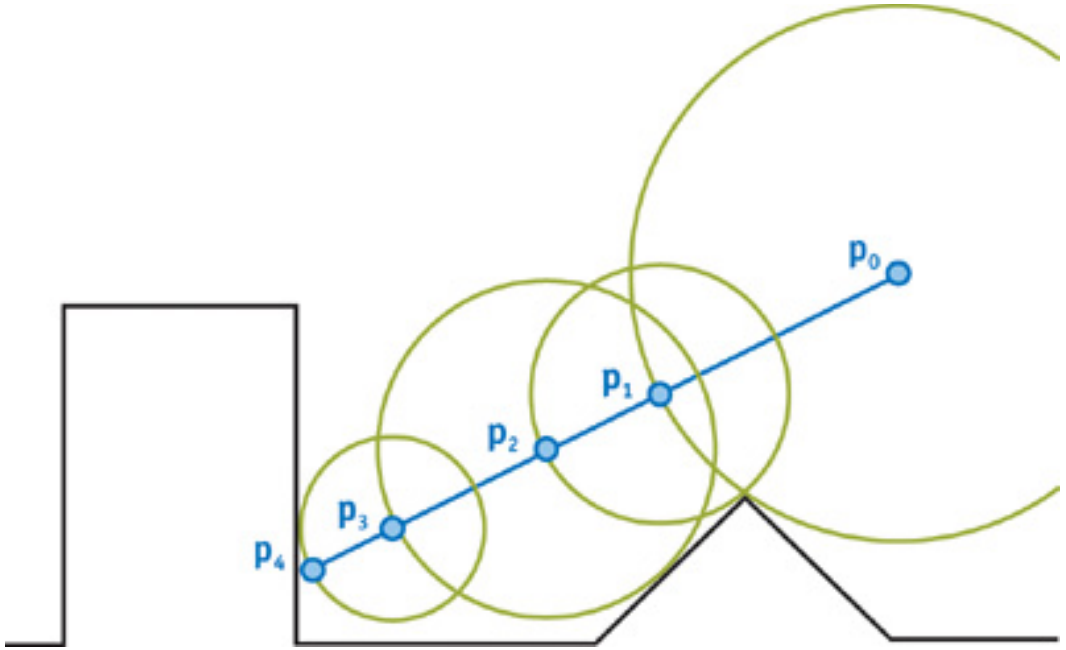
Rendering SDF

- Sphere Tracing
  - Use maximum step we know is safe.
  - Provided by SDF.



Transformation on SDF

- Need to compute inverse transformation.
- Easy for linear deformation.
- Hard for non-linear ones.



Non-linear Sphere Tracing

- The problem can be cast as the numerical integration of an ODE.
- Determine the initial value by an automatic construction.
- Maintain the strengths of SDF.
- Enable the rich palette of real-time deformation.
- Demonstrate on a prototypical implicit modeling/animation tool.

