Computers & Graphics 111 (2023) 63–76

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Saliency detection for large-scale mesh decimation[☆]

Rafael Kuffner dos Anjos [a,b,*], Richard Andrew Roberts [b], Benjamin Allen [b], Joaquim Jorge [d], Ken Anjyo [c,b]

[a] University of Leeds, Great Britain, United Kingdom
[b] CMIC, Victoria University of Wellington, New Zealand
[c] IMAGICA Group, Japan
[d] INESC-ID, Portugal

ARTICLE INFO

Article history:
Received 5 October 2022
Received in revised form 5 January 2023
Accepted 10 January 2023
Available online 24 January 2023

Keywords:
Geometric processing
Mesh saliency
Mesh decimation
Parallel algorithms

ABSTRACT

Highly complex and dense models of 3D objects have recently become indispensable in digital industries. Mesh decimation then plays a crucial role in the production pipeline to efficiently get visually convincing yet compact expressions of complex meshes. However, the current pipeline typically does not allow artists control the decimation process, just a simplification rate. Thus a preferred approach in production settings splits the process into a first pass of saliency detection highlighting areas of greater detail, and allowing artists to iterate until satisfied before simplifying the model. We propose a novel, efficient multi-scale method to compute mesh saliency at coarse and finer scales, based on fast mesh entropy of local surface measurements. Unlike previous approaches, we ensure a robust and straightforward calculation of mesh saliency even for densely tessellated models with millions of polygons. Moreover, we introduce a new adaptive subsampling and interpolation algorithm for saliency estimation. Our implementation achieves speedups of up to three orders of magnitude over prior approaches. Experimental results showcase its resilience to problem scenarios that efficiently scales up to process multi-million vertex meshes. Our evaluation with artists in the entertainment industry also demonstrates its applicability to real use-case scenarios.

1. Introduction

The ever increasing requirements for highly detailed three-dimensional meshes in movies, animations and interactive media such as games has been met with progress in hardware and rendering techniques. However, it is still common practice to simplify models for not only final usage, but in earlier stages of production, as loading and manually editing models with millions of vertices in interactive software is still costly. While traditional fast mesh simplification algorithms might be used [1] for this, these only allow control of **how much** decimation will be applied, not **where** it will be applied. Thus, artists end up manually removing superfluous faces (or even recreating the mesh by hand) to preserve details they believe are important, which is impractical for complex geometry.

Saliency-based decimation [2,3] retains details on visually significant regions while yielding coarser geometry in other areas. Notably, importance estimation and decimation become two distinct passes, supporting parameter tuning and artist feedback (favored in industry [4]) before decimating. However, state-of-the-art saliency estimation techniques may require several days to process million-polygon meshes, a standard in the entertainment industry. Factoring in multiple runs to adjust parameters, or failure cases on problematic meshes, manually painting saliency values on a model has surprisingly become the faster approach. While previous work [5] proposed pre-decimating meshes via standard methods [6] to yield quicker results, simplification irreversibly erases details, defeating the purpose of saliency-aware methods.

This paper addresses existing challenges to making saliency-aware decimation viable in production environments. The following two requirements for saliency detection must then be at least fulfilled:

- R1 **Robustness:** Saliency calculation should be stable under varying mesh topologies, given that sculpted or scanned models are a standard in the industry.
- R2 **Fast computation:** Each turnaround of saliency calculation should be performed at close to interactive rate for million-polygon meshes.

Unfortunately previous saliency detection methods have not considered these demands in production settings or could not meet them at a time.

☆ This article was recommended for publication by Marco Attene.
* Corresponding author at: University of Leeds, Great Britain, United Kingdom.
E-mail address: r.kuffnerdosanjos@leeds.ac.uk (R. Kuffner dos Anjos).

https://doi.org/10.1016/j.cag.2023.01.012

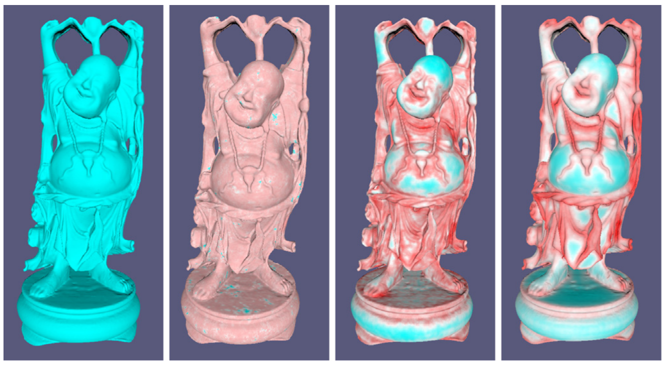


Fig. 1. Contribution of tropical angle and the change to the binning operation using β. From left to right: Buddha model using Local Curvature Entropy (LCE) [2]. Next, gamma correction (0.03) still fails to show distinctive features. Then, introducing $\beta = (0.1)$ allows distinctive features to be detected but still contains much noise in the base where the model is flat. Finally, by applying the tropical angle of curvature we obtain a smooth map using estimated β. Color mapping for all images in this paper is [0.0 − 1.0], cyan to red.

We therefore propose a new practical technique for saliency detection to meet these demands and its usage in production environments. This technique introduces a novel saliency estimation, based on the entropy of the *tropical angle of curvature* that estimates the maximum immediate difference of normals at each vertex on a surface (see Fig. 1). As illustrated later, the tropical angle of curvature ensures stable calculation for the saliency estimation. Our new *adaptive subsampling* approach coupled with efficient data structures and a parallel traversal algorithm provides speedups of up to three orders of magnitude for the saliency estimation over current techniques.

In addition, we consider the following requirement, which may be a general demand for a practical tool:

R3 **Artist-friendly parameter control:** The parameters provided should be easily tuned by artists to get their desired results.

Regarding this, we demonstrate that our method is configurable by one global parameter or via five high-level parameters for experienced artists, allowing flexibility to adapt to complex cases such as creases and 3D scanned models as well as to noise mitigation of the scanned data. The applicability of our technique to production settings is also confirmed by an evaluation performed with artists.

