

Feature-Preserving Mesh Decimation for Normal Integration

Moritz Heep
PhenoRob
University of Bonn
mheep@uni-bonn.de

Sven Behnke
Autonomous Intelligent Systems
University of Bonn
behnke@ais.uni-bonn.de

Eduard Zell
Independent Researcher
ezell@hotmail.de

Abstract

Normal integration reconstructs 3D surfaces from normal maps obtained e.g. by photometric stereo. These normal maps capture surface details down to the pixel level but require large computational resources for integration at high resolutions. In this work, we replace the dense pixel grid with a sparse anisotropic triangle mesh prior to normal integration. We adapt the triangle mesh to the local geometry in the case of complex surface structures and remove oversampling from flat featureless regions. For high-resolution images, the resulting compression reduces normal integration runtimes from hours to minutes while maintaining high surface accuracy. Our main contribution is the derivation of the well-known quadric error measure from mesh decimation for screen space applications and its combination with optimal Delaunay triangulation. Code is available at <https://moritzheep.github.io/anisotropic-screen-meshing>.

1. Introduction

Normal maps can be estimated from images through shape-from-shading [15] or photometric stereo [29] down to the pixel level, capturing delicate surface details. Normal integration reconstructs the underlying surface using these normals. The surface is usually estimated as a pixel-based depth map [7] by solving a linear system. At the pixel level, doubling the geometric resolution requires doubling both, image height and width. Due to this quadratic growth of the number of variables in the linear system, normal integration scales poorly to higher image resolutions. The fundamental problem of a regular grid is that fine details somewhere on screen require increasing the resolution everywhere on screen.

Recent research proposes substituting the pixel grid with an adaptive 2D triangle mesh *prior to* integration [14]. Triangle meshes allow local control of the resolution: the resolution remains high in areas of delicate details and is lowered within smooth areas. While this reduces the number

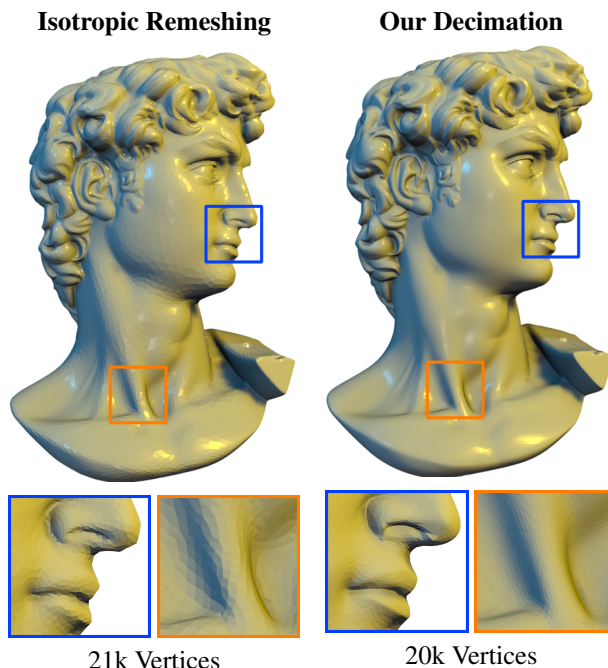


Figure 1. Visual comparison between the isotropic remeshing [14] (left) and our anisotropic decimation (right). In both examples, the 2048^2 normal map is compressed into a triangle mesh with approximately 20k vertices *before* integration. By aligning vertices and edges to ridges and furrows, we achieve higher accuracy with the same number of vertices.

of vertices compared to the number of pixels (and consequently runtime), this approach is isotropic and struggles with ridges and furrows by design. Our proposed method overcomes this limitation by shifting to an anisotropic formulation: By analysing the normal maps, we align edges and vertices of the mesh with ridges and furrows of the underlying geometry. Our mesh decimation technique maintains delicate surface details more effectively than previous methods, even with high compression ratios, see Fig. 1. In summary, our contributions are:

- We derive quadrics [11], originally designed for 3D meshes and crucial to mesh decimation methods but take

normal maps as input to facilitate mesh decimation in screen space.

- By taking advantage of the similarity between quadrics and a generalized Delaunay criterion [4], we align triangle edges with ridges and furrows, thus outperforming previous methods [14].
- We present a simple iterative algorithm, consisting only of three local mesh operations: edge collapse, edge flip, and vertex position update.

Our results suggest that even at around 90% compression, we achieve surfaces within the sub-millimetre range of pixel-based approaches. Furthermore, given a similar time range we can triangulate, decimate and integrate a 16MP normal map, while pixel-based integration succeeds only to integrate 4MP. A reference implementation is available under <http://moritzheep.github.io/anisotropic-screen-meshing>.

2. Related Work

Normal Integration The core of recent normal integration methods is a functional [7] quantifying the difference between the gradients of the actual depth-map and observed gradients, *e.g.* from photometric stereo. Normal integration is then achieved by finding the depth map that minimizes this functional. If the L^2 norm is used to measure the difference, the optimal depth map can be found by solving a Poisson equation in the form of a sparse linear system. This linear system is either found by discretising the functional itself [9] or using functional analysis and discretising the resulting Poisson equation [16]. An overview of the topic is given by Quéau et al. [25]. More recently, authors have raised concerns about various artefacts [2, 36] occurring in the functional setting. Although these artefacts can be avoided by using additional variables [2] or larger stencil sizes for the partial derivatives [35], replacing the functional over gradients by a functional over normals represents a more straightforward solution [3, 13]. Our method is derived from the same normal-based functional [14] and remains unaffected by the aforementioned artefacts.

Mesh-Based Integration Early mesh-based methods interpreted the pixel-grid as a grid of quadrilateral facets [30, 31]. These methods iterate between aligning the quads with the normal directions and gluing adjacent quads into a continuous or even discontinuous [32] surface. However, this approach maintains a one-to-one relation between pixels and quads. More recently, a variational approach to normal integration was introduced for arbitrary triangle meshes [14] together with an isotropic screen space meshing algorithm. By calculating curvatures from the normal maps, the triangle density is adjusted locally, facilitating much sparser representations and reducing runtimes without sacrificing quality. Unlike this approach, we abandon

isotropy in favour of an alignment of vertices and edges to ridges and furrows of the underlying geometry. The result is a much more faithful representation.

Remeshing and Decimation Modification or simplification of an existing triangle mesh has been extensively studied in the field of geometry processing [17]: Decimation aims to preserve the shape with a reduced number of vertices. Remeshing aims to improve mesh regularity and might increase or decrease the vertex number. A wide range of remeshing algorithms build either on Centroidal Voronoi Tessellations (CVT) [6] or Optimal Delaunay Triangulations (ODT) [4]. Both approaches create uniform or isotropic meshes, *i.e.* meshes with equilateral triangles of constant or varying size by iteratively moving vertices to the centroid of a local neighbourhood. Curvature-based densities [1, 5] are a common choice, as they ensure an accurate representation of the geometry [8]. Extending CVT or ODT to create anisotropic meshes is more elaborate. One option is to 'lift' vertex positions and normals into a six-dimensional space [19, 23, 34]. The resulting CVT is isotropic in this space but anisotropic in 3D. Since the change in normals is proportional to curvature, these triangulations still implicitly adapt to curvature.

Mesh simplification typically reduces the vertex count of an existing triangle mesh by collapsing edges that are inconsequential for the overall shape. Different metrics have been proposed to determine these collapse candidates, such as volume preservation [21] or an estimate of the distance to the original mesh [11]. Especially the algorithm by Garland and Heckbert [11] is part of most mesh-processing libraries. Both approaches use quadratic functions – quadrics – to determine how much a collapse would distort the original geometry. More recently, probabilistic versions of both algorithms have been proposed to handle noisy input data [28]. Although devised in the context of mesh simplification, quadrics have also found application in remeshing: Xu et al. [33] combine quadrics and CVT in an isotropic meshing approach. In contrast, our method uses anisotropy to achieve a more faithful representation. Our method is partly inspired by the aforementioned methods, but differs significantly in one aspect: To overcome the runtime limitation of pixel-based integration, we have to introduce the triangulation *before* normal integration. Hence, our method cannot rely on 3D geometry but can only use normal maps as input.

3. Screen Space Mesh Decimation

Transforming a regular pixel grid into an irregular 2D triangle mesh allows local adaptation to surface details. This can reduce runtime [14] while maintaining reconstruction quality. However, the creation of the triangulation requires careful consideration: First, the placement of vertices $v \in \mathcal{V}$

but also the choice of edges $e \in \mathcal{E}$ to connect them is of utmost importance for an accurate yet sparse representation. Second, the vertex and edge placement must occur without knowing the 3D surface in advance.

In this work, we advance the idea of triangulating surfaces based on normal maps by introducing an algorithm that decimates near-redundant vertices to achieve high compression ratios. To maintain high-accuracy, the remaining vertices and edges are aligned to the underlying geometry. Our algorithm is guided by the objective function:

$$E = E_{\text{Geo}} + \lambda \cdot E_{\text{ODT}}. \quad (1)$$

For an existing surface, the two energy terms can be written as surface integrals over $\vec{x} \in \mathbb{R}^3$. The first term estimates the deviation between the mesh and the underlying surface by relying on quadrics [11, 33]:

$$E_{\text{Geo}}(\vec{x}_1, \dots, \vec{x}_{|\mathcal{V}|}) = \sum_{v \in \mathcal{V}} \int_{\mathcal{S}_v} \langle \vec{n}(\vec{x}), \vec{x}_v - \vec{x} \rangle^2 d^2 x, \quad (2)$$

where \vec{x}_v for $v \in \mathcal{V}$ are the unknown 3D vertex positions and $\langle \cdot, \cdot \rangle$ is the standard scalar product. The star \mathcal{S}_v of v is the surface patch formed by all triangles touching v . The second term

$$E_{\text{ODT}}(\vec{x}_1, \dots, \vec{x}_{|\mathcal{V}|}) = \sum_{v \in \mathcal{V}} \int_{\mathcal{S}_v} \|\vec{x}_v - \vec{x}\|^2 d^2 x \quad (3)$$

guarantees a well-defined behaviour and an even vertex distribution in flat areas. We will see in Sec. 3.2 that E_{ODT} is closely connected to Delaunay triangulations [4]. This fundamental insight is crucial to align edges to features and to facilitate anisotropic meshing.

While it is easy to compute the energy terms for an existing surface in 3D, the depth coordinate of each \vec{x} is unknown in our case. A coarse 2D triangle mesh must be computed *before* the normal integration to gain any speed-up. In the following, we will show how both energy functions can be evaluated in screen space and derive a novel quadric formulation with normal maps as input. In Sec. 4, we focus on the practical aspects and implementation details. We break down the mathematical concepts into simple mesh operations, *i.e.* vertex relocation, edge flip and edge collapse.

3.1. Calculating the Objective Function On Screen

Image-based reconstruction can be understood as finding a projection

$$\phi : \Omega \rightarrow \mathbb{R}^3 \quad (4)$$

from the image foreground $\Omega \subset \mathbb{R}^2$ into 3D space. While existing approaches parametrise the projection through a pixel-wise depth map, we choose a 2D triangle mesh instead. The projection ϕ is then determined by one depth value z_v per vertex and bilinear interpolation everywhere

else. In Sec. 3.3, we derive these depth values from normal maps by solving the normal integration problem for a 2D triangle mesh, *c.f.* [14].

A unified treatment of E_{Geo} and E_{ODT} is obtained by introducing the norm

$$\|\vec{x}\|_M^2 := \langle \vec{x}, M\vec{x} \rangle \quad (5)$$

with a constant $M = \mathbb{1}$ for E_{ODT} and a locally varying $M(\vec{x}) = \vec{n}(\vec{x}) \cdot \vec{n}^t(\vec{x})$ for E_{Geo} . With this shorthand notation, both energy functions become a sum over

$$Q_v := \sum_{f \in \mathcal{F}} \int_f \|\vec{x}_v - \vec{x}\|_{M(\vec{x})}^2 d^2 x \quad (6)$$

for each vertex and with the respective choice for M . Such quadratic functions Q_v are commonly referred to as quadrics [11]. With a known projection ϕ , this integral could be evaluated in coordinates by substituting $\vec{x}_v - \vec{x} \rightarrow \phi(\vec{u}_v) - \phi(\vec{u})$ where \vec{u} and \vec{u}_p are the respective 2D coordinates on screen. However, runtime advantages are only achieved by decimating *before* the integration, *i.e.* before ϕ is known. Instead, we replace $\vec{x}_p - \vec{x} = J_f \cdot (\vec{u}_p - \vec{u})$ using a Taylor expansion where the face Jacobian J_f of ϕ is constant due to linear interpolation.

Unlike ϕ - which is only available after an expensive integration - the Jacobian J_f can be obtained directly from normals: The Jacobian $J = (\partial_u \phi, \partial_v \phi)$ consists of two surface tangents as columns. Both tangents are orthogonal to the surface normals and hence

$$\partial_u \phi = \vec{e}_x - \frac{n_x}{n_z} \cdot \vec{e}_z \quad \partial_v \phi = \vec{e}_y - \frac{n_y}{n_z} \cdot \vec{e}_z \quad (7)$$

in the orthographic case and

$$\partial_i \phi = \left(\partial_i \vec{r} - \frac{\langle \vec{n}, \partial_i \vec{r} \rangle}{\langle \vec{n}, \vec{r} \rangle} \cdot \vec{r} \right) \cdot z \quad \text{for } i \in \{u, v\} \quad (8)$$

in the perspective case where \vec{r} is the camera ray given by the intrinsics matrix, *cf. e.g.* [14]. In the latter case, we adopt the weak perspective projection and assume a constant camera-to-object distance. These two equations allow calculating tangent vectors from normals maps and we will require both equations throughout Sec. 4 to calculate normals for vertices, edges and faces.

To determine the influence of a vertex on the mesh surface and to facilitate finding optimal positions (Sec. 4.2), we need to know the change in Q_v if vertex v is moved by $\delta \vec{x}_v$:

$$Q_v(\delta \vec{x}_v) := \sum_{f \in \mathcal{F}_v} \int_f \|J_f(\vec{u}_v - \vec{u}) + \delta \vec{x}_v\|_{M(\vec{x})}^2 d\Omega, \quad (9)$$

where $d\Omega = |J_f^t J_f|^{\frac{1}{2}} d^2 u$ compensates for the distorted area on the screen due to foreshortening. This ensures that all faces contribute as if the integral was evaluated over the 3D surface.