Method

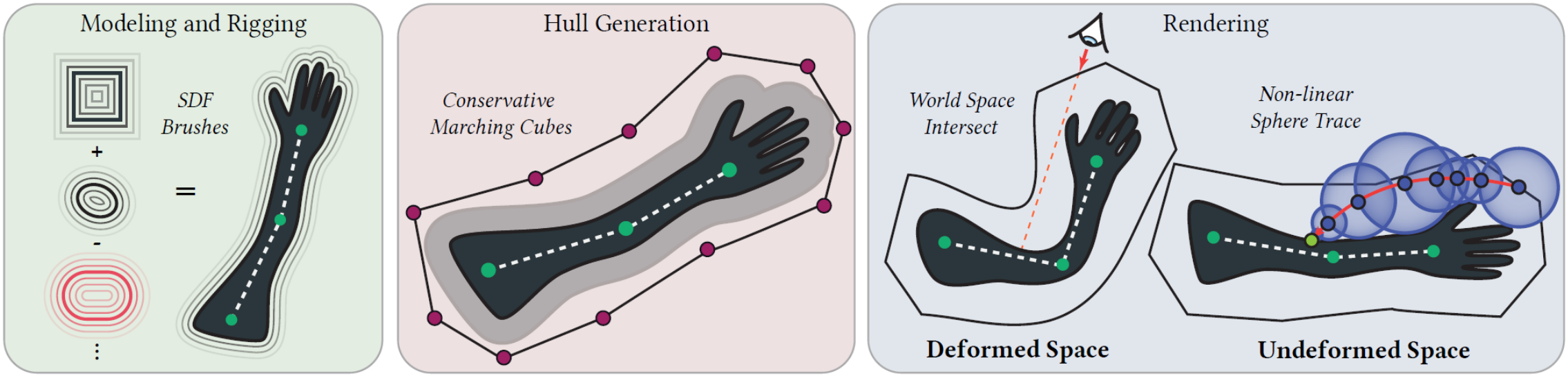
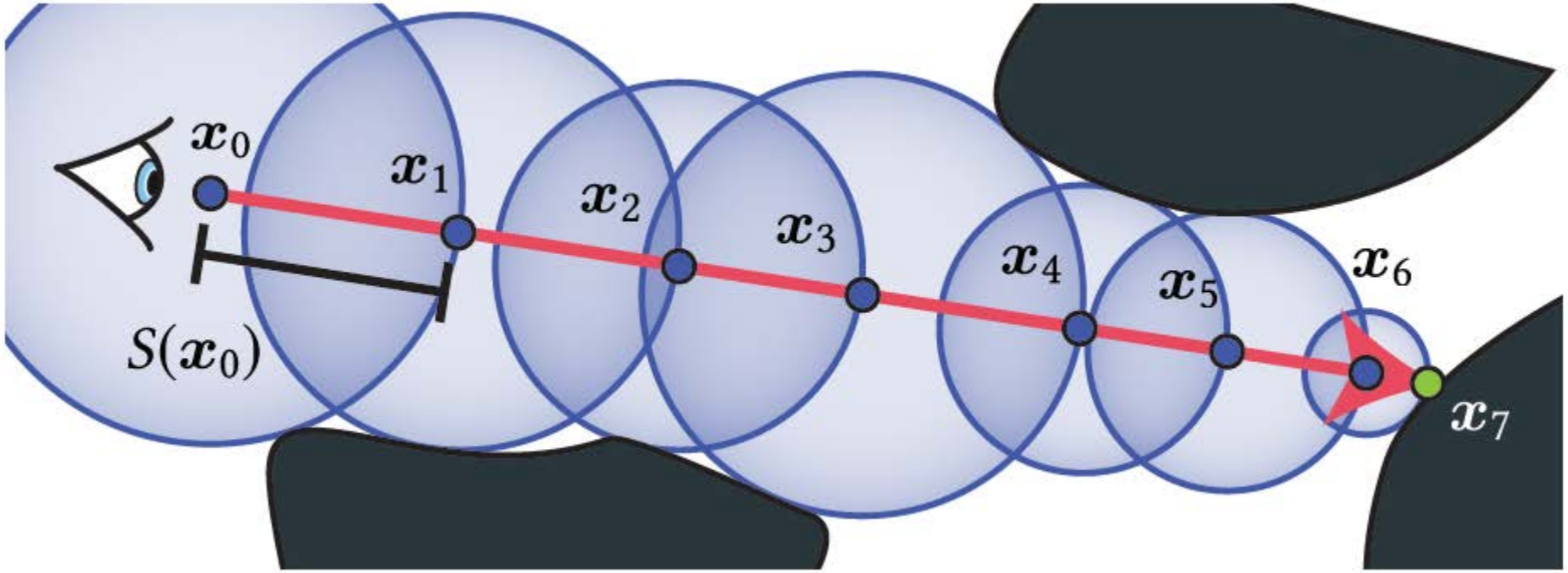


Fig. 2. Shown here is a high level overview over our method applied to linear blend skinning. After having modeled an implicit surface using a combination of *analytic brushes* and having defined an appropriate skeleton, we generate a *triangle mesh hull* that encloses the implicit surface via marching cubes at a low resolution. We then deform this hull with the chosen forward mapping deformation technique and rasterize it. The deformed space position retrieved from rasterization is easily transformed back to undeformed space via barycentric interpolation across the corresponding hull face (Section 3.2). A *non-linear sphere tracing* procedure is then started at the approximate undeformed space position (Section 3).

Sphere Tracing

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + |S(\boldsymbol{x}_i)|\,\boldsymbol{\omega} \quad \text{with} \quad \boldsymbol{x}_0 = \boldsymbol{p}.$$



Sphere Tracing

- For deformation, must have inverse:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \left|(S \circ D^{-1})(\boldsymbol{x}_i)\right| \boldsymbol{\omega}.$$