2. Related work

The landmark work from Koch and Ulmann [7] postulates that attention is naturally driven towards locally salient elements from their surroundings at different scales. Thus, mesh saliency [8] has traditionally been calculated with multi-scale shape analysis on 3D models. The survey from Liu et al. [3] classifies saliency detection methods as single-scale, global-scale and multiple-scale. Single-scale techniques are purely local methods that analyze each point independently using a small neighborhood. They have been successfully used for different applications, e.g curvature maxima to find contours [9] or 3D Harris descriptors as local saliency measurements for shape matching and alignment [10]. Similarly they have been used to calculate point cloud saliency [11], where point clusters are compared in terms of uniqueness to neighboring clusters. Their main limitation is focusing on local differences (differently than the visual attention system [7]), producing mostly binary saliency maps with the sharpest features.

Lee et al.'s [8] is a multi-scale method, combining Gaussian-averaged curvatures at multiple scales. Miao et al. [12] propose estimating a relief height from a mesh, trying to fit a surface to each vertex at different scales, where stronger deviations mean higher saliency. While they showed positive outcomes on terrain outlines, their results proved noisier on common models. Moreover, the plane fitting technique becomes considerably expensive at larger scales. Song et al. [5] propose a similar but even costlier approach. This is the follow-up to a well-known multiple-scale work that handles global features [13]. They calculate stochastic laplacians of the mesh at different scales. Then combine both average and max laplacian values, pooling at varying levels of detail. Finally, they introduce global features later in the process so that local features still appear in the final map, an improvement over their previous work. The global component of these algorithms is also computationally expensive. Because of this, both techniques rely on simplifying the model before calculating its salient features, which makes them not suited to estimating saliency for mesh simplification. See Section 5 for further details.

Sipiran and Bustos [14] developed a global method, using Harris 3D features to classify mesh elements, and estimate saliency by their grouping. Wu et al. [15] take a similar approach, but using the contrast in Zernike moments [16,17] within vertices and their neighbors. However, as all global contrast methods, these techniques are computationally expensive. Page et al. [18] propose an interesting shape analysis method, which defines saliency via Shannon's entropy of the Gaussian curvature of a surface. This inspires Limper et al. [2]'s approach, which measures local entropy of the mean curvature to indicate how much information is contained in a neighborhood of a vertex. Their approach is applied at multiple scales, and combined into a unified map that successfully contains quasi-global and locally salient elements. However, the curvature calculation in these approaches is affected by tessellation density, so that we need a more stable approach in dealing with highly complex and dense models. A recent review [19] proposes a saliency comparison technique, highlighting [2,8,13] as good distinctive representatives of current approaches. They conclude that a combination of methods should produce better results than isolated measurements.

Recent work [20–23] uses eye-tracking to annotate salient points on meshes based on where people look at. In this body of work we can see texture and lighting having a significant impact on the result of the detected saliency; which is consistent with the model described by Koch and Ulmann [7] as these details at a varying scale will draw attention; and that contextual information also draws attention (e.g. faces in humans, handles in mugs). This category of work tend to claim low agreement to surface based methods, which in their view may disprove those. However, the proposed experiments use but a limited number of views, over a limited time, and lighting conditions, highlighting just a couple of "most salient" points in a model. While these are useful for understanding human perception, their application to decimation is limited.

Song et al. [24,25] apply deep learning to detect mesh saliency, by extrapolating view dependent 2D salient points into a 3D mesh, which shows a higher degree of agreement with both previous surface based methods, and human-picked interest points (Schelling points) [26]. Their focus on saliency for view selection, also requires pre-decimation of meshes to achieve competitive run-times. Alternatively, Nousias et al. [27] use neural networks trained with spectral saliency [13] to allow estimation on larger meshes. While faster than calculating spectral saliency directly, it still shares the same limitations as the original method, and still has high execution times for simple models.

3. Saliency detection

In developing a practical tool for saliency detection, we first note that multi-scale surface-based methods approximate human-picked interest points [24], and closely mimic the human visual attention system [7]. Then, considering the requirements R1–R3, we found that entropy-based methods [2,18] were more scalable, amenable to parallelization, and well suited to our proposed subsampling scheme. This section focuses on each component of our saliency formulation and shows how they are combined. The whole process of our proposed method, together with pseudocode, will be presented in Section 4.

3.1. Local measurement: tropical angle of curvature ($\theta(v)$)

In order to measure "curviness" of a mesh surface, we first consider the angular difference between the two tangent planes at the closest vertices (sample points). For example, let T_a and T_b be the adjacent tangent planes at vertex a and b respectively. The angular difference is estimated by $\cos^{-1}(n_a \cdot n_b)$, where n_a, n_b are the normal vectors at a and b respectively. In our formulation, instead of $\cos^{-1}(n_a \cdot n_b)$, we consider the"normalized" angular difference, which is defined as $\frac{1-(n_a \cdot n_b)}{2}$. It takes values in [0, 1]. Then, for each vertex v in our input model we calculate $\theta(v)$ using Eq. (1) where $adj(v)$ are all vertices connected to v by a mesh edge:

$$\theta(v) = \max_{a,b \in adj(v)} \left\{ \frac{1-(n_a \cdot n_b)}{2} \right\}. \tag{1}$$

This means that we look for the largest (normalized) angular difference in the 1-ring neighborhood of vertex v, as an alternative of curvature at v. $\theta(v)$ plays a crucial role in our method and is referred to as *tropical angle of curvature* in this paper.

This novel geometric concept can be treated as a kind of curvature, providing new features as described in what follows. We find $\theta(v)$ to be a more stable measurement of local changes than the mean [2] or Gaussian curvature [18] as seen in Fig. 2.