3.2. Generalized Delaunay Triangulations

To achieve accurate mesh representations despite reducing the number of vertices, we must align edges and vertices to ridges and furrows of the underlying surface. Notably, E_{ODT} in Eq. (3) is closely linked to Delaunay triangulations: For a point set in 2D, the associated Delaunay triangulation minimizes E_{ODT} . As we have seen in Sec. 3.1, both E_{Geo} and E_{ODT} can be written in terms of a norm $\|\vec{x}\|_M^2 = \langle \vec{x}, M\vec{x} \rangle$ with the respective choices for M . We can combine E_{Geo} and E_{ODT} into a single norm with $M = \vec{n} \cdot \vec{n}^t + \lambda \cdot \mathbb{1}$. In this way, our objective function is an extension of E_{ODT} with a more general norm. Replacing $\|\cdot\| \rightarrow \|\cdot\|_M$ to generalize Delaunay triangulations in 2D was previously suggested [4]. However, the presented algorithm is based on convex hulls and does not extend to surfaces in 3D or locally varying $M \in \mathbb{R}^{3 \times 3}$. In Sec. 4.3, we will present a local version of this algorithm that extends the original idea to surfaces in 3D and with varying $M \in \mathbb{R}^{3 \times 3}$. Furthermore, we show how this local version can be computed in screen space.

3.3. Normal Integration

A unified description of the orthographic and perspective integration problem is achieved by minimising the functional

$$E_{\text{Int}} = \int_{\Omega} \left\| \langle \vec{n}, \vec{r} \rangle \cdot \begin{pmatrix} \partial_u z \\ \partial_v z \end{pmatrix} + D \cdot \begin{pmatrix} n_x \\ n_y \end{pmatrix} \right\|^2 d^2 u. \quad (10)$$

over the depth map $z : \Omega \rightarrow \mathbb{R}$ [3, 25]. For the orthographic projection, $\vec{r} = \vec{e}_z$ is the unit vector in the z -direction and $D = 1$. For the perspective projection, \vec{r} is the camera ray given by the intrinsics matrix and D is the inverse of the focal length. Furthermore, we need to take the exponential of z to obtain the final depth map in the perspective case.

Only recently, a discretisation of Eq. (10) for triangle meshes has been proposed [14]. This mesh version of the normal integration problem resembles the seminal Cotan-Laplacian [24] in geometry processing. As for pixel-wise integration, it leads to a sparse linear system but replaces pixel neighbourhoods with adjacent vertices. Please refer to the supplementary material for further discussion of the nuances of mesh-based integration.

4. Algorithm

Our algorithm decimates an existing triangle mesh, Fig. 2, by iteratively repeating the following three steps: We collapse edges (Sec. 4.4) to remove redundant triangles in smooth, featureless regions, we update vertex positions (Sec. 4.2) and flip edges (Sec. 4.3) to align better with ridges and furrows of the surface. The triangle mesh is initialized by converting each foreground pixel into two triangles.

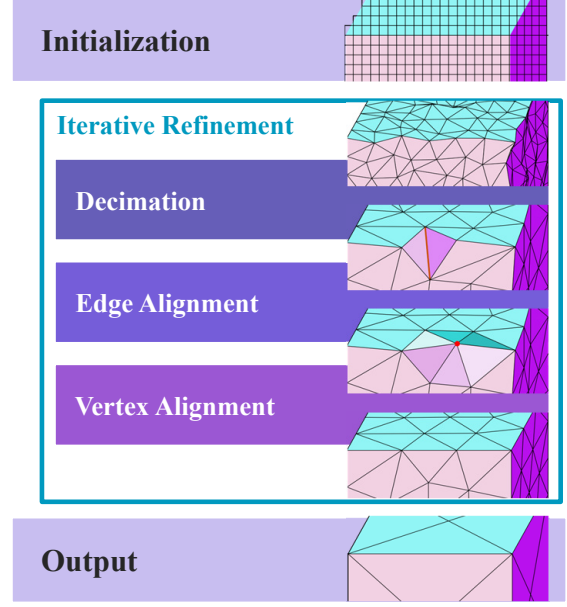


Figure 2. Schematic representation of our iterative screen space decimation: Decimation drastically reduces the number of vertices but may introduce misaligned edges and vertices which are fixed by our edge and vertex alignment procedures.

4.1. Quadrics from Normal Maps

Calculating the screen space quadrics is a two-step procedure. First, we calculate face normals for each triangle by

$$\vec{n}_f := \text{normalize} \left(\sum_{p \in \mathcal{P}_f} \vec{n}_p \right) \quad (11)$$

where \mathcal{P}_f contains all pixels covered by triangle f and \vec{n}_p are the input pixel normals. From the face normals \vec{n}_f , we calculate the face Jacobians J_f using Eqs. (7) and (8) depending on the projection.

Secondly, we calculate the quadrics

$$Q_v(\delta\vec{x}) = \sum_{f \in \mathcal{F}_v} \frac{A_f^{(3)}}{|\mathcal{P}_f|} \sum_{p \in \mathcal{P}_f} \|J_f \delta\vec{u}_{vp} + \delta\vec{x}\|_{M_p}^2, \quad (12)$$

where $\delta\vec{u}_{vp} = \vec{u}_v - \vec{u}_p$ is the vector on screen pointing from the pixel to the vertex. The norm $\|\cdot\|_{M_p}$ at pixel p is given through the matrix

$$M_p := \vec{n}_p \cdot \vec{n}_p^t + \lambda \cdot \mathbb{1}, \quad (13)$$

where \vec{n}_p the normal at pixel p . This matrix combines the contributions from E_{Geo} and E_{ODT} . The unforeshortened area $A_f^{(3)} = |J_f^t J_f|^{\frac{1}{2}} \cdot A_f^{(2)}$ of f can be calculated from the triangle area $A_f^{(2)}$ on screen by using the Jacobian.

4.2. Vertex Alignment

The key idea behind our vertex alignment algorithm is to place vertices along ridges and furrows of the surface to

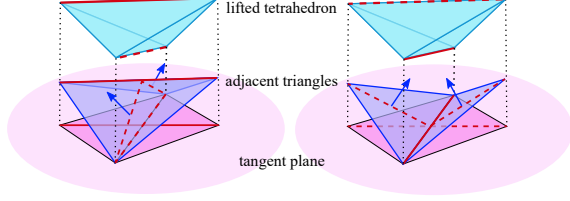


Figure 3. Schematic representation of our edge flip criterion: Each triangle patch is projected into its tangent plane. We then use $\|\cdot\|_{M_e}$ to lift the flattened patch. Among the two diagonals of the patch, the aligned edge is the lower edge in the resulting tetrahedron.

achieve a close representation even at low mesh resolutions. To translate a vertex displacement $\delta\vec{u}_v$ on screen into a vertex displacement $\delta\vec{x}_v$ in 3D space, we assume that the vertex is moving along the surface, *i.e.* in the tangent space. We define vertex normals

$$\vec{n}_v := \text{normalize} \left(\sum_{f \in \mathcal{F}_v} A_f^{(3)} \cdot \vec{n}_f \right), \quad (14)$$

as the area-weighted average of the adjacent face normals and use Eqs. (7) and (8) to calculate the Jacobian J_v that characterizes the tangent space. Then, moving the vertex by $\delta\vec{u}_v$ on screen causes a displacement $\delta\vec{x}_v = J_v \delta\vec{u}_v$ in 3D space. To move the vertex to its optimal position, we minimise

$$\tilde{Q}_v(\delta\vec{u}_v) := Q_v(J_v \delta\vec{u}_v). \quad (15)$$

Since quadrics are quadratic functions, the minimizer is found by solving a linear system. Finally, we apply the displacement

$$\vec{u}_v \leftarrow \vec{u}_v + \alpha \cdot \delta\vec{u}_v \quad \text{for } v \in \mathcal{V} \quad (16)$$

scaled by a step width $\alpha = 0.5$.

4.3. Edge Alignment

Aside from placing vertices along ridges and furrows of the surface, it is equally important to align the triangle edges with these. We identify and flip non-aligned edges using the theory of generalized Delaunay triangulations in 2D, see Sec. 3.2. For each non-boundary edge e , we consider the patch made of its two adjacent triangles f, f' . As for the vertices, we assign a normal

$$\vec{n}_e := \text{normalize} \left(A_f^{(3)} \cdot \vec{n}_f + A_{f'}^{(3)} \cdot \vec{n}_{f'} \right) \quad (17)$$

to the edge. Similarly, we approximate the quadric in the patch through

$$M_e = \frac{A_f^{(3)}}{|\mathcal{P}_f|} \sum_{p \in \mathcal{P}_f} M_p + \frac{A_{f'}^{(3)}}{|\mathcal{P}_{f'}|} \sum_{p \in \mathcal{P}_{f'}} M_p. \quad (18)$$

By projecting the four vertices forming the patch into the tangent plane, we obtain a flattened approximation of the patch. Similarly to the 2D generalized Delaunay triangulation, we use $\|\cdot\|_{M_e}$ to lift the vertices out of the plane to form a tetrahedron, see Fig. 3. Different to the 2D version, our localized version is free of any expensive convex hull calculations. Instead, the current edge is geometry-aligned if it is *lower* in the tetrahedron than the other diagonal of the two adjacent triangles. If this is not the case, we perform an edge flip.

4.4. Decimation

We remove vertices that least influence the object’s overall surface through edge collapses [11]. During a collapse the edge (v, w) between the vertices v and w is contracted into a single vertex. We judge the influence of a collapse on the overall shape by calculating a cost

$$C_{vw} := \min_{\vec{u}_{vw}} \tilde{Q}_v(\vec{u} - \vec{u}_v) + \tilde{Q}_w(\vec{u} - \vec{u}_w), \quad (19)$$

i.e. the joint value of the screen-quadrics Eq. (15) of the edge endpoints when contracted into an optimally positioned vertex at \vec{u}_{vw} . We restrict \vec{u}_{vw} to the edge itself, *i.e.* the solution is found by solving a 1D linear system.

All collapsible edges are placed in a priority queue with non-decreasing C_{vw} . During a collapse, we contract the edge into the optimal position \vec{u}_{vw} . After the collapse, the quadric for this newly positioned vertex needs to be calculated and the optimal collapses for adjacent edges need to be updated. In line with previous work [11], the quadric at the new vertex is approximated as the sum of the two just removed vertex quadrics, instead of evaluating the computationally more involved Eq. (12).

4.5. Implementation Details

Our decimation algorithm is written in C++ and relies on SURFACEMESH [26] to represent triangle meshes. We employ the NVDIFFRAST renderer [18] to translate between triangles and the pixel grid. All linear systems are solved using EIGEN [12]. We present two ways to control the mesh resolution: Either by setting the vertex target or by providing a threshold on edge collapses, *cf.* Eq. (19). In both cases, we perform five iterations of decimation each followed by edge- and vertex-alignment. If an error threshold is provided, each decimation step will perform all edge collapses with a cost below this threshold. If a vertex target is provided, we exponentially decrease the vertex count over five iterations starting at ten times the vertex target. We use $\lambda = 10^{-5}$ and $\alpha = 0.5$ in all our experiments. Code is available at <https://moritzzheep.github.io/anisotropic-screen-meshing>.

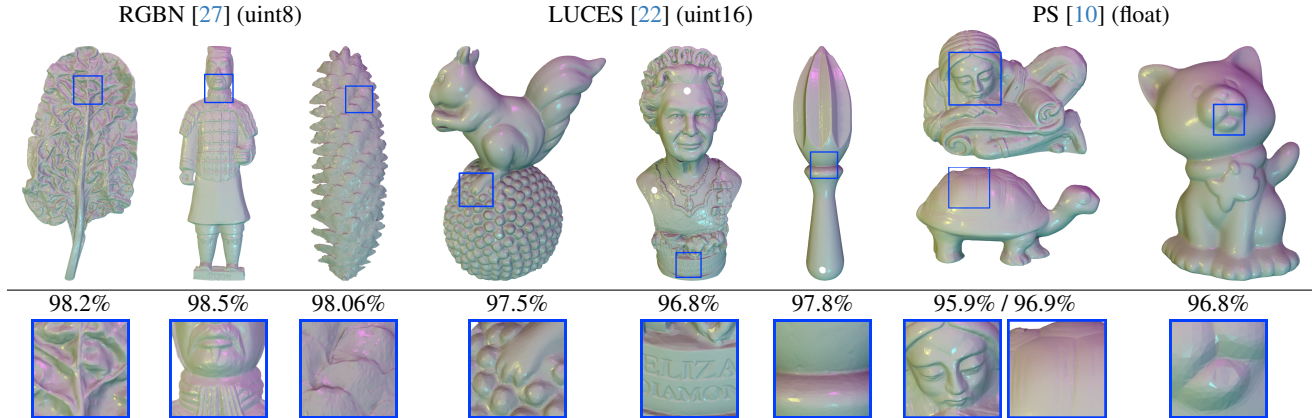


Figure 4. Results from testing different datasets and normal map discretisations. Even with compression rates beyond 95%, fine details are well preserved and visible. The compression ratios refer to the number of depth variables and do not reflect file size. Any holes within objects are subject to the provided masks of the dataset.

5. Evaluation

We test our algorithm, quantitatively and qualitatively on four publicly available datasets [10, 20, 22, 27] and additional synthetic data. The DiLiGenT-MV [20] dataset contains 5 objects from 20 different views and LUCES [22] consists of 14 objects. We observe that these datasets are either low resolution [20, 22] or lack ground-truth geometry [10, 27]. We mitigate this issue by complementing these published datasets with rendered normal maps. Figure 4 illustrates some results on a range of datasets at varying discretisation accuracy: floating points [10, 20], 16-bit [22] or 8-bit unsigned integers [27]. Despite over 95% fewer vertices than pixels in the input normal map, fine surface details remain clearly visible. Further results can be found in the supplementary material.

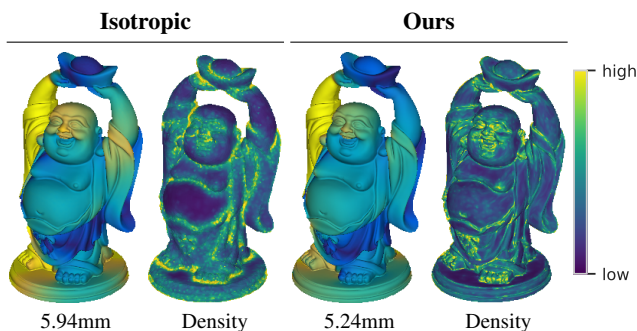


Figure 5. Absolute error and vertex density for isotropic remeshing (left) and our decimation (right). The isotropic algorithm places more vertices around ridges and furrows. Our algorithm places fewer vertices in these regions but still obtains a lower error due to the careful vertex and edge alignment.