- But not always exist, especially for non-linear deformation

Non-linear Sphere Tracing

- Trace in undeformed space

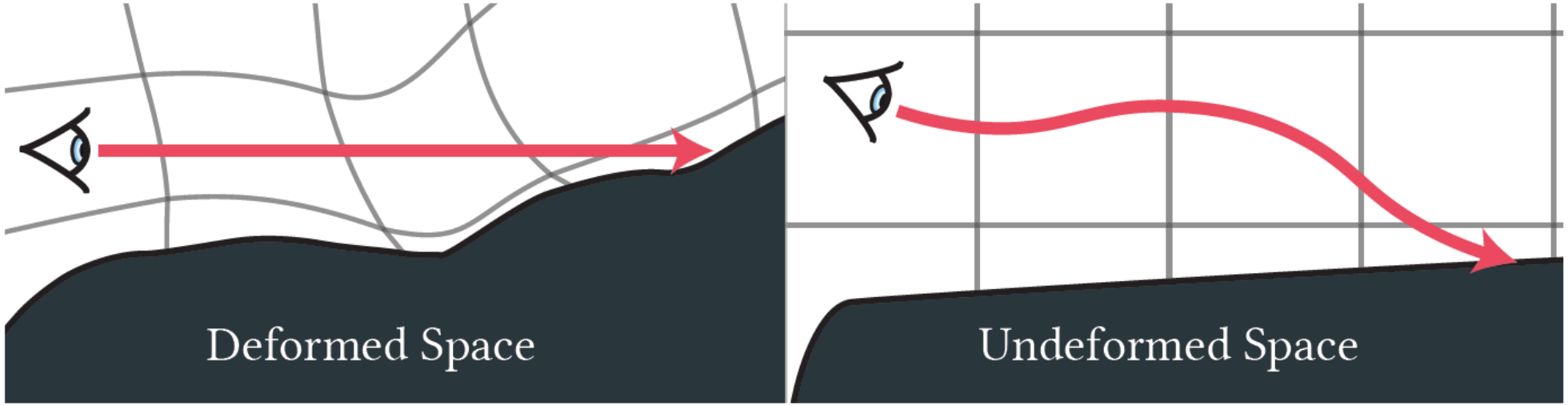


Fig. 4. A straight ray in deformed space (left) maps to a curve in undeformed space under deformation (right).

Non-linear Sphere Tracing

- Can be written as numerical integration

$$\boldsymbol{x}(s) = \boldsymbol{p} + s\boldsymbol{\omega} = \boldsymbol{p} + \int_0^s \boldsymbol{\omega}\,\mathrm{d}t.$$

- And with deformation:

$$\hat{\boldsymbol{x}}(s) = D^{-1}(\boldsymbol{p}) + \int_0^s J_{D^{-1}}(\boldsymbol{x}(t))\,\boldsymbol{\omega}\,\mathrm{d}t$$

Non-linear Sphere Tracing

- Two issues:
  - Cannot compute integral analytically
  - Still need to evaluate D-1 once, for start point p

A joint method for Root Finding and Ray Integration

- A naïve extension from sphere tracing to non-linear sphere tracing is:

$$\hat{\boldsymbol{x}}_{i+1} = \hat{\boldsymbol{x}}_i + |S(\hat{\boldsymbol{x}}_i)|\, \hat{\boldsymbol{\omega}}(\hat{\boldsymbol{x}}_i), \quad \text{with} \quad \hat{\boldsymbol{x}}_0 = D^{-1}(\boldsymbol{p}).$$

- Problem: error quickly accumulate when ray is highly curved.