For example, see Fig. 2 where the mean curvature and $\theta(v)$ are plotted. The model depicted has three different areas with varying tessellation levels. The absolute mean curvature suffers heavily from outliers caused by particular mesh topologies making an entropy analysis process volatile. When comparing to mean curvature with a manually set outlier boundary we show that the tropical angle of curvature values span the same range regardless of vertex area, and better reflect local differences. This is more suited to the discretization step in the entropy calculation, as it also allows us to control to more easily optimize the distribution of values to different bins (see Section 3.2) An example of usage of both measurements in saliency calculation can be seen in Fig. 2, where saliency is wrongly attributed to areas without salient details, just due to higher tessellation and small surface fluctuations.

The calculated $\theta(v)$ is used at two instances in our method: to calculate surface entropy over multiple areas (Section 3.2), and to identify local contours that are visually salient (Section 3.5).

A more detailed description and rationale about the tropical angle of curvature can be found in Appendix A.1 .

3.2. Entropy of tropical angle of curvature ($H(v, r)$)

As the main component of our saliency calculation, we consider the entropy of the tropical angle of curvature, and adapt the binning operation to improve the distribution of values.

The curvature calculation by Taubin [28] may suffer from tessellation variations. In this case, when discretizing the curvature

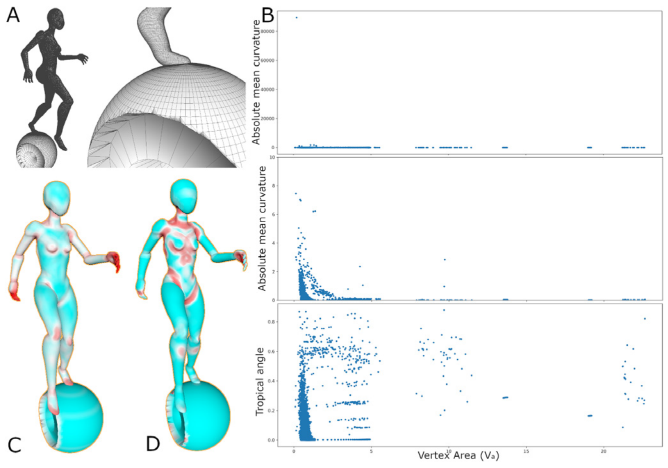


Fig. 2. (A) Model with varying tessellation in different areas. (B) Different measurements of curvature for a given model with three areas with different levels of tessellation. Tropical angle is the only one with no visible correlation to the tessellation level. (C) Saliency with tropical angle and (D) saliency with mean curvature.

```
1  Function H(v,r):
2      $\kappa_r\,[] = x : GD(x,v) < r$
3      $\sigma\,[n] = [0\ldots]$
4      for $x \in \kappa_r$ do
5          $i = \lfloor n * (\theta(x)^\beta - \theta^\beta_{min})/(\theta^\beta_{max} - \theta^\beta_{min}) \rfloor$
6          $\sigma\,[i] = \sigma\,[i] + A_x$
7      for $0 \le j < n$ do
8          $H = H - (\sigma\,[j]\,/A) * \log_2(\sigma\,[j]\,/A)$
9      return $H$
10
```

Algorithm 1: Algorithm for the entropy calculation for a given vertex v, with parameter radius r, a calculated distribution adjustment β, n = number of bins, A = total mesh area, A_x = vertex area and $GD(x, v)$ the geodesic distance between x and v.

measurement, we would increase the likelihood of distinct curvatures in highly-tessellated areas, while lesser tessellated areas will fall inside the same bin (i.e. the discretized range of possible curvature values), and thus be considered to be low saliency. We thus employ another definition of local entropy using the tropical angle, and adapt the binning operation.

Our local entropy at each vertex v of the mesh model considers the tropical angle of curvature (Eq. (1)) to the power of β at v, i.e, $\theta^\beta(v)$, where β is a parameter taking positive values.

Algorithm 1 describes the process to calculate this local entropy for θ^β, which is denoted by $H(v, r)$. We calculate it in the geodesic neighborhood of v with radius r: $U(v, r)$, which consists of all the points x whose geodesic distance $GD(x, v)$ from v is smaller than r. To evaluate the entropy $H(v, r)$, we now consider θ^β as a stochastic variable in $U(v, r)$. The probability density function of θ^β is then approximated with its "area-weighted" histogram. Given θ_{min}, θ_{max}, the minimum and maximum calculated values of the tropical angle of curvature of a given mesh, n a set number of discrete intervals in the range of possible curvatures (we call each one a "bin" for simplicity) in an array $\sigma\,[n]$, we accumulate the value A_v for each discrete range of curvature in our mesh (Alg 1, lines 5–6). A_{v_i} being the influence area of a vertex $v_i \in \kappa_r$, which is calculated using mixed Voronoi cells.

Parameter β will either spread out differences to different bins ($\beta < 1$), or concentrate them ($\beta > 1$). We can automatically determine a reasonable value for β from the observation that the entropy of the global curvature distribution approximates the mean entropy of all neighborhood curvature distributions. We solve for a value of β such that the global entropy is equal to some desired max histogram. We aim for mean saliency equal to the saliency midpoint (white in our figures), but do not know the saliency range beforehand as it depends on β. We evaluated the deviation of the mean saliency (R_μ) from the midpoint (R_{mid}) for various models from public repositories, made by various artists, and having medium to high polygon count with a range of saliency values $R_{\min}, R_{\max}$ by calculating $\frac{(R_\mu - R_{\text{mid}})}{(R_{\max} - R_{\min})}$. We found that, on average, a target entropy factor of 0.453 yields 0 deviation with variance <0.1, except for small models (<5000 vertices), where it becomes easier to manually adjust the parameter as the algorithm runs interactively.

The local entropy $H(v, r)$ is finally calculated by summing the area weighted contribution of each bin, where a larger variety of $\theta(v)$ would yield a higher entropy (Alg. 1, line 8).

Fig. 1 shows an in-depth analysis of the contribution of these first two changes to the saliency formulation.

3.3. Mitigating small scale noise ($\eta(v, r)$)

Outputs of 3D scanning software or photogrammetry data often include small-scale noise that should not be considered visually salient. Alternatively, in modern rendering pipelines, one may choose to ignore small geometry details that can be emulated by bump mapping.

