An Adaptive Screen-Space Meshing Approach for Normal Integration

Moritz Heep & Eduard Zell

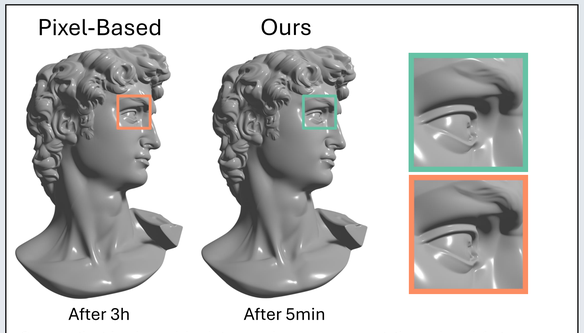
University of Bonn

Moritz Heep
PhD Student

Niebuhrstraße 1a
53113 Bonn
+49 228/73-60834
moritzheep@gmx.de
moritzheep.github.io

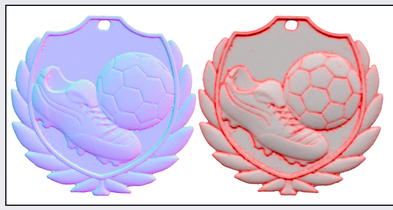
Motivation

Increasing the resolution of the normal map improves the accuracy of fine structures but increases computational complexity. In smooth, featureless regions, this added complexity yields little additional information.



Our adaptive triangle mesh leads to a much sparser representation and much faster integration times while generating comparable results. The normal map was 64MP.

We introduce an adaptive screen-space meshing approach to reduce complexity *before* integration and give a full derivation of the normal integration on general triangle meshes.



Fine details (in red) typically only make up a fraction of an object's surface.

By focusing on fine details and removing redundant information, we can avoid the quadratic growth of variables with increasing geometric resolution in pixel-based methods.

References

[1] X. Cao et al., 'Normal Integration via Inverse Plane Fitting With Minimum Point-to-Plane Distance', *CVPR*, 2021.

[2] M. Dunyach et al., 'Adaptive remeshing for real-time mesh deformation', *Eurographics, 2013*.

[3] M. Heep and E. Zell, 'ShadowPatch: Shadow Based Segmentation for Reliable Depth Discontinuities in Photometric Stereo', *Pacific Graphics*, 2022

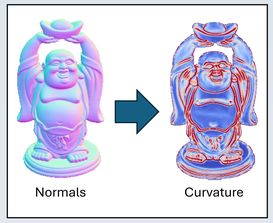
[4] M. Li et al., 'Multi-view photometric stereo: A robust solution and benchmark dataset for spatially varying isotropic materials', *TIP*, 2020.

[5] W. Xie et al., 'Surface-from-gradients: An approach based on discrete geometry processing', *CVPR*, 2014.

Acknowledgements

The "David Head" by 1d_inc and the "Football Medal2 - PhotoCatch" Moshe Caine were licensed under CC BY 4.0.
This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 (PhenoRob).

Curvature



Calculate curvature by solving the generalised Eigenproblem

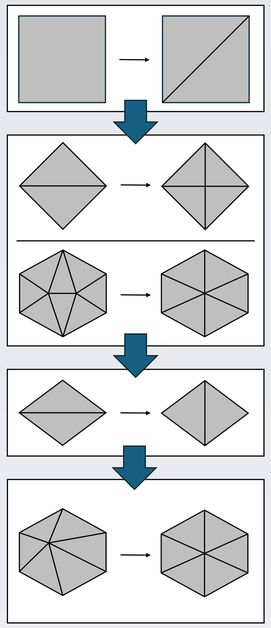
$$\kappa_i \cdot I\vec{v}_i = II\vec{v}_i$$

1st fundamental form $I_{ij} = \partial_i\vec{x} \cdot \partial_j\vec{x}$

2nd fundamental form $II_{ij} = \partial_i\vec{x} \cdot \partial_j\vec{n}$

We calculate surface gradients from $\vec{n} \cdot \partial_i\vec{x} = 0$ and use finite differences for $\partial_i\vec{n}$.

Adaptive Screen-Space Meshing

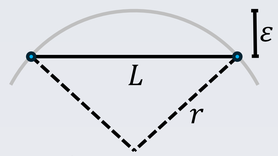


Initialize Mesh by covering each pixel with two triangles.

Split / Collapse Edges to make their length closer to the optimal length

$$L = \sqrt{\frac{6\varepsilon}{\kappa} - 3\varepsilon^2}$$

that aims to keep the deviation between mesh and underlying surface below a user threshold ε.



Since $r = \kappa^{-1}$, curvature can be used to estimate the deviation ε between the mesh and the underyling surface [2].