A joint method for Root Finding and Ray Integration

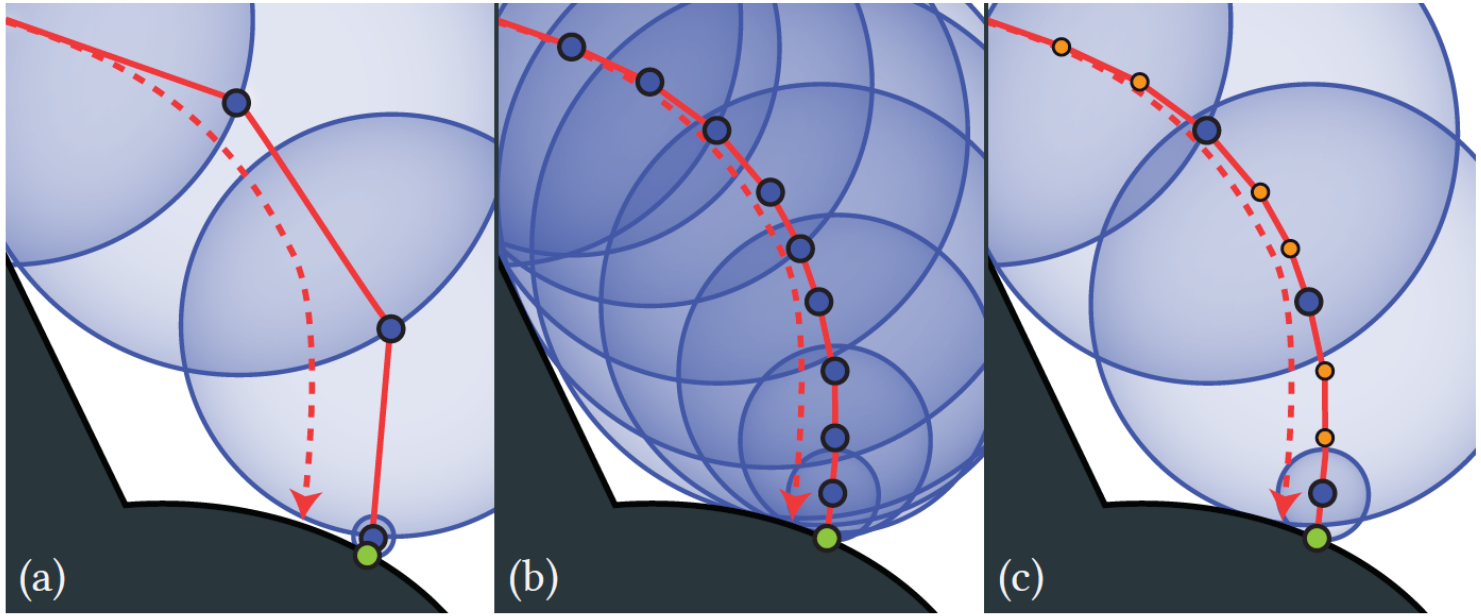


Fig. 5. With strongly deformed rays, the step size given by $S(\hat{\boldsymbol{x}}_i)$ might be too large to accurately sample the ray (a). Reducing the step size naïvely improves accuracy, but necessitates many more evaluations of S, degrading performances (b). Our method reproduces the ray to the same accuracy while only minimally increasing the number of evaluations of S (c).

A joint method for Root Finding and Ray Integration

- More generally, it can be cast as ODE:

$$\boldsymbol{y}'(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{\omega}}(\hat{\boldsymbol{x}})$$

- The solve takes initial condition and integration duration as parameters.

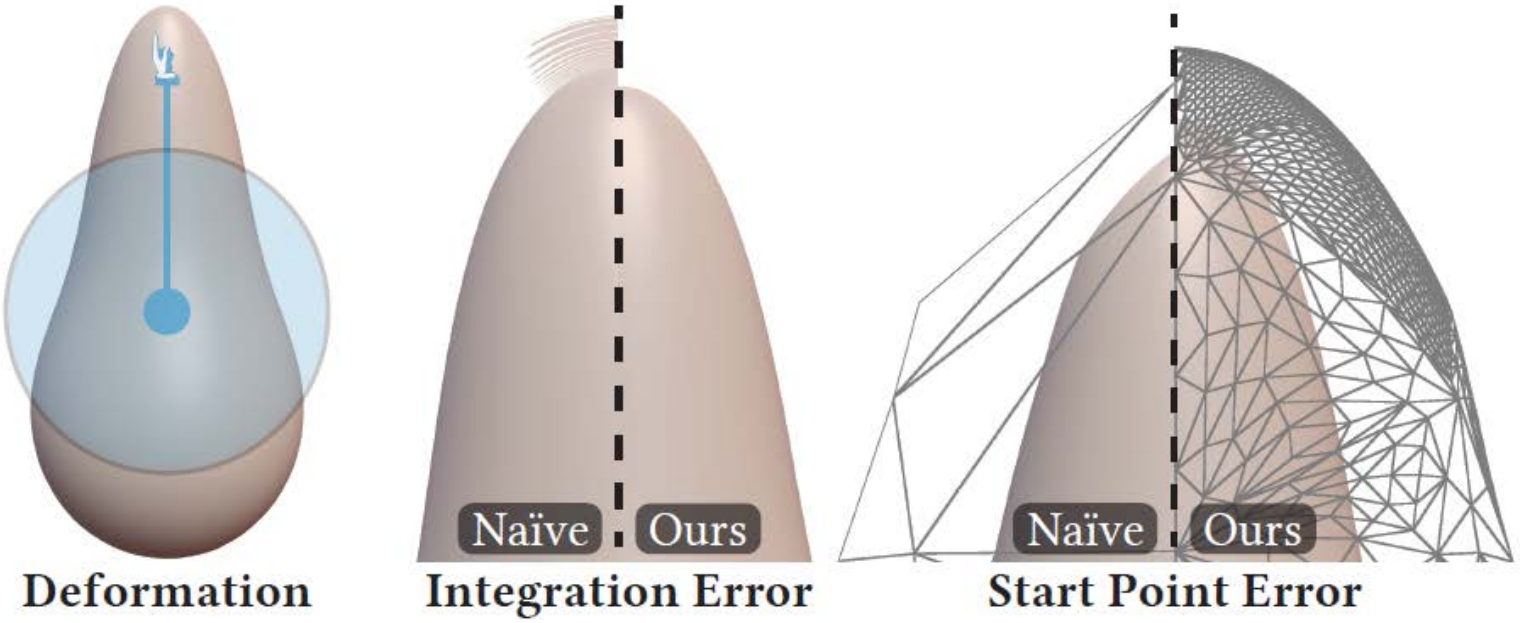
- Still need to find start point.