5.1. Benchmark Comparisons

As a first step, we compare our results to the only previous mesh-based integration method [14] we are aware of. For an even comparison, we set our vertex target to match the numbers for the low-, mid- and high-resolution settings reported in [14]. Tab. 1 lists a summary of the root mean square error (RMSE) over all 20 views for DiLiGenT-MV. For reference, we also list the results of the pixel-based method in [2] which performed best in our tests. The RMSE is always measured after a non-rigid alignment to absolve the inherent scale ambiguity in normal integration. Our method reliably outperforms the isotropic meshing and even gets within sub-millimetres of the pixel-based approach for the 'high' setting, where the number of vertices is typically still 10 times smaller than the number of pixels. Figure 6 displays the visual results at 'mid' resolution together with the RMSE and mean angular error (MAE), which confirm the advantages of our method. Comparing the vertex density images of both methods in Fig. 5, we identify that our method concentrates vertices directly at ridges and furrows, while the isotropic method [14] creates a much more smoothly varying vertex density. The isotropic approach trades isotropy

	[2]	Isotropic [14]			Ours		
Dataset	Ref	low	mid	high	low	mid	high
BEAR	2.97	3.95	3.65	3.37	3.84	3.38	3.04
BUDDHA	6.74	7.74	7.54	7.33	6.86	6.68	6.61
COW	2.45	3.42	3.12	2.96	3.07	2.85	2.74
POT2	5.15	5.89	5.77	5.65	5.63	5.47	5.29
READING	6.34	7.08	6.93	6.83	6.82	6.67	6.50

Table 1. Average RMSE over all 20 views of the DiLiGenT-MV dataset in mm using orthographic projection.

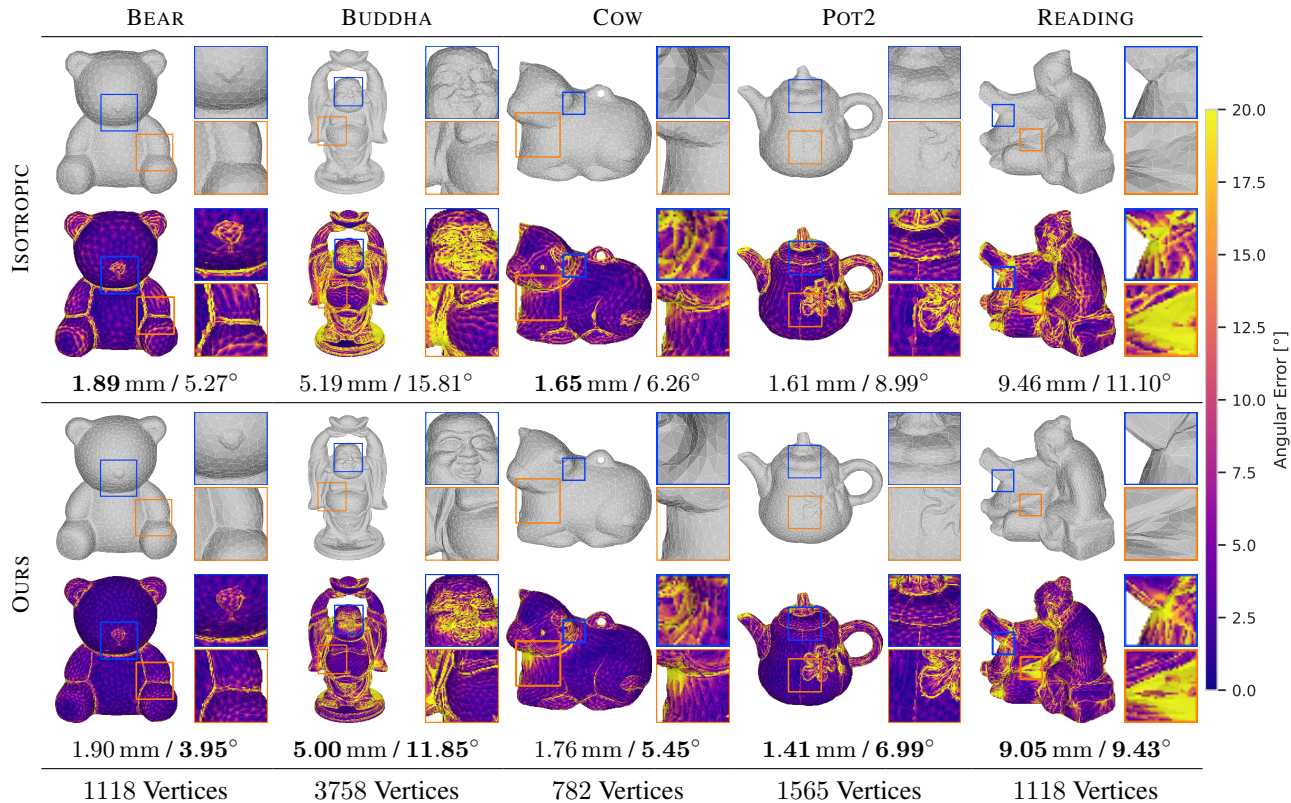


Figure 6. Comparison between isotropic [14] (top) and our anisotropic meshes (bottom), both in screen space for the DiLiGenT-MV dataset. We show wireframes (as vector graphics), angular error maps and report RMSE and MAE as error metrics. With the same number of vertices, we outperform both locally (MAE) and globally (RMSE) and preserves details better, e.g. the bear’s nose or the Buddha’s face.

for accuracy.

5.2. Controllability

For a second evaluation, we use the 14 objects in the LUCES dataset and control the mesh resolution by providing three decimation thresholds, cf. Eq. (19). The results are listed in Tab. 2. LUCES is a challenging dataset as objects are placed very close to the camera. Nonetheless, a lower decimation threshold correlates with a lower reconstruction error. As for DiLiGenT-MV, the highest mesh resolution is within the submillimetre range of the pixel-based baseline [13]. Additional results are in the supplementary material.

5.3. Ablation Studies

For an ablation study, we consider the three operations of our algorithm, see Fig. 7: Decimation alone, decimation with either vertex alignment or edge alignment as well as the full algorithm. Decimation alone preserves the rough shape but struggles with high-frequency details. Adding either vertex- or edge-alignment improves RMSE and MAE respectively but only the full algorithm leads to the best results both qualitatively and quantitatively.

5.4. Runtime

To evaluate the runtime at higher resolutions, we created synthetic normals maps from high-resolution 3D scans with

Dataset	Ref	Threshold		
	[13]	2	2^6	2^{11}
BELL	0.30	0.33	0.49	0.72
BUDDHA	3.46	3.58	3.51	3.32
BUNNY	3.38	3.67	3.83	4.15
DIE	1.62	1.65	2.11	3.09
HIPPO	2.73	2.85	2.91	3.09
JAR	0.50	0.45	0.45	0.46
OWL	4.89	5.41	5.63	5.94
QUEEN	3.74	4.02	4.41	4.98
SQUIRREL	1.91	2.07	2.37	3.00
TOOL	0.91	0.98	0.98	1.04

Table 2. RMSE in mm for some of the objects in LUCES [22] with increasing decimation threshold, see Eq. (19). The chosen thresholds should lead to a constant reduction rate of the RMSE.

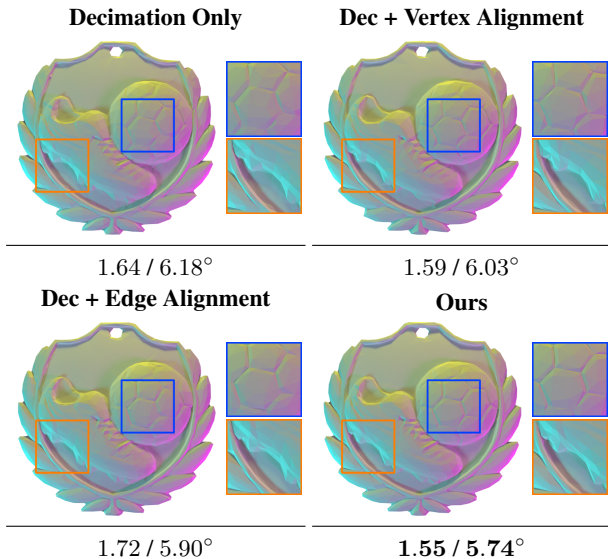


Figure 7. Ablation study of the three components of our method: Results for a vertex resolution of 1k using only decimation, decimation and vertex alignment, decimation and edge alignment as well as our full algorithm.

resolutions ranging from 512^2 to 8192^2 and applied the method in [14] and our method with a constant quality setting. Compared to the simpler isotropic remeshing [14], the improved accuracy of our anisotropic approach has the downside of slightly higher runtimes. Nevertheless, our method is still considerably faster than pixel-wise integration, especially for increasing resolutions. According to Fig. 8, the overhead of converting pixels to meshes *before* the integration already pays off at resolutions as low as 1MP. At resolutions beyond 10MP, mesh-based methods are up to 100 times faster than traditional pixel-based integration.

5.5. Robustness

Given that noise might be present in experimentally obtained normal maps, we tested the robustness of our method towards imperfect inputs by distorting the ground-truth normal maps with Gaussian noise of various amplitudes, see Fig. 9. As expected, the quality of the input is reflected in the quality of the output. However, our vertex and edge alignment to ridges and furrows remains consistent even at high levels of noise.

6. Conclusion and Limitations

We proposed a screen space decimation approach for mesh-based integration. Our results show that careful alignment of vertices and edges to ridges and furrows of the underlying surface is key to surpassing the quality of previous methods and maintaining high geometric faithfulness even at high compression ratios. Conversely, we achieve compa-

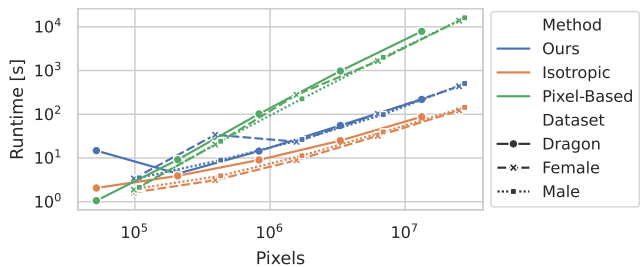


Figure 8. Total runtime (meshing and surface integration) as a function of the number of foreground pixels. For the pixel-based method it is only integration time. Please note the log-log scale.

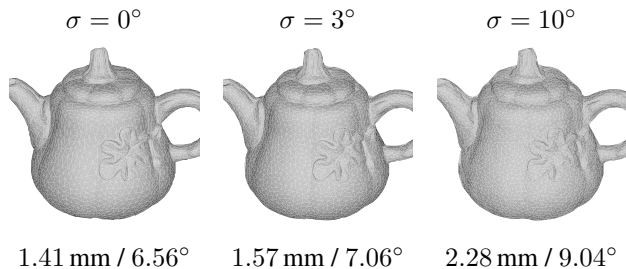


Figure 9. Robustness against increasing levels of normally distributed noise: Numbers are RMSE and MAE respectively. All meshes have a 2k vertices and are vector graphics to allow close examination in the digital version.

table results to pixel-based methods at moderate compression ratios. Unlike previous approaches, our method offers greater control over the output. This can be achieved by either setting an error threshold to reach a desired quality or by defining a fixed number of vertices to stay within limitations like GPU memory. All things considered, we presented a versatile approach that allows balancing runtime and quality and can be adjusted to the needs of almost any photometric stereo pipeline.

As with all single-view reconstruction approaches, normal integration suffers from an inherent scale ambiguity for the final geometry. Combining our approach with multi-view reconstruction could overcome this limitation and is an interesting direction for future research. Furthermore, our method is currently designed for continuous surfaces. The major challenge in introducing discontinuities into mesh-based integration so far has been aligning edges with discontinuities. It seems that our alignment strategies to geometric details are a stepping stone in this direction.

Acknowledgements The "David Head" by [1d.inc](#) and the "Football Medal2 - PhotoCatch" by [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).

References

- [1] Pierre Alliez, Éric Colin de Verdière, Olivier Devillers, and Martin Isenburg. Centroidal Voronoi diagrams for isotropic surface remeshing. *Graphical Models*, 67(3):204–231, 2005. 2
- [2] Xu Cao, Boxin Shi, Okura Fumio, and Yasuyuki Matsushita. Normal Integration via Inverse Plane Fitting With Minimum Point-to-Plane Distance. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2382–2391, 2021. 2, 6
- [3] Xu Cao, Hiroaki Santo, Boxin Shi, Fumio Okura, and Yasuyuki Matsushita. Bilateral Normal Integration. In *European Conference on Computer Vision, (ECCV)*, pages 552–567. Springer Nature Switzerland, Cham, 2022. 2, 4
- [4] Long Chen. Mesh Smoothing Schemes Based on Optimal Delaunay Triangulations. In *Proc. of International Meshing Roundtable (IMR)*, pages 109–120. Citeseer, 2004. 2, 3, 4
- [5] Zhonggui Chen, Juan Cao, and Wenping Wang. Isotropic Surface Remeshing Using Constrained Centroidal Delaunay Mesh. *Computer Graphics Forum*, 31(7):2077–2085, 2012. 2
- [6] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review*, 41(4):637–676, 1999. 2
- [7] Zhouyu Du, Antonio Robles-Kelly, and Fangfang Lu. Robust Surface Reconstruction from Gradient Field Using the L1 Norm. In *Conf. on Digital Image Computing Techniques and Applications (DICTA 2007)*, pages 203–209, 2007. 1, 2
- [8] Marion Dunyach, David Vanderhaeghe, Loïc Barthe, and Mario Botsch. Adaptive remeshing for real-time mesh deformation. In *Eurographics 2013 - Short Papers*. The Eurographics Association, 2013. 2
- [9] Jean-Denis Durou and Frédéric Courteille. Integration of a Normal Field without Boundary Condition. In *Proc. of the Workshop on Photometric Analysis For Computer Vision-PACV 2007*, pages 8–p. INRIA, 2007. 2
- [10] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(4):439–451, 1988. 6
- [11] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 209–216. ACM Press, 1997. 1, 2, 3, 5
- [12] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3, 2010. 5
- [13] Moritz Heep and Eduard Zell. ShadowPatch: Shadow Based Segmentation for Reliable Depth Discontinuities in Photometric Stereo. *Computer Graphics Forum*, 41(7):635–646, 2022. 2, 7
- [14] Moritz Heep and Eduard Zell. An Adaptive Screen-Space Meshing Approach for Normal Integration. In *European Conference on Computer Vision, (ECCV)*, pages 445–461, Milan, Italy, 2025. Springer. 1, 2, 3, 4, 6, 7, 8
- [15] Berthold Klaus Paul Horn. *Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View*. PhD thesis, Massachusetts Institute of Technology, Cambridge, 1970. 1
- [16] Berthold Klaus Paul Horn and Michael J. Brooks. The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33(2):174–208, 1986. 2
- [17] Dawar Khan, Alexander Plopski, Yuichiro Fujimoto, Masayuki Kanbara, Gul Jabeen, Yongjie Jessica Zhang, Xiaopeng Zhang, and Hirokazu Kato. Surface remeshing: A systematic literature review of methods and research directions. *IEEE Trans. on Visualization and Computer Graphics*, 28(3):1680–1713, 2020. 2
- [18] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Trans. on Graphics*, 39(6):1–14, 2020. 5
- [19] Bruno Lévy and Nicolas Bonneel. Variational Anisotropic Surface Meshing with Voronoi Parallel Linear Enumeration. In *Proc. of the International Meshing Roundtable (IMR)*, pages 349–366. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. 2
- [20] Min Li, Zhenglong Zhou, Zhe Wu, Boxin Shi, Changyu Diao, and Ping Tan. Multi-view photometric stereo: A robust solution and benchmark dataset for spatially varying isotropic materials. *IEEE Trans. on Image Processing*, 29:4159–4173, 2020. 6
- [21] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *9th IEEE Visualization Conference (VIZ)*, pages 279–286, 1998. 2
- [22] Roberto Mecca, Fotios Logothetis, Ignas Budvytis, and Roberto Cipolla. LUCES: A Dataset for Near-Field Point Light Source Photometric Stereo. In *32nd British Machine Vision Conference (BMVC)*, 2021. 6, 7
- [23] Vincent Nivoliers, Bruno Lévy, and Christophe Geuzaine. Anisotropic and feature sensitive triangular remeshing using normal lifting. *Journal of Computational and Applied Mathematics*, 289:225–240, 2015. 2
- [24] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993. 4
- [25] Yvain Quéau, Jean-Denis Durou, and Jean-François Aujol. Normal Integration: A Survey. *Journal of Mathematical Imaging and Vision*, 60(4):576–593, 2018. 2, 4
- [26] Daniel Sieger and Mario Botsch. The Polygon Mesh Processing Library, 2023. <https://github.com/pmp-library/pmp-library>. 5
- [27] Corey Toler-Franklin, Adam Finkelstein, and Szymon Rusinkiewicz. Illustration of complex real-world objects using images with normals. In *Proc. of Int. Symposium on Non-photorealistic Animation and Rendering NPAR*, pages 111–119, San Diego California, 2007. ACM. 6
- [28] P. Trettner and L. Kobbelt. Fast and Robust QEF Minimization using Probabilistic Quadrics. *Computer Graphics Forum*, 39(2):325–334, 2020. 2
- [29] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):191139, 1980. 1
- [30] Wuyuan Xie, Yunbo Zhang, Charlie CL Wang, and Ronald C.-K. Chung. Surface-from-gradients: An approach based on discrete geometry processing. In *Proc. of the IEEE*

- Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2195–2202, 2014. [2](#)
- [31] Wuyuan Xie, Chengkai Dai, and Charlie CL Wang. Photometric stereo with near point lighting: A solution by mesh deformation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4585–4593, 2015. [2](#)
- [32] Wuyuan Xie, Miaohui Wang, Mingqiang Wei, Jianmin Jiang, and Jing Qin. Surface Reconstruction from Normals: A Robust DGP-based Discontinuity Preservation Approach. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5328–5336, 2019. [2](#)
- [33] Rui Xu, Longdu Liu, Ningna Wang, Shuangmin Chen, Shiqing Xin, Xiaohu Guo, Zichun Zhong, Taku Komura, Wenping Wang, and Changhe Tu. CWF: Consolidating Weak Features in High-quality Mesh Simplification. *ACM Trans. on Graphics*, 43(4):80:1–80:14, 2024. [2](#), [3](#)
- [34] Zichun Zhong, Xiaohu Guo, Wenping Wang, Bruno Lévy, Feng Sun, Yang Liu, and Weihua Mao. Particle-based anisotropic surface meshing. *ACM Trans. on Graphics*, 32(4):99–1, 2013. [2](#)
- [35] Mingyuan Zhou, Yuqi Ding, Yu Ji, S. Susan Young, Jingyi Yu, and Jinwei Ye. Shape and Reflectance Reconstruction using Concentric Multi-Spectral Light Field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 42(7):1594–1605, 2020. [2](#)
- [36] Dizhong Zhu and William A. P. Smith. Least Squares Surface Reconstruction on Arbitrary Domains. In *European Conference on Computer Vision, (ECCV)*, pages 530–545. Springer International Publishing, Cham, 2020. [2](#)

Feature-Preserving Mesh Decimation for Normal Integration

– Supplementary Material –

Moritz Heep
PhenoRob
University of Bonn
mheep@uni-bonn.de

Sven Behnke
Autonomous Intelligent Systems
University of Bonn
behnke@ais.uni-bonn.de

Eduard Zell
Independent Researcher
ezell@hotmail.de

Overview

This supplementary material, provides additional technical details on the normal integration, the final algorithm and additional evaluation results that have been removed from the main document due to page constraints. For easier navigation and cross-referencing, we follow the section titles of the submitted paper. For a quick overview, we briefly list the content of the Appendix:

- Appendix A provides additional details on the integration, improving further the surface reconstruction accuracy compared to previous work[5].
- Appendix B describes additional details on the computation of quadrics and the linear system to solve.
- Appendix C provides additional comparisons to previous work, quantitative analysis on the reconstruction accuracy and qualitative examples.
- Appendix D is a list of all the datasets we used together with the urls to find them.

Finally, we included samples of the reconstructed surfaces using our method (`ours.obj`) and using [5] (`adaptive.obj`) for

- Michelangelo’s David (Fig. 1 of the manuscript) and
- all five objects in the DiLiGenT-MV [6] dataset (reconstructed from the first view each).

These samples can be found in the respective subdirectories of `meshes/` of this supplementary material.

A reference implementation of our method is available under <https://moritzheep.github.io/anisotropic-screen-meshing>.

A. Screen Space Mesh Decimation

In the following, we will provide more detail on our mesh-based integration and the underlying sparse linear system, that slightly differs from [5] but improves the reconstruction accuracy even further.

A.1. Details on Mesh-Based Normal Integration

The unified functional for normal integration is

$$E_{\text{Int}} = \int_{\Omega} \left\| \langle \vec{n}, \vec{r} \rangle \cdot \begin{pmatrix} \partial_u z \\ \partial_v z \end{pmatrix} + D \cdot \begin{pmatrix} n_x \\ n_y \end{pmatrix} \right\|^2 d^2 u. \quad (1)$$

with the respective choices for \vec{r} and D depending on the projection type – orthographic or perspective. In the case of pixel-based integration, the partial derivatives are discretised by using each pixel and its immediate neighbourhood. In the case of mesh-based integration, the pixel neighbourhood is replaced by adjacent vertices.

Previous work [5] showed that the minimiser of E_{Int} is found by solving the sparse linear system

$$\sum_{w \in \mathcal{V}_v} \sum_{f \in \mathcal{F}(v,w)} \omega_{f,vw} \cdot (m_f \cdot \delta z_{vw} + \langle \vec{b}_f, \delta \vec{u}_{vw} \rangle) = 0 \quad (2)$$

for each vertex $v \in \mathcal{V}$. All vertices w that are connected by an edge with vertex v have an impact on depth z_v . We denote these vertices by \mathcal{V}_v . The faces adjacent to the edge (v, w) , denoted as $\mathcal{F}_{u,v}$, define the strength of the influence of w . The edge weights $\omega_{f,vw} := \cot(\alpha_{f,vw})$ are given by the cotangent of the angle in f that is opposite to (v, w) . The weights are identical to those of the Cotan-Laplacian [8]. Together with the two parameters [5]

$$m_f = \int_f \langle \vec{r}, \vec{n} \rangle^2 d\Omega \quad (3)$$

$$\vec{b}_f = D \cdot \int_f \langle \vec{r}, \vec{n} \rangle \cdot \begin{pmatrix} n_x \\ n_y \end{pmatrix} d\Omega \quad (4)$$

they ensure that the integration yields the same results for different triangulations. Previously, these parameters have been calculated by assuming a constant face normal \vec{n}_f . In contrast, we found that

$$m_f = \frac{1}{|\mathcal{P}_f|} \sum_{p \in \mathcal{P}_f} \langle \vec{r}_p, \vec{n}_p \rangle^2 \quad (5)$$

$$\vec{b}_f = \frac{D}{|\mathcal{P}_f|} \langle \vec{r}_p, \vec{n}_p \rangle \cdot \begin{pmatrix} n_x \\ n_y \end{pmatrix}_p \quad (6)$$

		[1]	Isotropic [5]			Our Decimation, [5]’s Integration			Ours		
	Dataset	Ref	low	mid	high	low	mid	high	low	mid	high
Orthographic	Bear	2.97	3.95	3.65	3.37	3.67	3.33	3.19	3.84	3.38	3.04
	Buddha	6.74	7.74	7.54	7.33	7.30	7.10	7.08	6.86	6.68	6.61
	Cow	2.45	3.42	3.12	2.96	3.23	3.00	2.86	3.07	2.85	2.74
	Pot2	5.15	5.89	5.77	5.65	5.72	5.59	5.48	5.63	5.47	5.29
	Reading	6.34	7.08	6.93	6.83	6.88	6.76	6.64	6.82	6.67	6.50
Perspective	Bear	2.91	3.94	3.72	3.47	3.64	3.48	3.33	3.69	3.22	2.90
	Buddha	6.75	7.74	7.53	7.40	7.31	7.13	7.09	6.83	6.68	6.62
	Cow	2.35	3.49	3.24	3.07	3.29	3.09	2.99	2.97	2.77	2.63
	Pot2	4.99	6.04	5.86	5.76	5.81	5.69	5.61	5.48	5.32	5.16
	Reading	6.28	7.19	6.94	6.85	6.91	6.77	6.69	6.74	6.52	6.45

Table 1. Comparison of the average RMSE over all 20 views of DiLiGenT-MV: Isotropic remeshing and integration from [5], our decimation combined with the integrator of [5] and our decimation with our integrator. Our finer approximation of the integration parameters yields to a tighter approximation of the underlying surface.

is a tighter approximation that considers variations of the normals within the faces and yields more accurate surface integrations, see Tab. 1.

B. Algorithm

In this section, we want to express the quadrics in a more familiar form of a quadratic function and derive the linear system to solve during vertex alignment.

B.1. Explicit Form of the Quadrics

In Eq. (9) of the main paper, we defined the quadric for the continuous case as follows:

$$Q_v(\delta\vec{x}_v) := \sum_{f \in \mathcal{F}_v} \int_f \|J_f(\vec{u}_v - \vec{u}) + \delta\vec{x}_v\|_{M(\vec{x})}^2 d\Omega, \quad (7)$$

where both energy terms can be unified into one term by the norm induced by

$$M(\vec{x}) = \vec{n}(\vec{x}) \cdot \vec{n}(\vec{x})^t + \lambda \cdot \mathbf{1}, \quad (8)$$

cf. Eq. (13) of the main work. To obtain the known form of a quadratic problem,

$$Q_v(\delta\vec{x}_v) = \langle \delta\vec{x}_v, A_v \delta\vec{x}_v \rangle - 2 \langle \vec{b}_v, \delta\vec{x}_v \rangle + c_v \quad (9)$$

we have to apply the binomial formula to the integrand and rearrange the addends. In the end, we get

$$A_v = \sum_{f \in \mathcal{F}_v} \int_f M(\vec{x}) d\Omega \quad (10)$$

for the quadratic part,

$$\vec{b}_v = \sum_{f \in \mathcal{F}_v} \int_f M(\vec{x}) \cdot J_f(\vec{u}_v - \vec{u}) d\Omega$$

for the linear part and

$$c_v = \sum_{f \in \mathcal{F}_v} \int_f \|J_f(\vec{u}_v - \vec{u})\|_{M(\vec{x})} d\Omega \quad (11)$$

for the constant part.

In the discretised version (Eq. 12 of the main document), we replace the integral by a sum:

$$Q_v(\delta\vec{x}) = \sum_{f \in \mathcal{F}_v} \frac{A_f^{(3)}}{|\mathcal{P}_f|} \sum_{p \in \mathcal{P}_f} \|J_f \delta\vec{u}_{vp} + \delta\vec{x}\|_{M_p}^2. \quad (12)$$

Similarly, the integral is replaced by a sum in the coefficients of the linear system:

$$A_v = \sum_{f \in \mathcal{F}_v} \frac{A_f^{(3)}}{|\mathcal{P}_f|} M_p, \quad (13)$$

$$\vec{b}_f = \sum_{f \in \mathcal{F}_v} \frac{A_f^{(3)}}{|\mathcal{P}_f|} \cdot M_p \cdot J_f(\vec{u}_v - \vec{u}), \quad (14)$$

$$c_f = \sum_{f \in \mathcal{F}_v} \frac{A_f^{(3)}}{|\mathcal{P}_f|} \|J_f(\vec{u}_v - \vec{u})\|_{M_p}, \quad (15)$$

where $M_p = \vec{n}_p \cdot \vec{n}_p^t + \lambda \mathbf{1}$ from the pixel normals \vec{n}_p . However, when solving for the optimal vertex positions, we consider the following quadric:

$$\tilde{Q}_v(\delta\vec{u}_v) := Q_v(J_f \cdot \delta\vec{x}_v) \quad (16)$$

which is now in \mathbb{R}^2 since $J_f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$. Hence, we must replace $\delta\vec{x} \mapsto J_f \cdot \delta\vec{x}_v$ in Eq. (9). As a result, the coefficients

of this quadratic function are:

$$\tilde{A}_v = J_f^t A_v J_f, \quad (17)$$

$$\tilde{b}_f = J_f^t \vec{b}_v, \quad (18)$$

$$\tilde{c}_v = c_v. \quad (19)$$

To find the final displacement $\delta \vec{u}_v$ that moves vertex v into its optimal position, we solve

$$\tilde{A}_v \delta \vec{u} = \tilde{b}_v. \quad (20)$$

By doing so, we neglect the influence of a vertex displacement on the adjacent vertices and their quadrics, which is a common approximation in mesh-processing [2, 10].

C. Evaluation

In this section, we provide additional benchmark results and insights on how the compression ratio and reconstruction error depend on the user-set decimation threshold. Further error maps are found in Appendix C.3. We also discuss two interesting outliers: One, where our anisotropic decimation method is more accurate than the pixel-based reference and one, where a higher decimation threshold, *i.e.* lower resolution mesh, surpasses higher resolutions. At the end of this section, we depict reconstructions of all objects of the LUCES [7], RGBN [9] and PS [3] dataset for various values of the decimation threshold and report vertex count and compression ratios.