Given a level of "noise" ϵ, our goal is to remove artifacts from surface fluctuations smaller than this value. The effect is harsher on low entropy surfaces such as flat plane-like structures. By approximating small neighborhoods to a plane (which is true in such surfaces and locally valid for more complex structures), we can find which points in that area have a variation within the noise level by taking their distance to the estimated plane. To this end, we first consider $\mathbf{N} = \{\vec{\text{avg}} \cdot w\}_{w \in U(v,r)}$, where $\vec{\text{avg}}$ is the average normal vector calculated over the neighborhood $U(v, r)$. Given $\Delta\mathbf{N} = \max \mathbf{N} - \min \mathbf{N}$, η is defined by Eq. (2):

$$\eta(v, r) = \begin{cases} 3(\frac{\Delta \mathbf{N}}{\epsilon})^2 - 2(\frac{\Delta \mathbf{N}}{\epsilon})^3 & \text{, if } \Delta\mathbf{N} < \epsilon \\ 1 & \text{otherwise.} \end{cases} \quad (2)$$

Our function $\eta(v, r)$ is defined using a common smooth-step function ($0, 3x^2 - 2x^3, 1$, where x is the interpolated variable) to avoid sharp discontinuities in the range where change is close to ϵ. Thus, ϵ will either scale down the contribution of a vertex v to the saliency calculation at that level, if its position deviates just slightly from a plane, or keep it unchanged if not the case.

3.4. Multi-scale calculation $R(v)$

We run each step of saliency estimation using $H(v, r)$ for a given geodesic radius r and a number of steps ℓ. This means that we estimate $H(v, \frac{r}{2^k})(0 \leq k \leq \ell - 1)$. The radius r of the geodesic search is a percentage value of the total area of the mesh, being decreased to $\frac{r}{2}$ in each iteration from 0 to ℓ. We combine the detected saliency at each level with the evaluation of $\eta(v, r)$ which can diminish the effect from noise. The multi-scale computation can be written as:

$$R(v) = \frac{1}{\ell} \sum_{k=0}^{\ell-1} \eta(v, \frac{r}{2^k}) H(v, \frac{r}{2^k}) \quad (3)$$

3.5. Integrating crease detection: final formulation $S(v)$

Finally, we introduce a weighted amount of "crease detection" which highlights immediately local edges which can be hard to parametrize for with the entropy formulation. This is sometimes desirable in a decimation process in cases where handcrafted local patterns must be preserved (e.g. pleated pattern on a skirt).

We normalize $R(v)$ based on its range $U(v, r)$ over all vertices. The normalization of $R(v)$ is denoted by $\widehat{R(v)}$:

$$\widehat{R(v)} = \frac{R(v) - R_{\min}}{R_{\max} - R_{\min}}, \quad (4)$$

where $R_{\min} = \min_{w \in U(v,r)}\{R(w)\}$ and $R_{\max} = \max_{w \in U(v,r)}\{R(w)\}$.

We thus define the complete saliency formula $S(v)$:

$$S(v) = \min\left(1,\ \alpha\theta(v) + \widehat{R(v)}\right). \quad (5)$$

For each vertex v in the mesh, Eq. (5) computes its saliency $S(v)$. $R(v)$ is computed in a first step, then it is normalized before adding the α weighted contribution of $\theta(v)$, the tropical angle of curvature at v, as curvature measurements have been widely used as a component of contour detection [29]. This limits its contribution to the final saliency measurement $S(v)$, not substantially increasing the relevant saliency of regions with creases when compared to the rest of the mesh. It can be used at no extra computation cost, since it is already used to calculate $H(v, r)$.

Our approach is thus controlled by five parameters: $\ell, r, \beta, \alpha, \epsilon$. We reduce these to one "globality" parameter Γ, that controls ℓ, r, α as a hyper-parameter. ϵ is estimated by the average noise present in the mesh, and the optimal value for β can be estimated as described in Section 3.2. A detailed description of this mapping and the effects of varying these parameters can be found in the Appendix and video accompanying this manuscript.

4. Subsampling scheme, and method implementation

```
Function Saliency($\ell$, $r_0$):
    for $k \in \{0, \ldots, \ell - 1\}$ do
        $r = r_0/2^k$
        $r_e = \sqrt{2} \cdot r/\sqrt{n_s}$
        candidates = vertices
        samples = {}
        while candidates $\neq$ {} do
            $v_i$ = PopRandom(candidates)
            samples = samples $\cup \{v_i\}$
            $\widetilde{H}_i = \eta(v_i, r) \cdot H(v_i, r)$
            for $v_j \in$ vertices | GD$(v_i, v_j) < r_e$ do
                candidates = candidates \ $\{v_j\}$
        for $v_i \in$ vertices \ samples do
            $H_\Sigma = 0$
            $w_\Sigma = 0$
            for $v_j \in$ samples | GD$(v_i, v_j) \leq 2r_e$ do
                $w = \text{MAX}\left(r_e/\text{GD}(v_i, v_j) - 1/2\ , 0\right)$
                $\widetilde{H}_\Sigma = \widetilde{H}_\Sigma + \widetilde{H}_j \cdot w$
                $w_\Sigma = w_\Sigma + w$
            $\widetilde{H}_i = \widetilde{H}_\Sigma / w_\Sigma$
```

Algorithm 2: Pseudo-code of our subsample and interpolate algorithm. Per level we do a first loop to select samples and calculate saliency over a ground truth neighborhood, and a second to perform interpolation.

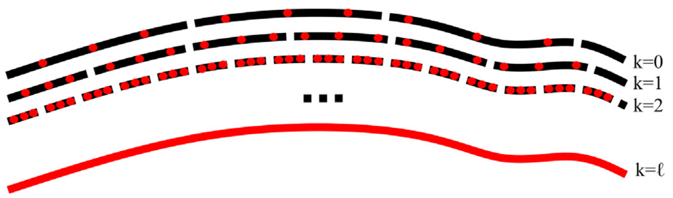


Fig. 3. Description of the sampling process with $n_s = 3$. When sampling smaller neighborhoods, details are detected as more vertices out of the original mesh are chosen as samples.