Find Undeformed Space Ray Start

- Generate a low resolution explicit hull contain the surface.
- Deform the hull vertices using the forward transform.
- Intersect the deformed triangle mesh with the deformed space view ray.
- Actually a generalization of the bounding-box technique(Barr et. al, 1986) to determine the initial value.

Principled Methods for Controlling Error

- Two types of error:
  - Numerical integration
  - Approximation of the inverse via hull linearization



Principled Methods for Controlling Error

- Choose an appropriate ODE solver: hybrid approach
  - When step is large, use adaptive Runge-Kutta integrator
  - When step is small, use simple Euler integration

Principled Methods for Controlling Error

- Reducing hull linearization error via adaptive subdivision

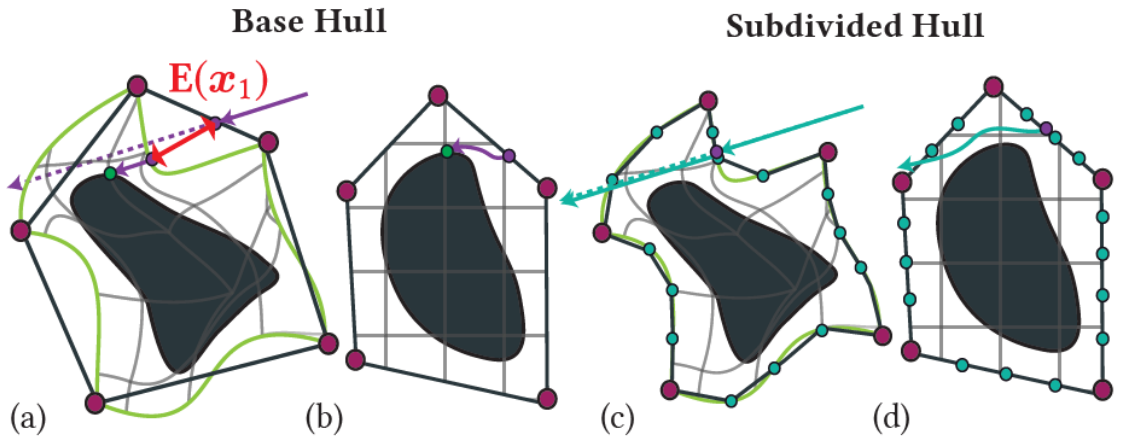


Fig. 7. Using linear interpolation to find $\widetilde{D^{-1}}(\boldsymbol{p}) \approx D^{-1}(\boldsymbol{p})$ assumes that $D(\boldsymbol{p})$ is linear. This is rarely the case in the deformations we discuss here. So for large triangles (a/b), $\widetilde{D^{-1}}(\boldsymbol{p}) \neq D^{-1}(\boldsymbol{p})$ and particularly $D(\widetilde{D^{-1}}(\boldsymbol{p})) \neq \boldsymbol{p}$. This leads to computing the wrong undeformed space ray start points, resulting in surface artifacts in the final image. For example, in (a) the ground truth ray represented by the dashed line does not hit the shape, but the ray NLST traces (b) generates a hit due to error in the start point. When we reduce the size of each linear segment, the error reduces as well and the ray traced by NLST corresponds closely to the ground truth (c/d).