C.1. Benchmark Comparison

To evaluate our method against previous work on mesh-based integration [5], we listed only the orthographic projection for the DiLiGenT-MV dataset [6] in the main document. We complement this comparison with the results of the perspective projection in Tab. 1. Furthermore, we perform the same evaluation on the LUCES dataset [7], but only for the perspective projection. Since this dataset is dedicated to near-field photometric stereo, *i.e.* objects are very close to the camera, the orthographic projection is an unsuitable approximation. As in previous tests, we match the vertex count to the low, mid and high settings of previous work [5]. Results are listed in Tab. 2. Again, our method generates tighter approximations of the underlying surface.

C.1.1 Inspecting the Buddha Results in DiLiGenT-MV

We noticed that our decimation-based method at the 'high' accuracy setting outperforms the pixel-based baseline in the case of the BUDDHA figurine in the DiLiGenT-MV dataset. This is surprising as our method uses fewer vertices than the pixel-based method. To investigate the origin of this result, we examined the differences in each view. Figure 1

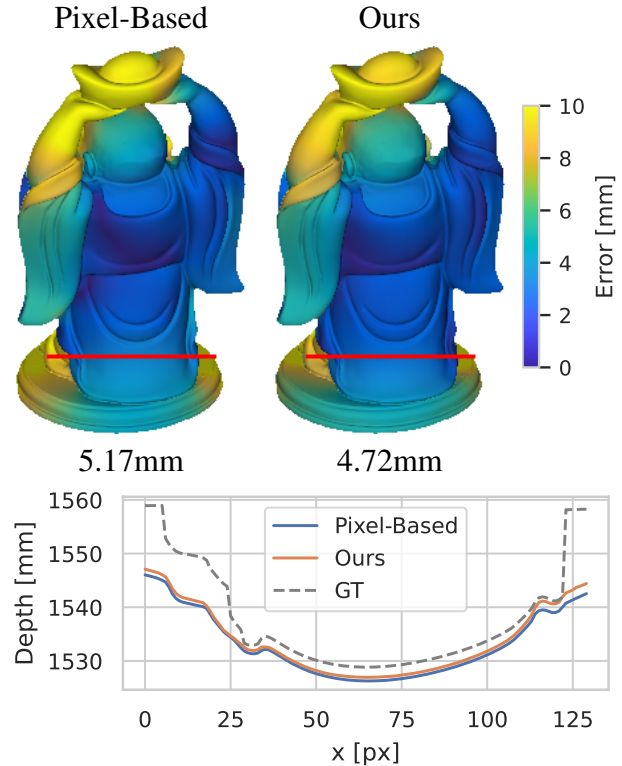


Figure 1. Top: Error map and RMSE for pixel-based [1] (*left*) and our (*right*) integration. Bottom: Slice of the aligned depth maps (along the red line indicated above).

illustrates the error map for the tenth view. While our decimation is generally lossy – except for perfectly flat regions – discontinuities and highly slanted (near-discontinuous) surfaces are problematic and error prone regions for normal integration. For the Buddha statue, such surfaces are present around the base and the lower part of the garment. It seems that our method performs slightly better in these situations. We believe this is due to differences between our integrator and the integrator in [1]. In [1] all normals are weighted equally while our method compensates for foreshortening assigning a higher weight to normals in slanted regions, as their real surface is bigger than it appears in screen space.

C.1.2 Inspecting the Buddha Results in LUCES

The BUDDHA figurine in the LUCES dataset shows an atypical behaviour: Lower mesh resolutions yield a better surface approximation. This inverted connection between resolution and reconstruction quality is also present for the previous isotropic normal integration [5], *cf.* Tab. 2. A visual inspection revealed that the normal integration places the Buddha’s face too far in the front compared to the figurine’s base, see Fig. 2. This is true for both pixel- and mesh-based integration. At lower mesh resolutions, there are more pix-

	[1]	Isotropic [5]			Ours		
Dataset	Ref	low	mid	high	low	mid	high
BALL	0.40	0.56	0.48	0.47	0.54	0.51	0.49
BELL	0.30	0.82	0.62	0.54	0.54	0.51	0.47
BOWL	0.08	0.35	0.22	0.15	0.15	0.14	0.12
BUDDHA	3.46	3.59	3.68	3.73	3.46	3.55	3.56
BUNNY	3.38	4.03	3.93	3.83	3.90	3.80	3.74
CUP	0.01	0.36	0.20	0.08	0.06	0.03	0.02
DIE	1.62	2.98	2.67	2.46	2.83	2.57	2.63
HIPPO	2.73	3.13	2.96	2.91	3.04	2.88	2.86
HOUSE	11.08	11.08	11.30	11.35	11.38	11.49	11.32
JAR	0.50	0.55	0.46	0.43	0.43	0.43	0.43
OWL	4.89	6.22	5.86	5.69	6.00	5.47	5.34
QUEEN	3.74	5.17	4.43	4.30	5.05	4.08	3.98
SQUIRREL	1.91	2.87	2.47	2.34	2.62	2.15	2.03
TOOL	0.91	1.04	0.96	0.93	0.91	0.90	0.90

Table 2. RMSE in mm for all objects of the LUCES dataset using the perspective projection. The vertex count is given by the respective low, mid and high settings of [5] and matched by our decimation pipeline.

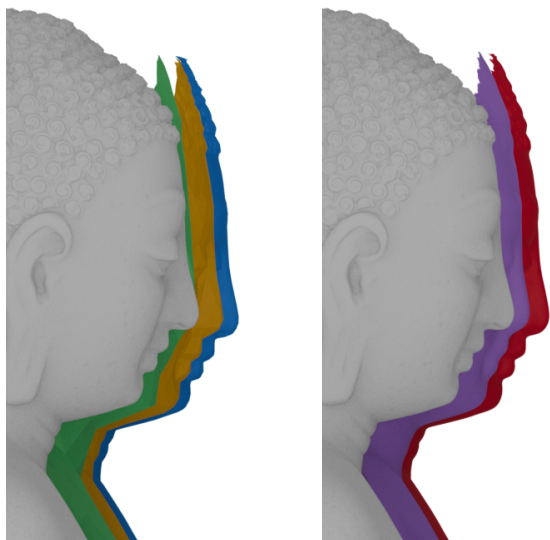


Figure 2. Side profiles of the BUDDHA figurine in LUCES [7]. Left: Our decimation with thresholds 2048 (green), 64 (yellow) and 2 (blue). Right: Pixel-based integration [4] with ground-truth normals (red) and smoothed normals (purple). All examples were aligned to the ground-truth surface (grey) at the base of the figurine.

els per triangle which implicitly smoothens the surface normals and flattens the integrated surface. This flattening coincidentally reduces the constant offset to the ground-truth surface. This hypothesis is supported by the fact that we can recreate this behaviour in pixel-based integration by applying a Gaussian kernel to the normal map, see Fig. 2.

C.2. Controllability

In the main paper, we studied the controllability using the LUCES [7] dataset. For completeness, we list all results in Tab. 3. For a more extensive study of the influence of the decimation threshold on the final mesh quality, we tested all normal maps of the DiLiGenT-MV dataset [6] for threshold values ranging from 0.25 to 512 and evaluated both root mean square error (RMSE) and mean absolute deviation (MADE) – both after the appropriate rigid-alignment to absolve the scale ambiguity – as well as mean angular error (MAE) and vertex count. The results are depicted in Fig. 3. As expected, there is virtually no difference between the results for orthographic and perspective projection. All objects are far away from the camera, *i.e.* the orthographic projection is a good approximation of the true perspective projection. The compression ratios reflect how our algorithm adapts to the complexity of the datasets: The BEAR and COW mostly consist of smooth featureless surfaces and achieve higher compression ratios than the more complex BUDDHA dataset, especially for lower thresholds.

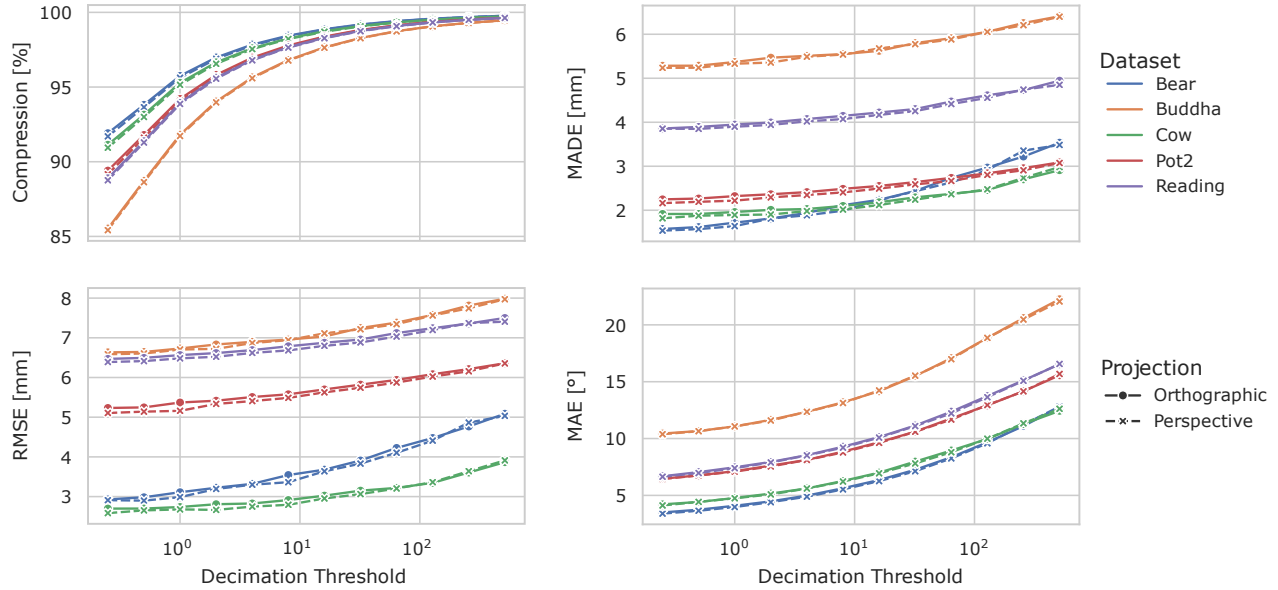


Figure 3. Influence of the decimation threshold on compression ratio, RMSE, MADE and MAE. All numbers are averages over the 20 views for each object. We investigate both orthographic and perspective projection. Please note the logarithmic x -axis.

Dataset	Ref	Threshold		
	[4]	2	2^6	2^{11}
BALL	0.40	0.48	0.54	0.60
BELL	0.30	0.33	0.49	0.72
BOWL	0.08	0.10	0.13	0.17
BUDDHA	3.46	3.58	3.51	3.32
BUNNY	3.38	3.67	3.83	4.15
CUP	0.01	0.01	0.04	0.14
DIE	1.62	1.65	2.11	3.09
HIPPO	2.73	2.85	2.91	3.09
HOUSE	11.08	11.30	11.56	11.39
JAR	0.50	0.45	0.45	0.46
OWL	4.89	5.41	5.63	5.94
QUEEN	3.74	4.02	4.41	4.98
SQUIRREL	1.91	2.07	2.37	3.00
TOOL	0.91	0.98	0.98	1.04

Table 3. RMSE in mm for all of the objects in LUCES [7] with increasing decimation threshold. The chosen thresholds should lead to a constant reduction rate of the RMSE.

C.3. Error Maps for the Benchmark Comparisons

LUCES Dataset (1 of 3)

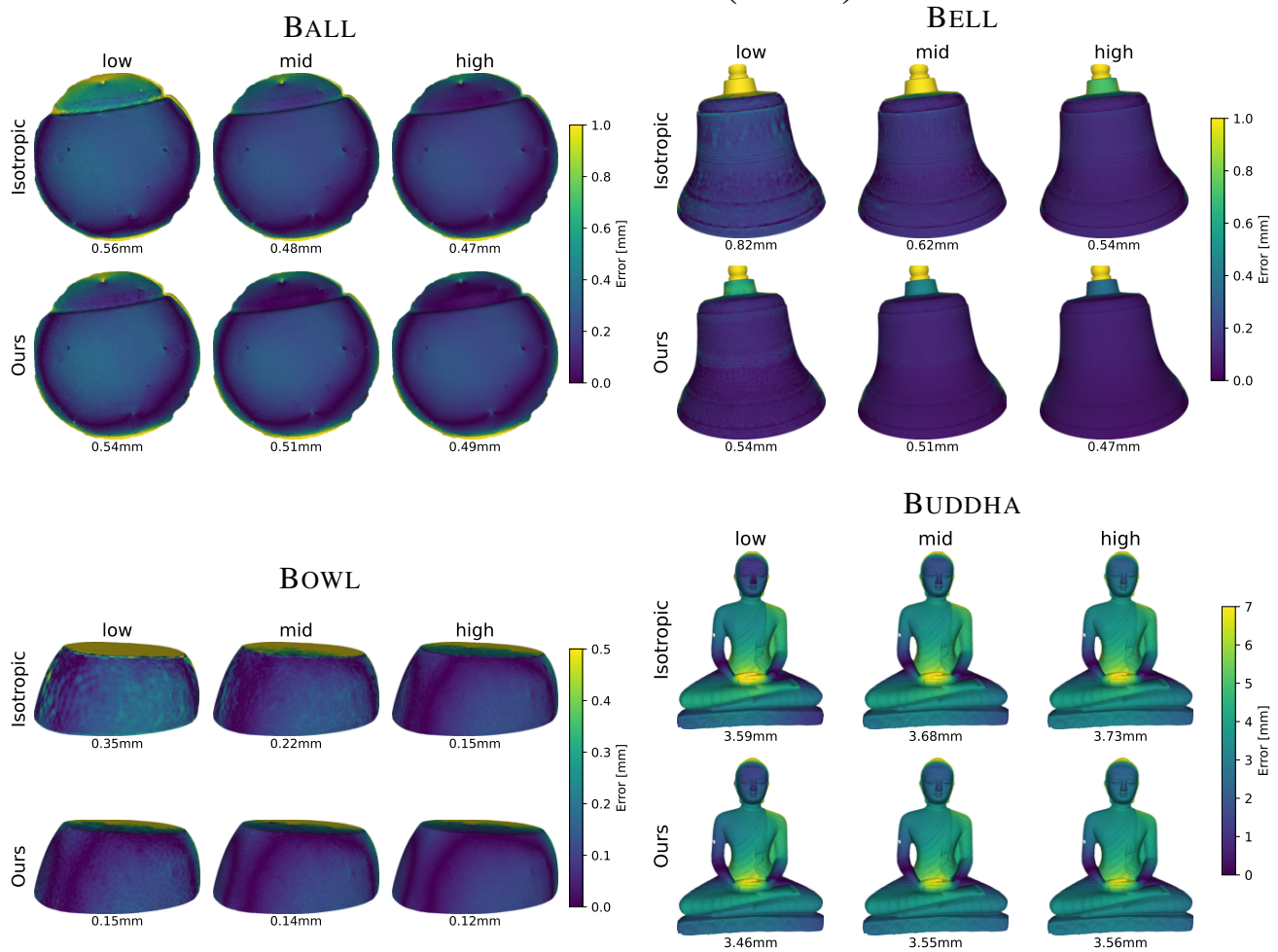


Figure 4. Error maps of the LUCES dataset after rigid alignment. We show results for all three quality settings in [5] and match the respective vertex number for our method. Pictured are the results of the perspective projection.