Fig. 4 presents an overview of our method, where we utilize the mathematical notations introduced in Section 3 along with the corresponding images for all steps. This section will introduce the subsampling approach, and how it implements the theory presented in the previous section.

Subsampling is a popular approach to speeding-up computation over large datasets [30], and it will typically be associated with a quality vs. speed trade-off. However, our method can reduce the introduced error to negligible levels by tailoring the sampling algorithm to our problem domain.

Adaptive: $S(v)$ is inherently a multi-scale measurement. It combines information about larger regions of high saliency with individual features that have been detected at smaller scales. When r is large, the computation is more expensive, but the nature of the result is primarily averaged and perceived as smooth over the model, which can be achieved with fewer samples. The computation is quicker with a small r, but we can easily miss local detail by not choosing enough samples, requiring a higher sample count. Thus, we control our sampling rate via the parameter n_s, which defines the number of *samples-per-neighborhood*. This process can be seen in Fig. 3; As r gets smaller, n_s gets closer to the number of points in the geodesic neighborhood κ_r.

True samples:, even though we might be running each level $0, \dots, \ell - 1$ of $R(v, r)$ on vertices chosen from a sub-sampled set, each individual calculation of $H(v, r)$ and $\eta(v, r)$ (line 10 of Algorithm 2) iterates over a geodesic neighborhood κ_r from the **ground truth mesh**, not a sub-sampled mesh. Thus, we interpolate between different ground-truth saliency values, not simplified ones, which already introduce non-recoverable errors, e.g., pre-decimated meshes (cf. Section 5). Moreover, as tropical angle calculation precedes the saliency computation, the range of values $[\theta_{min}, \theta_{max}]$ stays the same, as well as the effect of parameters such as β.

Independent: our measurements are independent of tessellation density, meaning that the sampled points are accurate regardless of what part of the mesh they come from, thus making their interpolation accurate as well.

4.1. Poisson-disc subsampling

Two issues are relevant to our problem domain: (1) the selected vertices must represent the local entropy H of the mesh with similar quality to that of the entire mesh, and (2) our method must be fast (a slow subsampling technique would hinder its benefit).

We employ a Poisson-disc technique inspired by [31] that combines fast random selection with a near-uniform distribution in order to minimize interpolation error for a given number of samples. Vertices are randomly selected in parallel, and all vertices within the geodesic r_e neighborhood of each selected vertex are excluded from future selection. We define the exclusion radius r_e with respect to the hyper parameter n_s as $r_e = \sqrt{2} \cdot r/\sqrt{n_s}$ (lines 3–4 of Algorithm 2). Sampling then continues until all vertices are excluded. Fast random selection is facilitated by iterating through a randomly shuffled list of vertices. The threads start at indices evenly distributed through the list and select their next sample vertex by iterating and skipping vertices previously flagged as excluded. When a vertex has been sampled, interpolation is performed by distributing the weighted entropy to vertices within the $2r_e$ neighborhood (see Section 4.2).

4.2. Interpolating saliency values

After calculating values for the sampled vertices, we then finish the saliency map by propagating those values to unsampled vertices. For each unsampled vertex v_i, our method collects nearby sampled vertices $[v_j, v_{j+1}, \underset{\sim}{\dots}, v_{j+n}]$ that were assigned calculated values. The interpolant $\widetilde{H}$ is then calculated as the inverse-distance-weighted mean of sample values:

$$\widetilde{H}(v_i) = \frac{\sum_{k=0}^{n} \eta(v_{j+k}, r) \cdot H(v_{j+k}, r) \cdot w(GD(v_i, v_{j+k}))}{\sum_{k=0}^{n} w(GD(v_i, v_{j+k}))}, \tag{6}$$

where we search for sampled vertices within $2r_e$ of the vertex v_i, and the weighting function w is defined as:

$$w(d) = \max\left(r_e/d - {}^1\!/_2\ , 0\right). \tag{7}$$

This guarantees interpolation coverage ($2r_e$ exceeds the largest distance between two nearby subsampled vertices) and also ensures a continuous interpolation (values further from v_i receive zero weight).

4.3. Memory and grouped traversal optimizations

When n_s becomes close to verifying every one out of four vertices, choosing samples and interpolating becomes more expensive than the ground truth calculation. We introduce two more optimizations that aid efficient operation once we fall back to full sampling. Only the grouped traversal optimization is specific to the non-subsampled case; the memory layout is always applicable.

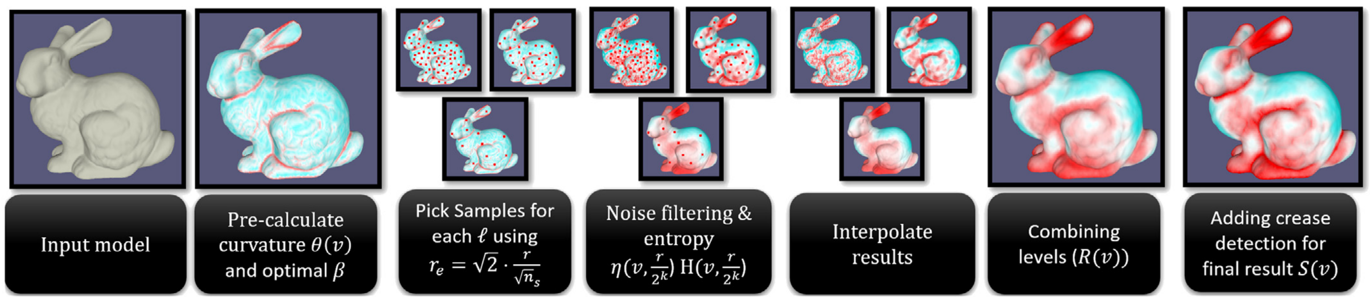

Input model
Pre-calculate curvature $\theta(v)$ and optimal β
Pick Samples for each ℓ using $r_e = \sqrt{2} \cdot \frac{r}{\sqrt{n_s}}$
Noise filtering & entropy $\eta(v, \frac{r}{2^k})$ $H(v, \frac{r}{2^k})$
Interpolate results
Combining levels $(R(v))$
Adding crease detection for final result $S(v)$

Fig. 4. Full overview of our method, with every step visually represented. Purely illustrative samples were chosen).