Evaluation and Results

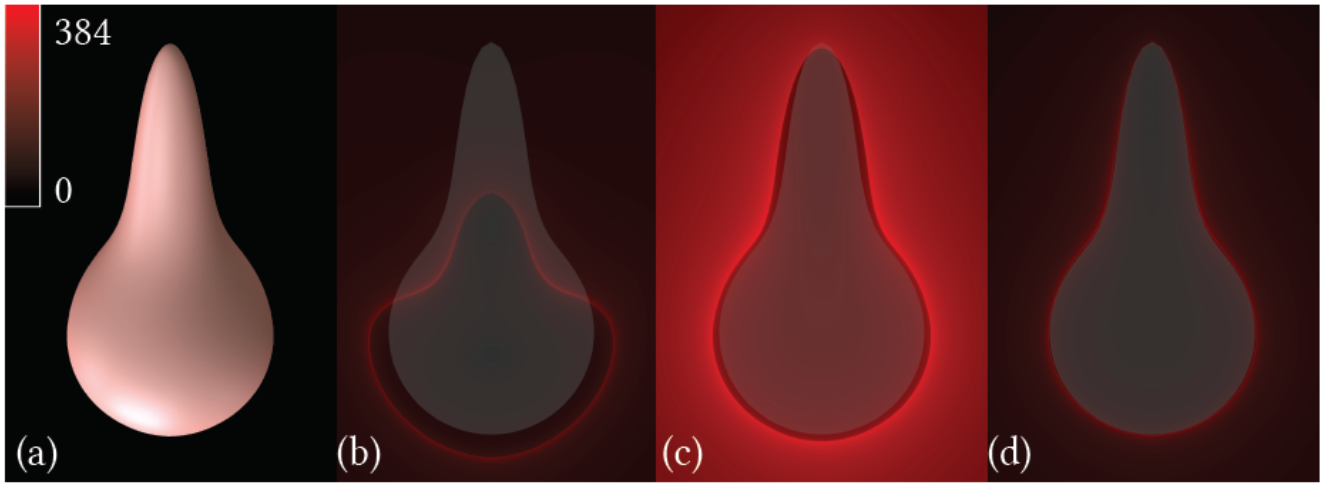
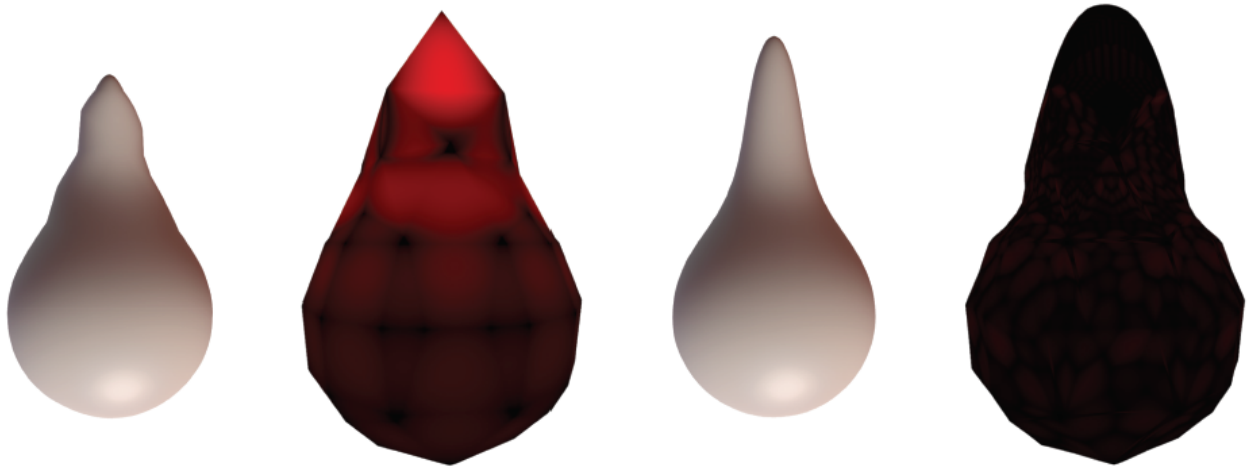


Fig. 8. Above we show the number of SDF evaluations for each pixel in red and the ground truth shape is overlaid in gray on the rendered images. When rendering a strongly deformed object (a), naive non-linear sphere tracing (b) completely fails to reproduce the ground truth shape. Artificially reducing the sphere tracing step size (c) solves this issue, but greatly increases the number of S evaluations (visualized in red). Our sub-steps do not require the evaluation of S, allowing us to accurately reproduce the ground truth at little additional cost (d).

Evaluation and Results



No Hull Subdivision

With Adaptive Subdivision

Fig. 9. When we do not subdivide the hull sufficiently, NLST does not reproduce the isosurface faithfully (left). This is due to the error $E(\boldsymbol{p})$, shown in red, in the undeformed space start point (middle left). Once we apply our adaptive subdivision scheme (middle right), error is reduced to imperceptible levels (right).