LUCES Dataset (2 of 3)

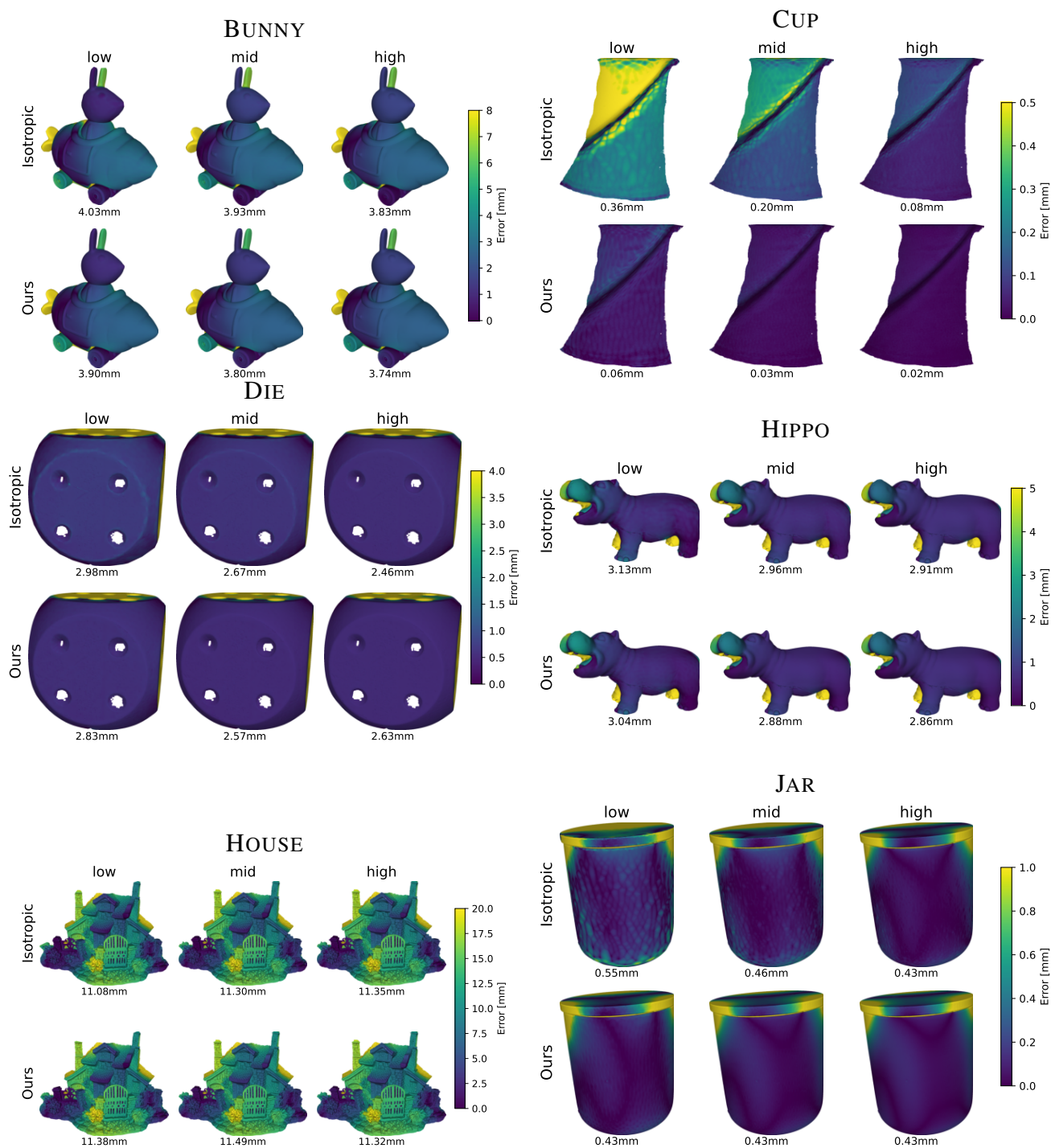


Figure 5. Error maps of the LUCES dataset after rigid alignment. We show results for all three quality settings in [5] and match the respective vertex number for our method. Pictured are the results of the perspective projection.

LUCES Dataset (3 of 3)

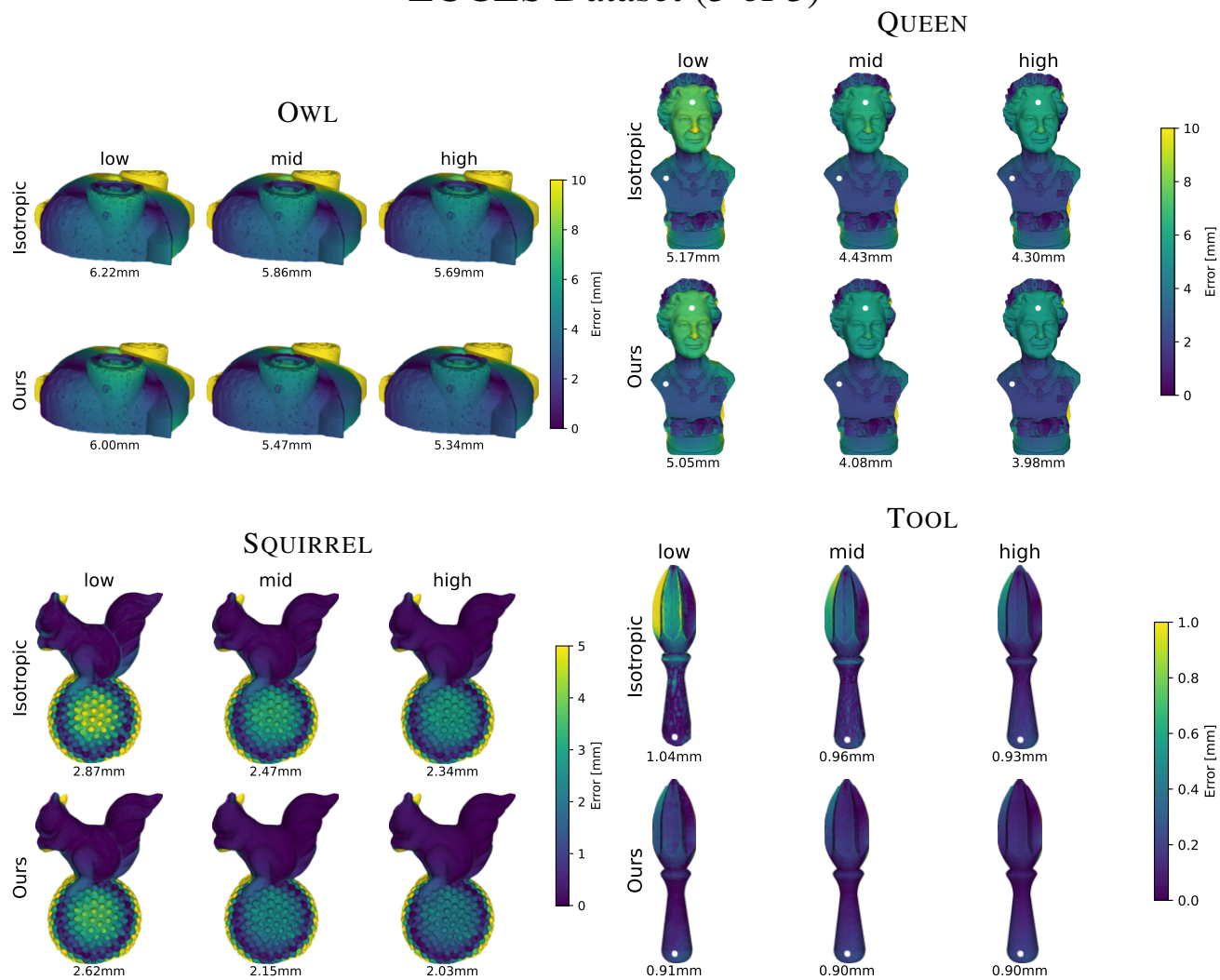


Figure 6. Error maps of the LUCES dataset after rigid alignment. We show results for all three quality settings in [5] and match the respective vertex number for our method. Pictured are the results of the perspective projection.

DiLiGenT-MV Dataset

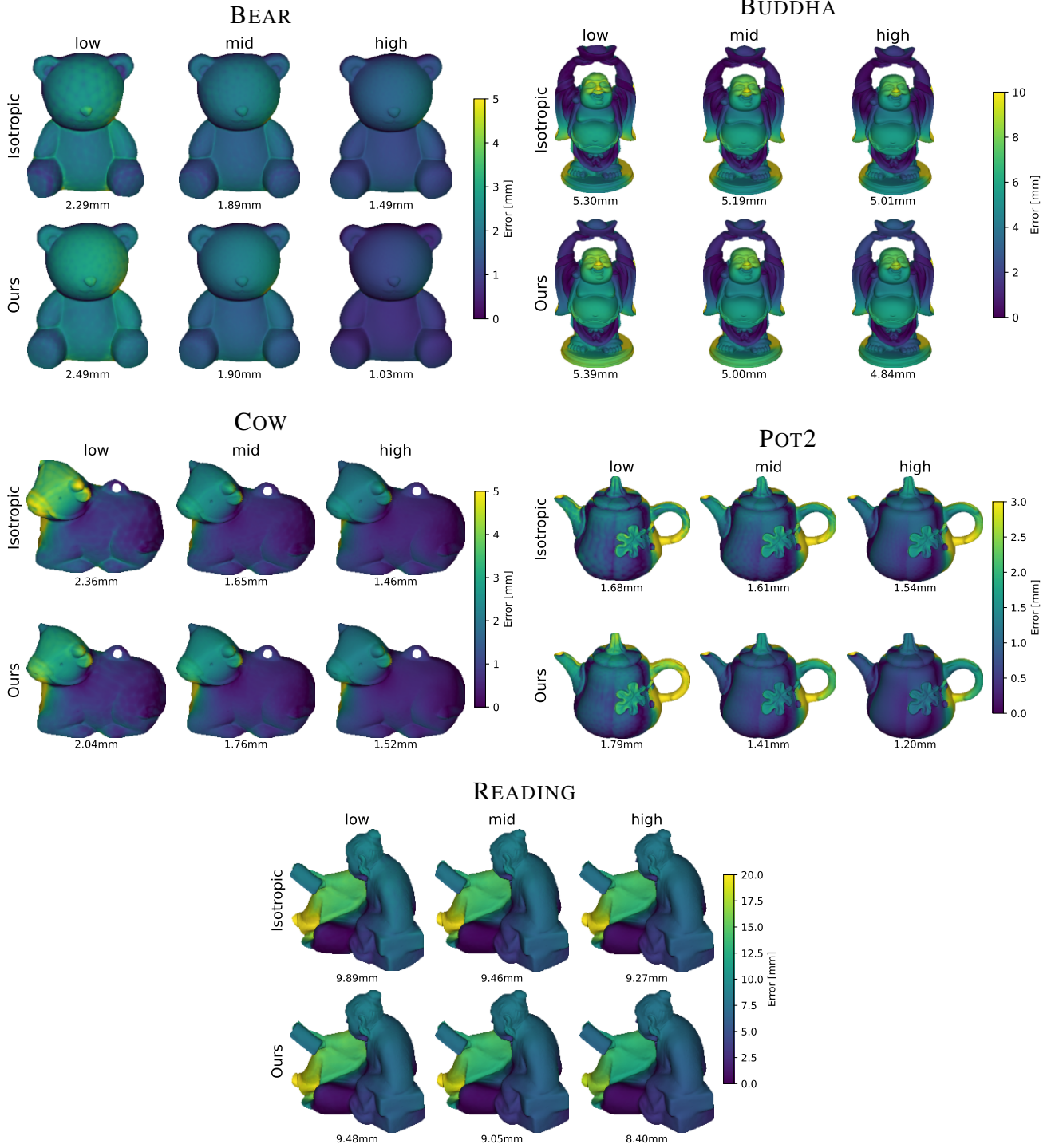


Figure 7. Error map for the first view in the DiLiGenT-MV dataset after rigid alignment. We show results for all three quality settings in [5] and match the respective vertex number for our method. Pictured are the results of the orthographic projection.

C.4. Additional Datasets

Finally, we show reconstructions from all the datasets we used. Except for LUCES [7], these datasets come without ground-truth geometry. The RMSEs for all objects in LUCES were reported in Tab. 3 which complements Tab. 2 from the submitted manuscript. For visual inspection, all objects can be found in Fig. 8 to Fig. 14. We also indicate the vertex count and compression ratio to put the results into perspective.