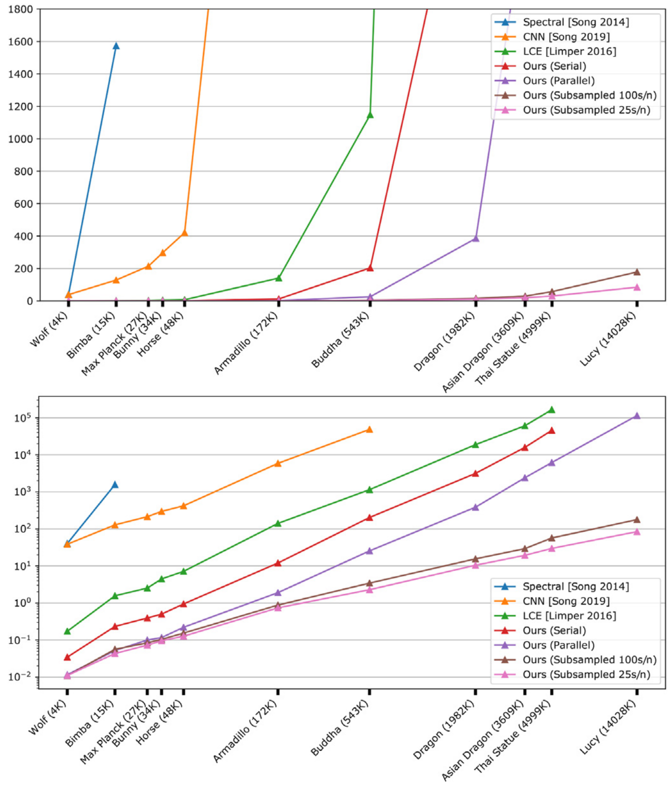


Fig. 5. Execution times in seconds for the evaluated techniques in a linear scale (top) with cutoff execution time on 1 h, and log scale (bottom) with cutoff of 48 h.

Memory Layout: We use a flat data structure that packs all vertices and their lists of neighbors into a single contiguous memory region and then consistently use offsets to identify vertices, significantly reducing indirection overhead when traversing the mesh. We also layout the vertices such that neighbor points are roughly grouped to improve cache locality. A series of truncated breadth-first searches are used to gather groups that are then placed sequentially. A secondary inner search gathers inner groups of four direct neighbors to aid the grouped traversal.

Grouped Traversal: The traversal step identifies the r-neighborhood of a given vertex in preparation for the saliency calculation because the r-neighborhoods of directly adjacent vertices inherently have a significant overlap. To optimize, we visit the r-neighborhoods of four neighboring vertices together in a single traversal to share work between them. Again, we use a modified first-in-first-out queue instead of the traditional priority queue to implement this, which is also faster in this specific case.

As the operations performed during traversal and entropy calculation for each of the four root vertices are identical, we can further reduce computational overhead by taking advantage of SIMD operations (our implementation employs x86-64 SSE compiler intrinsics, hence the use of four root vertices).

5. Results & discussion

We present and discuss results showing that our method is faster and stabler than alternatives when applied to highly complex models. Similar to previous work, we chose models from the Stanford 3D Scanning Repository [32] and SHREC [33] for benchmarking. Section 5.1 compares execution times of different approaches. Section 5.2 compares our adaptive subsampling to the pre-decimation of models used in previous work. Section 5.3 highlights the accuracy of our saliency calculation against related work. Finally, Section 5.4 concludes with an industry-focused discussion.

5.1. Execution time

We compare the execution time of our method to [2,13], and [24]. We choose these methods for this comparison because they are popular, recent alternatives with source code available for testing. [2] is the most similar recent work to ours in that it also uses a multi-scale saliency measure, and [13,24] are similar in that both use global evaluation of saliency. All results were gathered using a computer with the following specifications: AMD Ryzen 2700X (8 core, 16 thread), 32 GB RAM.

We executed each of the methods on the original models (avoiding the pre-decimation step for the techniques that use it) to evaluate the scalability of each approach. We used two subsampling configurations in this test, at $n_s = 100$ and $n_s = 25$. These introduce marginal error in the final computation (root mean square error (RMSE) < 0.004 and < 0.0071 respectively for all models). We also included a non-sub-sampled version of our method to test the upper bound of increasing n_s and also to evaluate better the contributions from parallelization optimizations.

Fig. 5 plots the results (detailed timings can be found in the Appendix). Examining the figure, we observe that our approach scales better than others as the complexity of the models increases (note that the difference between execution times increases in proportion to complexity) for any value of n_s.

The results show that the computation time required for all methods alternative to ours (the version with all optimizations) seemingly scales exponentially with model complexity, as can be seen in Fig. 5. Our approach (pink and brown lines) seemingly scales sub-linearly with the number of vertices. On a log–log scale, this arguably corresponds to a sub-exponential growth rate. The fastest alternative to our approach is [2]. Comparing [2] to our method configured to use $n_s = 25$, we are faster by 4.54 times on Bimba, 111.56 times on Buddha, and 1593.35 times on Thai Statue. Our data show that their method's computation time dramatically increases when applied to models with over one million polygons, each run taking hours to complete. Larger models, such as Lucy, had to be terminated due to computation times exceeding our 48-hour limit. Indeed, the performance advantage of our algorithm grows with polygon count. [13] required 26 min to complete the Bimba model with 15 516 vertices, while [24] only needed 2:38. In comparison, our $n_s = 25$ configuration completed this model fast enough for interactive use. Notably, the data structures necessary to execute their technique did not support processing more complex models since their global method is very computationally expensive. Because of this, [5,13,24] use pre-decimation of models as a speed-up strategy, which is the only way to deliver competitive execution times. The following section questions how viable this speed-up strategy is against our adaptive subsampling.