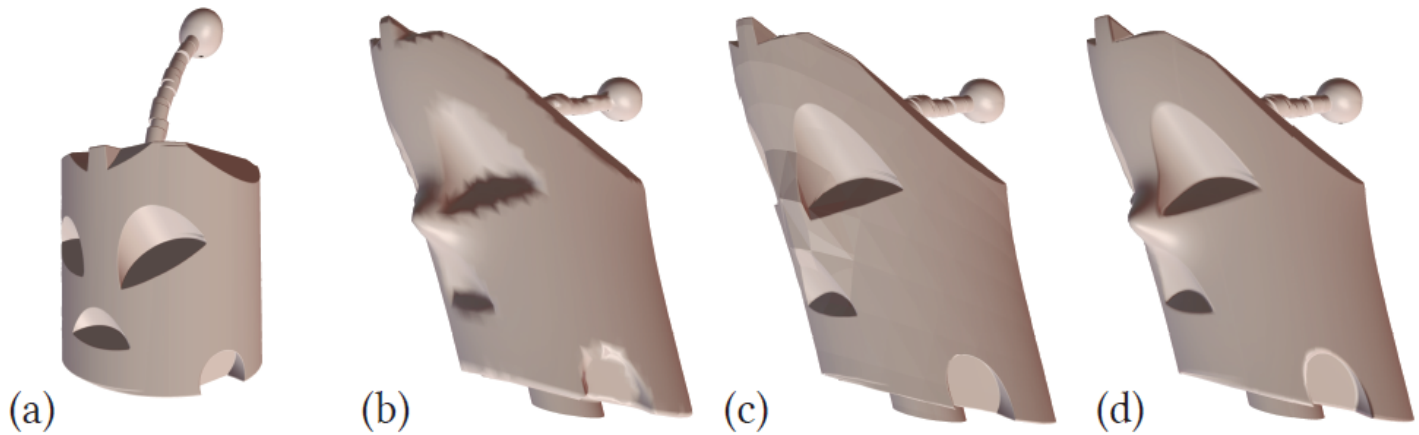


Fig. 12. Starting with an analytically described distance field (a) we apply a forward deformation composed of two Kelvinlet brushes via isosurface extraction (b), linearizing the deformation (c) and our method (d). Our method combines the support for small scale deformations that a dense mesh provides with the accurate reproduction of sharp features typical of direct implicit surface rendering techniques.

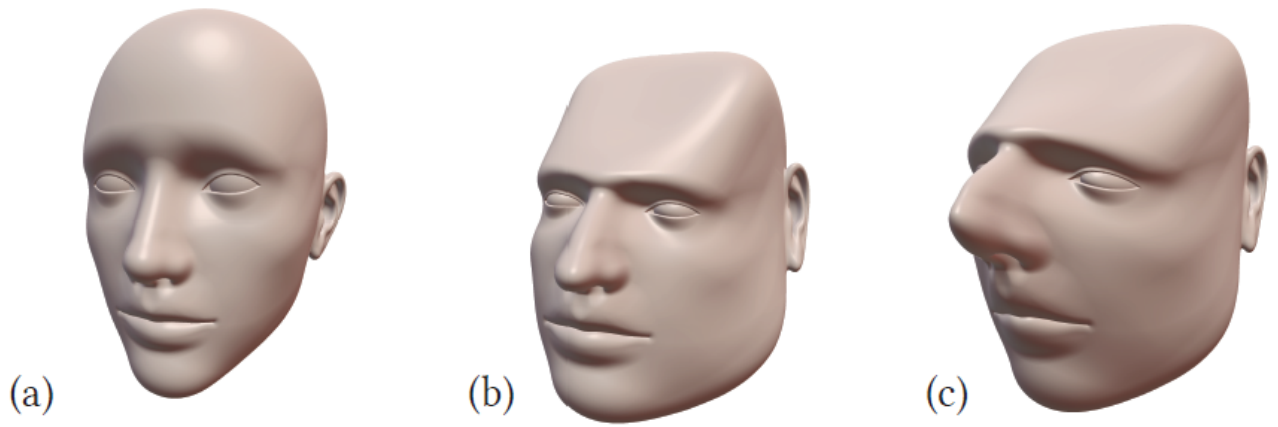


Fig. 13. We apply cubic free form deformation to a signed distance field (a), resulting in a deformed shape (b). A benefit of our method is that combining multiple deformations is trivial. We show this by applying a "Regularized Kelvinlet" [De Goes and James 2017] edit to the result of free form deformation (c).