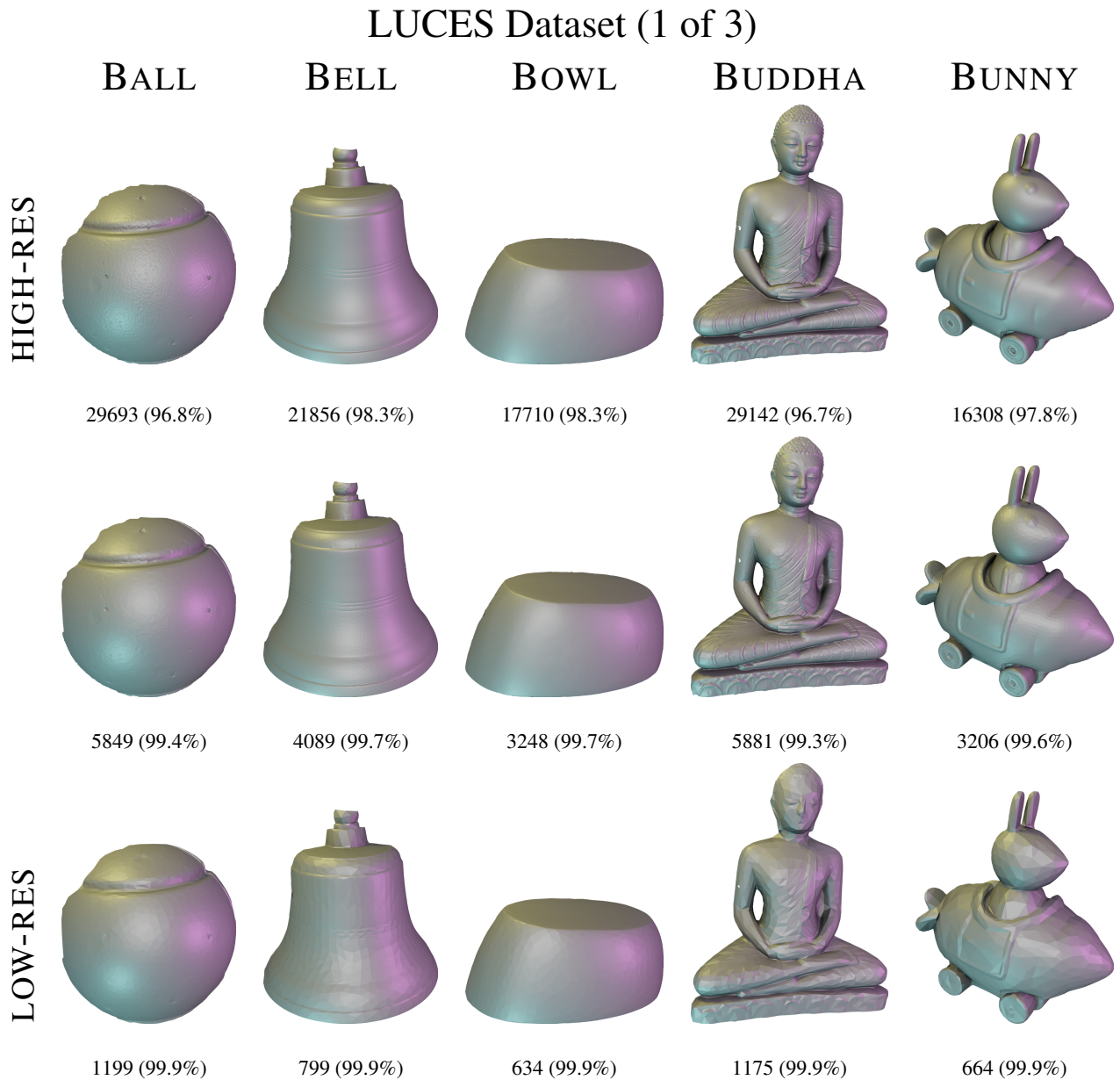


Figure 8. Reconstruction results for the LUCES dataset [7] for decimation thresholds of 2, 64 and 2048. These reconstructions correspond to the numbers reported in Tab. 2 of the manuscript. Any holes in the mesh surface are part of the provided foreground mask.

LUCES Dataset (2 of 3)

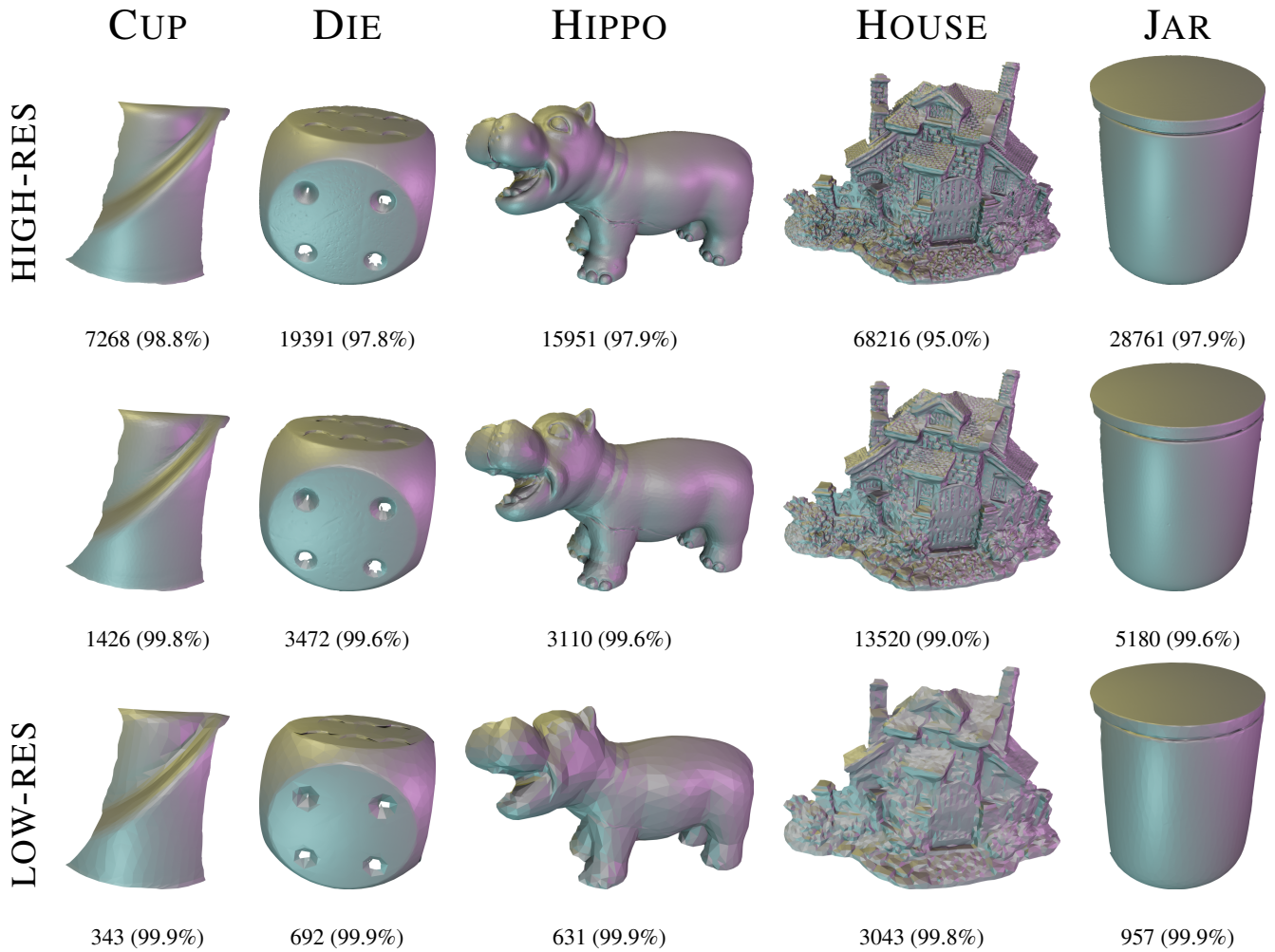


Figure 9. Reconstruction results for the LUCES dataset [7] for decimation thresholds of 2, 64 and 2048. These reconstructions correspond to the numbers reported in Tab. 2 of the main paper. Any holes in the mesh surface are part of the provided foreground mask.

LUCES Dataset (3 of 3)

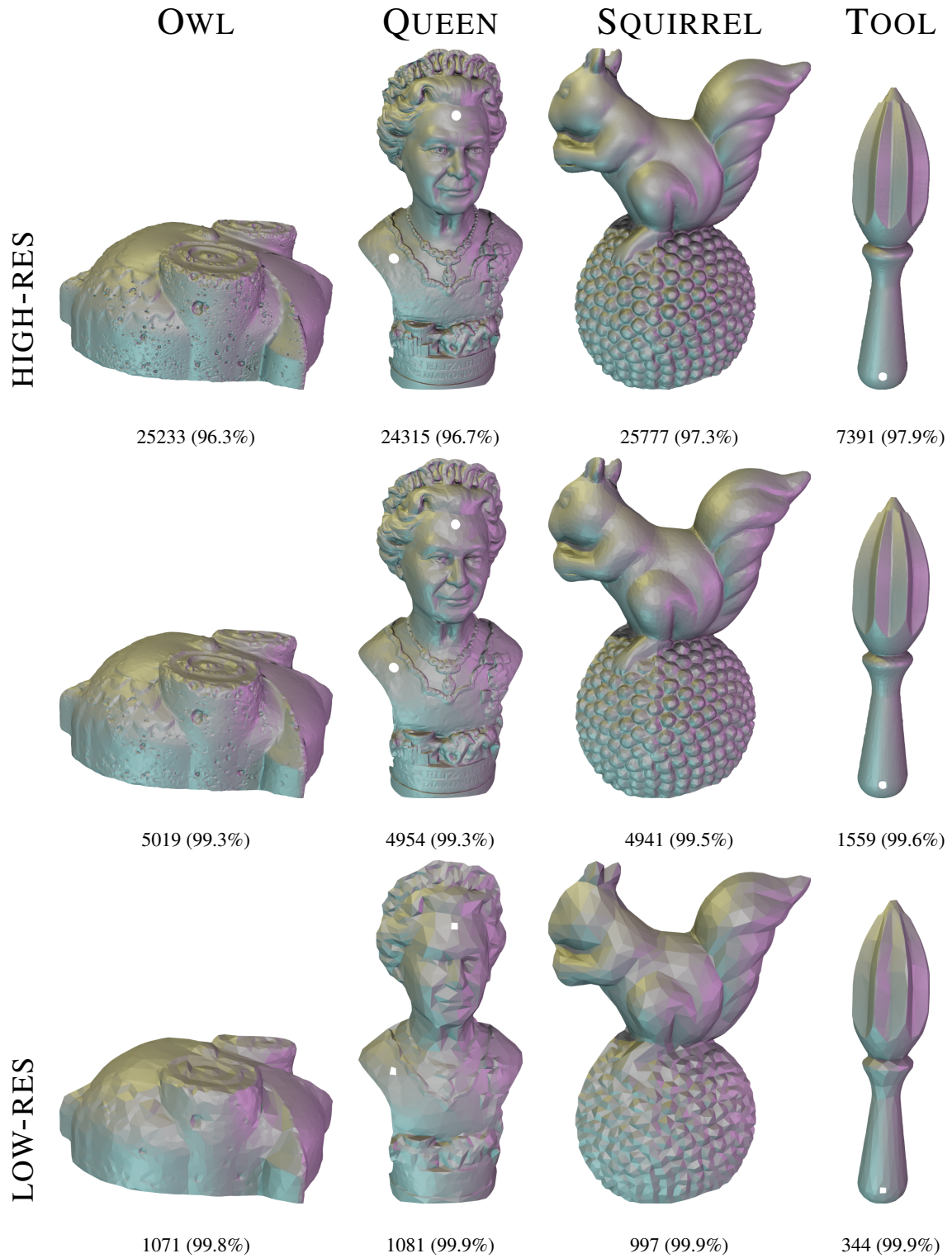


Figure 10. Reconstruction results for the LUCES dataset [7] for decimation thresholds of 2, 64 and 2048. These reconstructions correspond to the numbers reported in Tab. 2 of the main paper. Any holes in the mesh surface are part of the provided foreground mask.

RGBN Dataset (1 of 2)

High-Res

Low-Res

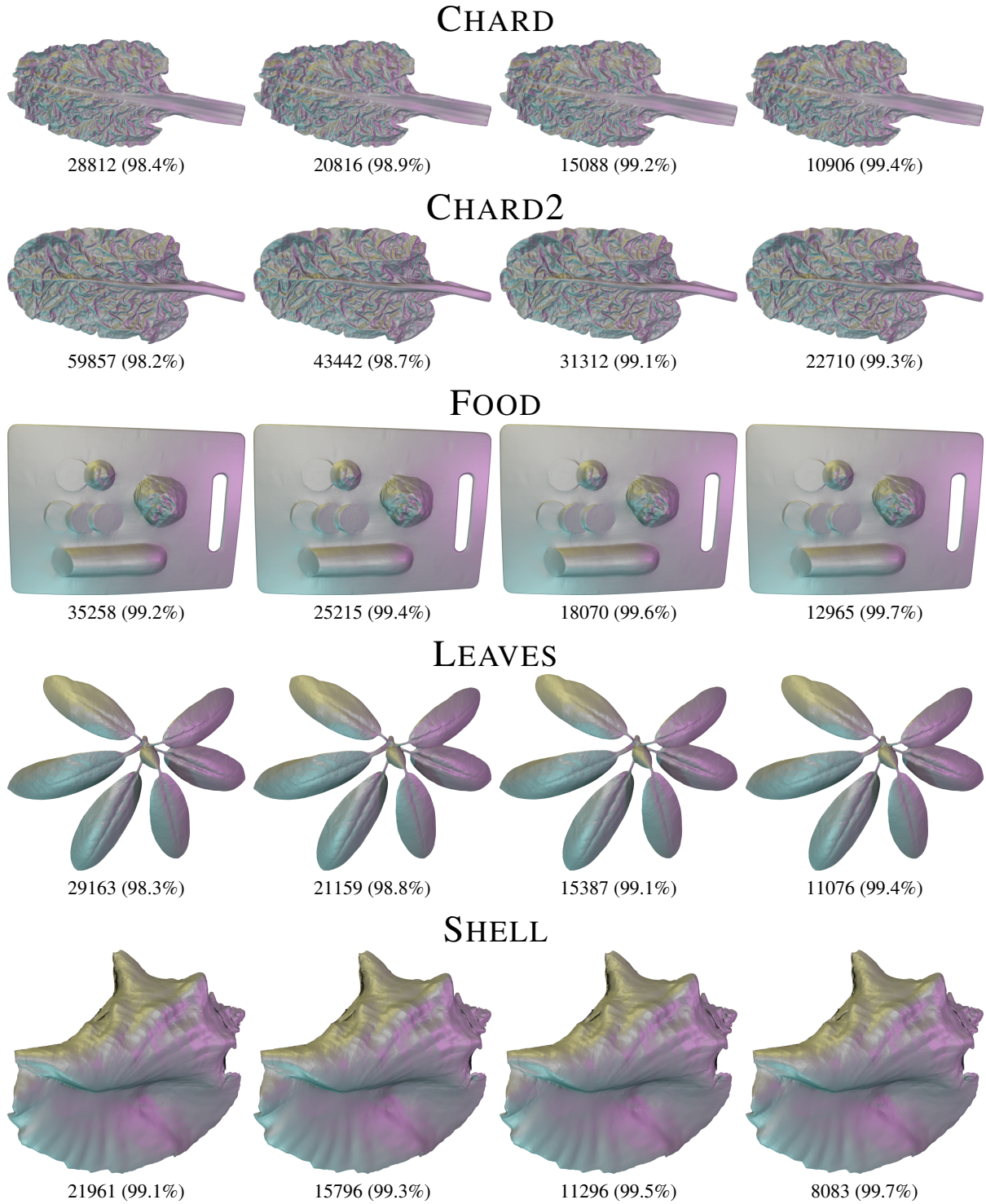


Figure 11. Reconstruction results for the RGBN dataset [9] for decimation thresholds of 8, 16, 32 and 64.