5.2. Adaptive subsampling vs. pre-decimation

In this section, we compare our adaptive subsampling to the pre-decimation approaches used in [5,13,24]. As outlined in Section 3, and in previous work [2,3,8], a primary application of saliency is to enable the decimation of highly complex meshes without losing visually salient details. We argue that decimating a mesh using a standard algorithm is prone to discarding these details. Since important features were discarded, a saliency map computed on the pre-decimated mesh will inevitably yield wrong results. Moreover, it is seemingly counter-intuitive to decimate a

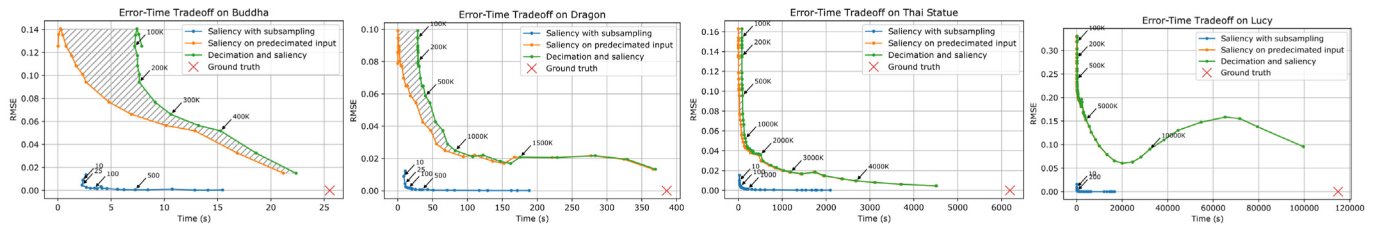


Fig. 6. Comparison between speed-up strategies: Subsampling vs. pre-decimation. Subsampling shows lower RMSE for the same execution times, even if not considering the decimation component runtimes.

model and then calculate saliency to decimate the model without loss of information.

Nevertheless, we compare the achieved speed-ups related to the RMSE to the ground truth saliency estimation. Song et al. [5, 13] do not present such numerical comparisons to ground truth due to the high computation costs of these approaches. Indeed, ground truth calculations were only feasible for less complex models (where the simplification was minor), introducing less error.

We calculated the ground truth saliency using our method (parameters clarified in the next section) and using the two different speed-up approaches: pre-decimation and our adaptive subsampling. Fig. 6 plots execution time against RMSE that we observed when applying each method to four different models (Buddha, Dragon, Thai Statue, and Lucy). Two lines are plotted for the pre-decimation strategy: one line specifies the combined time for the decimation (QSlim [6] as done by [5,13]) and saliency, and one line for just the time necessary for saliency. Replacing QSlim with a faster alternative would yield results in the shaded area between both lines. The same plot displays our subsampling method performance for different values of n_s.

From Fig. 6, we can observe a worst-case 0.02 RMSE for all results using our method, with typical cases being close to zero. Even discarding the decimation time, our execution times remains similar to that observed for pre-decimation. As pre-decimation tends toward zero (green line), RMSE increases considerably. As for subsampling, there is a point where using fewer samples no longer reduces execution time (due to the increased overhead of choosing samples). However, RMSE only increases marginally, even when using fewer samples. Fig. 7 shows a concrete example of the two approaches running on a large model (Thai Statue from Stanford Repository). Our speed-up approach delivers visually indistinguishable results from the ground truth (RMSE $\leq$ 0.0046) for similar execution times. Although we could tweak the parameters on the decimated model to reach results closer to the ground truth, that details were lost in the decimation means that no configuration could outperform our subsampling approach in terms of accuracy.

Finally, Fig. 8 depicts the relation between RMSE and n_s. Despite varying complexities and topologies, we observed similar trends throughout all tested models. Thus, we conclude that n_s can control the trade-off between acceptable error and speed-up. Recalling Fig. 5, we can observe that our subsampling configuration ensures a marginal worst-case error (0.004 for $n_s = 100$ and 0.0071 for $n_s = 25$). Notably, our worst-case error outperforms the best-case error across all pre-decimation approaches (which also require much longer execution times).

5.3. Accuracy of saliency maps

In what follows, we compare our generated saliency maps to recently published work [2,24]. Fig. 9 shows a side-by-side comparison between our method and LCE [2] that also estimate saliency via multi-scale entropy calculations. LCE saliency maps

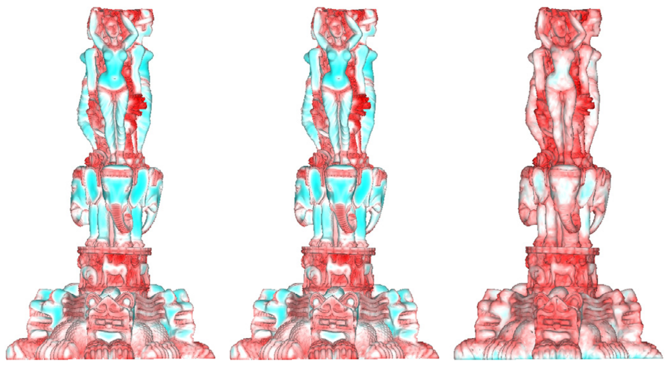


Fig. 7. Left to right: Ground truth, Subsampling at 75 samples per neighborhood (120 s) and pre-decimation to 50 000 vertices (114 s). All other parameters unchanged.