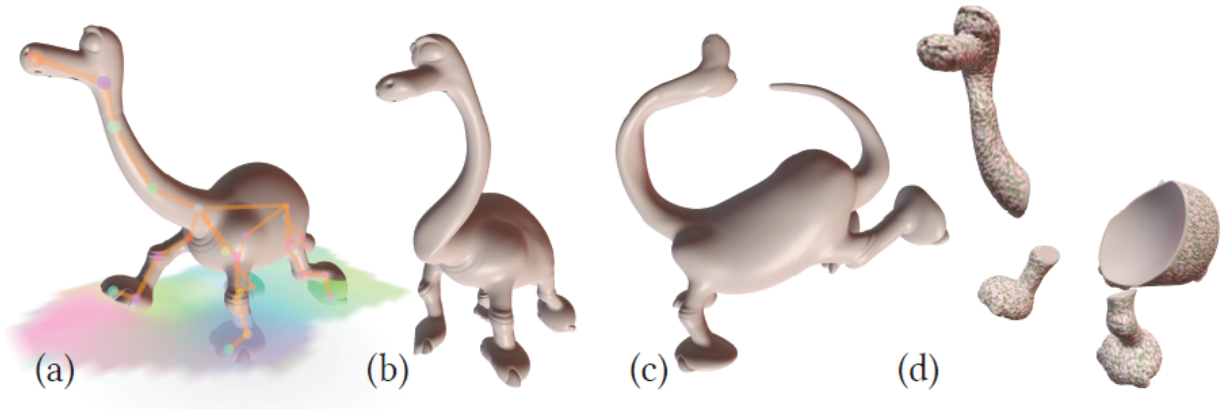


Fig. 14. We generate bounded biharmonic weights [Jacobson et al. 2011] volumetrically based on a user defined skeleton and the hull mesh (a). We can then use the weights and the skeleton to derive D. NLST treats this deformation like any other deformation function and we can pose the character (b). In (c) we show a pose where the head and the tail of the character meet and the deformation is *globally* non-bijective. We can still render this pose with NLST, since *locally* (within the head and within the tail), the deformation is invertible. In (d) we manipulate the underlying SDF before deformation (for texture detail) and after (for geometry manipulation).

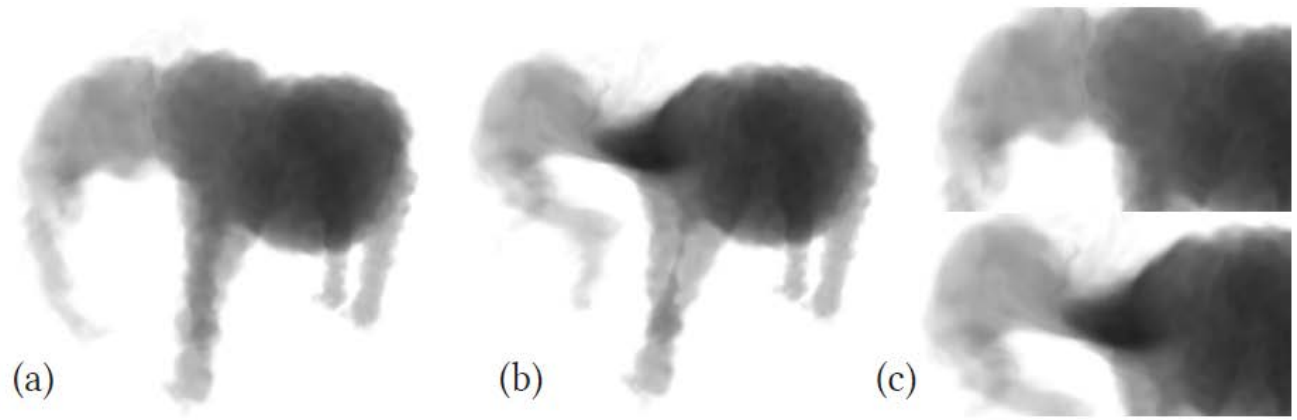


Fig. 15. We deform a volume with a density function derived from an SDF with added noise (a). NLST correctly accounts for volume compression and expansion, making compressed areas darker (b,c).

Conclusion

- Non-linear sphere tracing
- Inherit limitation from root finding and sphere tracing
- Performance worse than ray tracing or rasterization
- Concave objects, runs out of iteration
- Only applicable to SDFs
- Have to be locally foldover-free

Discussion

Thank you!