RGBN Dataset (2 of 2)

High-Res

Low-Res

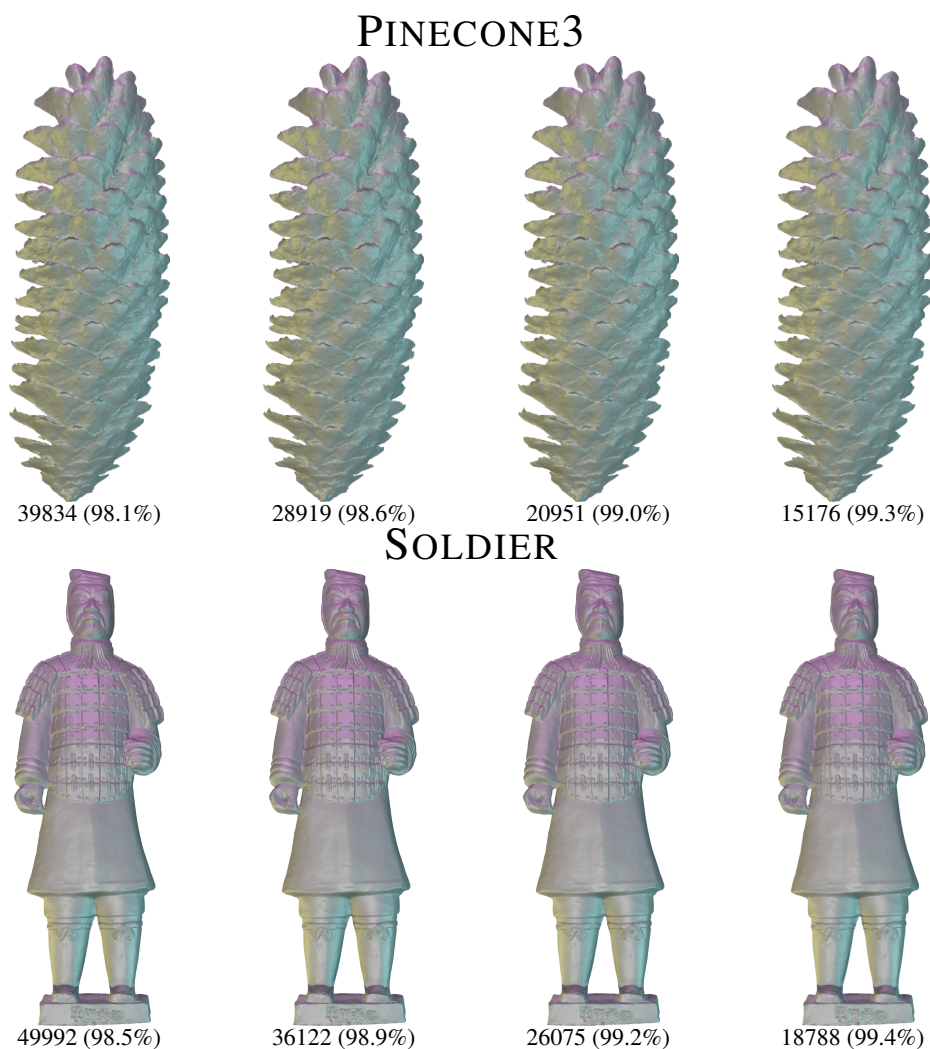


Figure 12. Reconstruction results for the RGBN dataset [9] for decimation thresholds of 8, 16, 32 and 64. Objects were rotated to the upright position.

PS Dataset (1 of 2)

High-Res

Low-Res

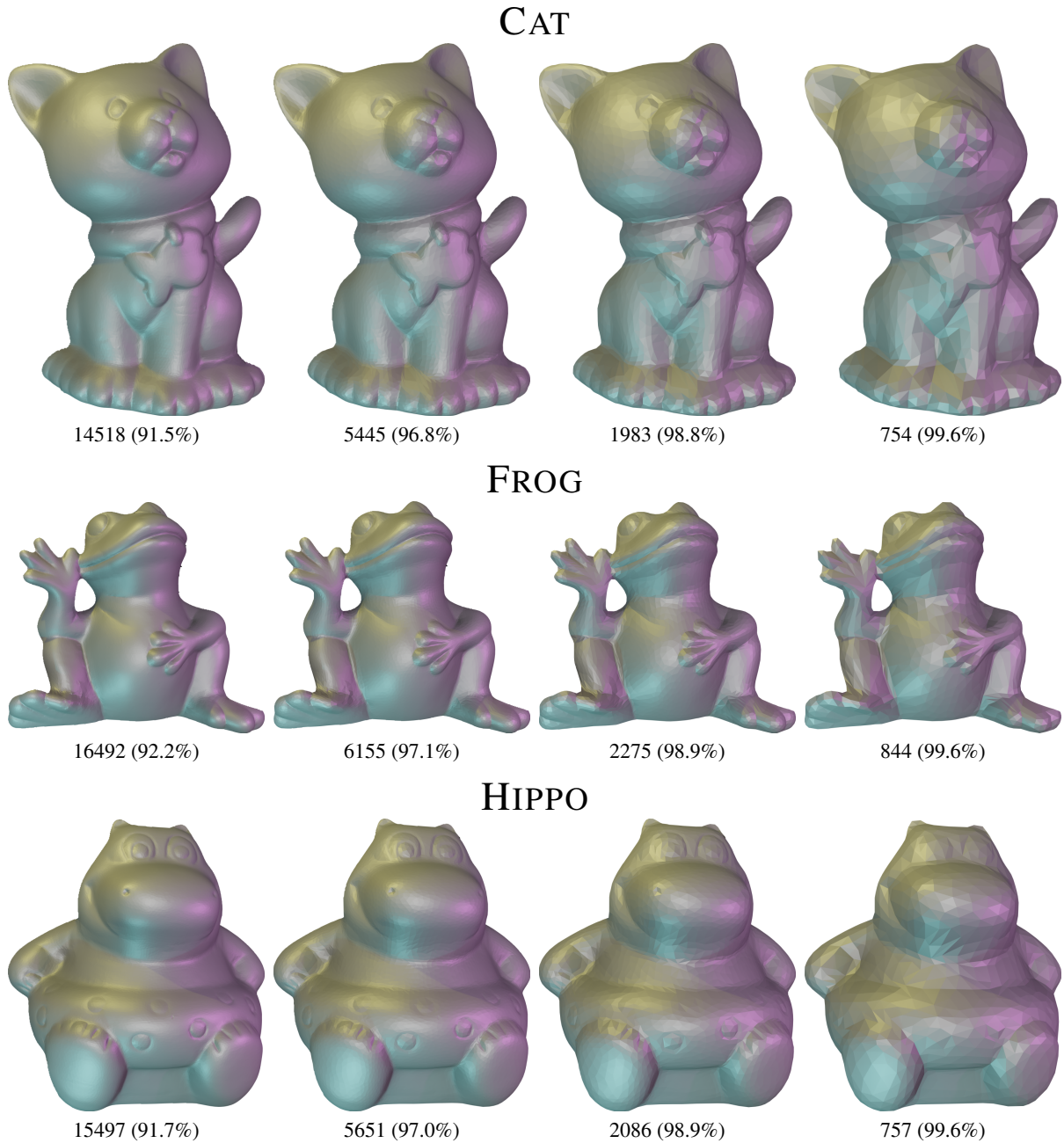


Figure 13. Reconstruction results for the PS dataset [3] with decimation thresholds of 0.125, 1, 8 and 64. The decimation threshold increases from left to right, *i.e.* mesh resolution decreases from left to right.

PS Dataset (2 of 2)

High-Res

Low-Res

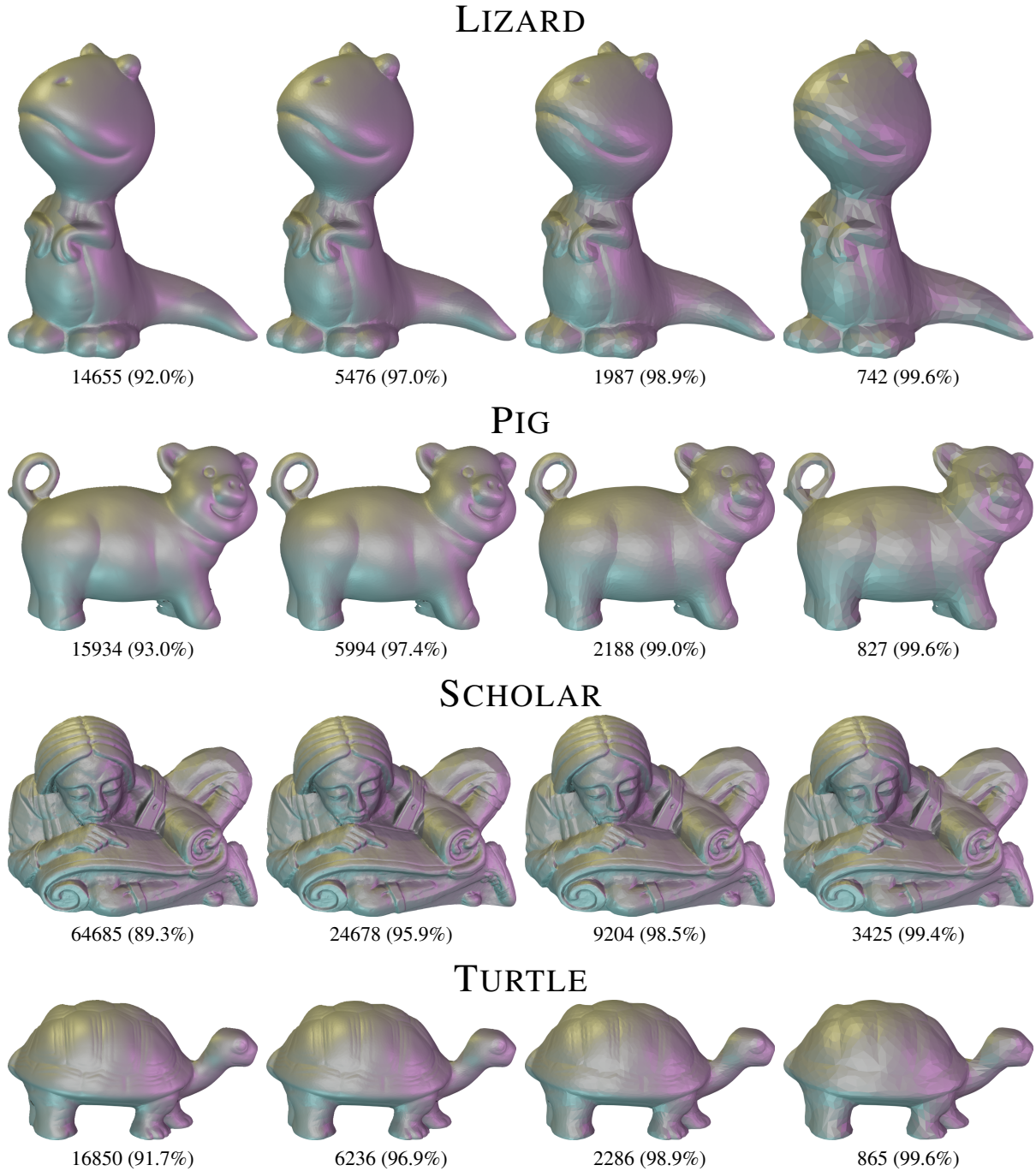


Figure 14. Reconstruction results for the PS dataset [3] with decimation thresholds of 0.125, 1, 8 and 64. The decimation threshold increases from left to right, *i.e.* mesh resolution decreases from left to right.

D. Overview of all Datasets

In this work, we used the following photometric stereo datasets:

- DiLiGenT-MV [6]: <https://sites.google.com/site/photometricstereodata/mv>
- LUCES [7]: <http://www.robortomecca.com/luces.html>
- RGBN [9]: <https://gfx.cs.princeton.edu/gfx/proj/rgbn/>
- PS Dataset [3]: <https://vision.seas.harvard.edu/qsfs/Data.html>

Furthermore, we generated synthetic datasets using the following 3D models:

- David Head [1d.inc]: <https://sketchfab.com/models/39a4d01bef37495cac8d8f0009728871/>
- Football Medal 2 [Cain]: <https://sketchfab.com/models/54d54534f11d4d23aecb945fd7eb1df4/>
- Female Head: <https://www.3dscanstore.com/3d-head-models/raw-expression-bundles/female-02-x36-expression-bundle>
- Male Head: <https://www.3dscanstore.com/3d-head-models/raw-expression-bundles/male-01-36x-expression-scan-bundle>

References

- [1] Xu Cao, Boxin Shi, Okura Fumio, and Yasuyuki Matsushita. Normal Integration via Inverse Plane Fitting With Minimum Point-to-Plane Distance. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2382–2391, 2021.
- [2] Long Chen. Mesh Smoothing Schemes Based on Optimal Delaunay Triangulations. In *Proc. of International Meshing Roundtable (IMR)*, pages 109–120. Citeseer, 2004.
- [3] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(4): 439–451, 1988.
- [4] Moritz Heep and Eduard Zell. ShadowPatch: Shadow Based Segmentation for Reliable Depth Discontinuities in Photometric Stereo. *Computer Graphics Forum*, 41(7):635–646, 2022.
- [5] Moritz Heep and Eduard Zell. An Adaptive Screen-Space Meshing Approach for Normal Integration. In *European Conference on Computer Vision, (ECCV)*, pages 445–461, Milan, Italy, 2025. Springer.
- [6] Min Li, Zhenglong Zhou, Zhe Wu, Boxin Shi, Changyu Diao, and Ping Tan. Multi-view photometric stereo: A robust solution and benchmark dataset for spatially varying isotropic materials. *IEEE Trans. on Image Processing*, 29: 4159–4173, 2020.
- [7] Roberto Mecca, Fotios Logothetis, Ignas Budvytis, and Roberto Cipolla. LUCES: A Dataset for Near-Field Point Light Source Photometric Stereo. In *32nd British Machine Vision Conference (BMVC)*, 2021.
- [8] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.
- [9] Corey Toler-Franklin, Adam Finkelstein, and Szymon Rusinkiewicz. Illustration of complex real-world objects using images with normals. In *Proc. of Int. Symposium on Non-photorealistic Animation and Rendering NPAR*, pages 111–119, San Diego California, 2007. ACM.
- [10] Rui Xu, Longdu Liu, Ningna Wang, Shuangmin Chen, Shiqing Xin, Xiaohu Guo, Zichun Zhong, Taku Komura, Wenping Wang, and Changhe Tu. CWF: Consolidating Weak Features in High-quality Mesh Simplification. *ACM Trans. on Graphics*, 43(4):80:1–80:14, 2024.