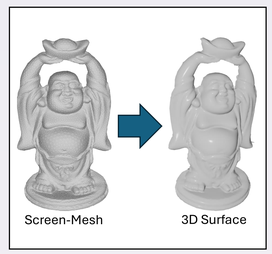
Flip Edges that are locally Non-Delaunay.

Move Vertices to the weighted centroid of their star by solving

$$\left(\sum_{f\in\mathcal{F}_v} \frac{A_f}{L_f^2} \cdot \sqrt{\det I_f} \cdot I_f\right) \cdot \vec{u}_v^{(k+1)} = \left(\sum_{f\in\mathcal{F}_v} \frac{A_f}{L_f^2} \cdot \sqrt{\det I_f} \cdot I_f \cdot \vec{c}_f\right)$$

Including the first fundamental form I_f compensates for foreshortening.

Mesh-Based Integration



A unified treatment of the orthographic and perspective projection can be achieved by using the functional

$$E_{\text{Int}} = \int_\Omega (\vec{n} \cdot \vec{r}\,\partial_u z + n_x)^2 + (\vec{n} \cdot \vec{r}\,\partial_v z + n_y)^2 \,\mathrm{d}u\,\mathrm{d}v$$

camera ray — (log) depth

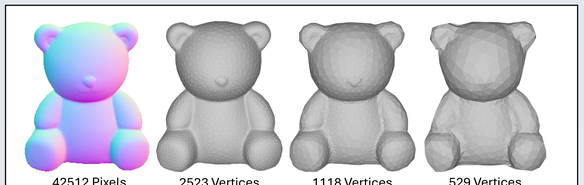
We derive a discretized version of this energy for triangle meshes. As in the pixel case, integration is performed by solving a linear system but with much fewer variables.

Results

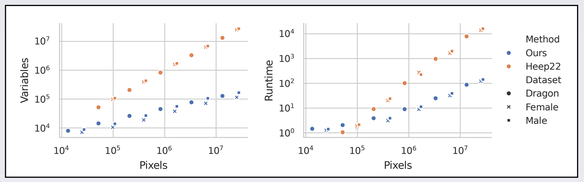
Our method is not bound to the image resolution. Instead, the mesh resolution is controlled by a user parameter ε. Therefore, we conduct all experiments at three quality settings: high (0.1 mm), medium (0.3 mm) and low (1 mm).

Compression

We investigated the compression rates of our method on the DiLiGenT-MV dataset [4]. Even for the tightest error threshold, we only need 6 to 18% of vertices compared to pixels.



At higher resolutions, we achieve even more compression. Experiments suggest that the number of vertices grows significantly slower than the number of pixels.



The fewer variables also make our linear solver converge significantly faster. The results suggest that our method has a lower runtime complexity with increasing resolutions.

Benchmark Evaluation

Given the high level of compression achieved by our method, it is not surprising that pixel-based approaches perform slightly better.

		Pixel-based			Combined [1, 2]			Uniform			Ours		
		[1]	[5]	[3]	Low	Mid	High	Low	Mid	High	Low	Mid	High
Ortho	Bear	**2.97**	3.36	n/a	3.12	2.99	2.98	4.39	4.07	3.74	3.95	3.65	3.37
	Buddha	**6.74**	6.85	n/a	6.78	6.76	6.75	7.79	7.62	7.41	7.74	7.54	7.33
	Cow	**2.45**	2.51	n/a	2.67	2.55	2.53	3.59	3.39	3.10	3.42	3.12	2.96
	Pot2	5.15	**5.02**	n/a	5.18	5.15	5.15	5.99	5.84	5.73	5.89	5.77	5.65
	Reading	6.34	**6.05**	n/a	6.37	6.35	6.34	7.18	7.00	6.87	7.08	6.93	6.83
Persp	Bear	**2.91**	n/a	3.05	3.09	2.94	2.92	4.62	4.17	3.84	3.94	3.72	3.47
	Buddha	6.75	n/a	**6.60**	6.80	6.77	6.76	7.85	7.64	7.43	7.74	7.53	7.40
	Cow	**2.35**	n/a	2.51	2.59	2.46	2.43	3.71	3.47	3.21	3.49	3.24	3.07
	Pot2	**4.99**	n/a	5.23	5.04	4.99	4.99	6.08	5.97	5.81	6.04	5.86	5.76
	Reading	**6.28**	n/a	6.29	6.33	6.29	6.29	7.20	7.03	6.89	7.19	6.94	6.85

For the *Combined* column, we performed a pixel-wise integration [1] followed by conventional remeshing [2]. For the *Uniform* column, we created meshes with uniform edge lengths that match the number of vertices of their adaptive counterparts.