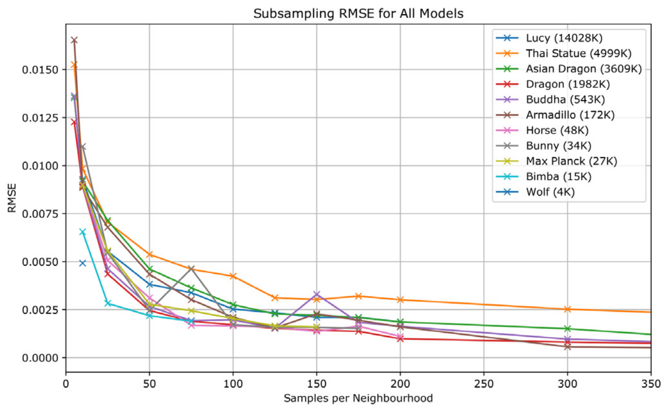


Fig. 8. RMSE for all models w.r.t. samples-per-neighborhood parameter, showing a similar trend for all models regardless of complexity or topology.

appear on the top, while our approach is shown on the bottom. Regarding unique parameters to our approach, we set $\alpha = 1$ (crease detection) for our method, used the automatic estimation of β (contrast), and did not use noise removal by setting $\epsilon = 0$. We ensured fair comparison by setting other parameters similarly for either approach (geodesic neighborhood $r = 0.02$, number of levels $l = 5$). We also adopted the same number of bins σ for entropy calculation and avoided smoothing saliency maps in either case.

For the three smallest models (Wolf, Horse, and Bunny), the LCE maps and ours present similar features as salient. We can observe specific differences, as our approach includes crease-like features missed by LCE, which are essential for decimation. These are due to our method's α parameter that helps to ensure creases remain salient.

LCE could not highlight the salient elements of the two dragon models and the Buddha model, the more complex models created from laser scans. This observation highlights our novel saliency

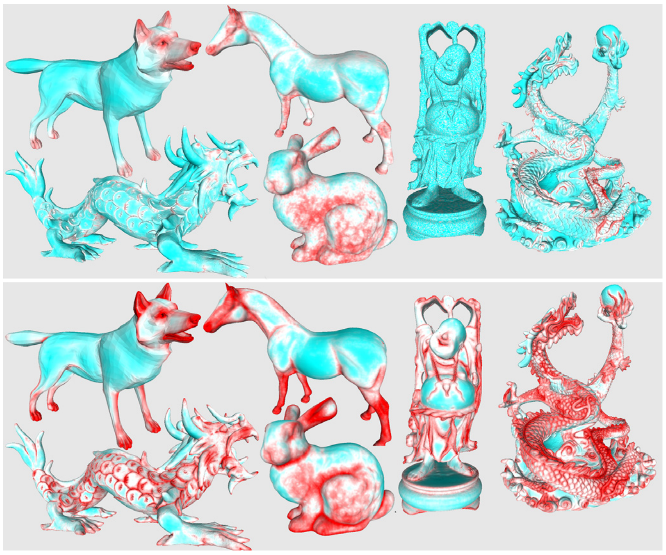


Fig. 9. Top row: results from [2], Bottom: our results running on the same meshes. Limitations of LCE revealed in the more complex meshes. Matching parameters set to the same values. $\alpha = 1, \epsilon = 0$, and automatic β for the ones unique to our approach.

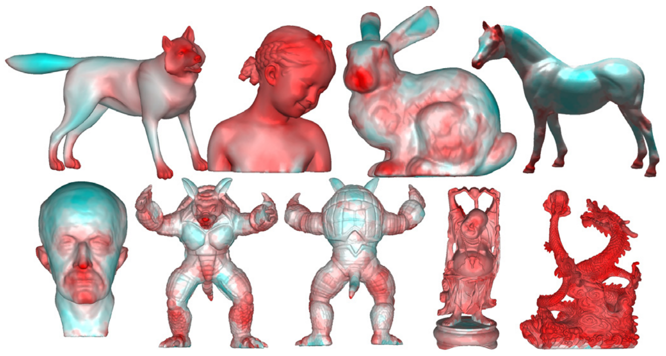


Fig. 10. Results from Machine Learning approach from [24] using *pre-decimated meshes*. Results on Dragon, Bimba and Buddha models fail to isolate salient regions, while the others see a focus on more global regions instead of outlining local features.

formulation's advantage, which adopts the *tropical angle of curvature* $\theta(v)$ over main curvature measurements. We hypothesize that LCE could not highlight these features due to the problem described in Fig. 2, where their approach fails due to varying levels of tessellation, which adversely impacts how curvature values are computed and ultimately mask salient features. One could argue that merely reducing the range of acceptable curvatures to remove gross outliers is a viable solution (thus enabling LCE to successfully identify the missed features in cases where malformed faces are present). However, we can observe that even well-formed meshes can feature widely varying curvature (see Fig. 2), leading to results seen in Fig. 9. In contrast to LCE, our approach assigns appropriate saliency values to each complex model, despite the irregular tessellation and noise levels. While being locally accurate, our approach can highlight salient features across larger areas (such as Buddha's face in Fig. 9).

We could replicate the results from [24], using pre-decimation, as the authors made the pre-trained networks available. Fig. 10 shows these results. The quality of Dragon, Buddha, and Bimba is seemingly not better than [2], with most of the meshes being considered salient. This figure exemplifies the argument made in Section 5.2: pre-decimating meshes will remove fine-scale details. Therefore, potentially essential features will be lost for saliency detection. We found this especially problematic for Bimba, where the simplification rate was considerably lower than for the other two. The same defects occur for Armadillo, albeit to a lesser extent. Prominent regions appear roughly outlined on the smaller models, but some salient local features are lost.

As discussed in Section 2, [24] and other similar methods are still seemingly suitable to other application areas such as object retrieval and matching, and specific to [24], estimating the saliency of scenes containing several models. In the appendices we highlight some of the differences in the saliency maps of our technique, focused on decimation, and visual attention points [26], and how they are not suitable for decimation. However, when processing highly complex models, pre-decimation loses essential details, is less sensitive to local features, and does not scale to shapes with over 10k vertices.

Finally, Fig. 11 shows the results of our algorithm on a new dataset using the default settings in our technique, showing that no parameter tuning is necessary to achieve good results. For example, unlike prior methods, our algorithm successfully detects the edge lines of the pedestals (top) and shell (top rightmost)(see more detailed discussions in Section 5.4, and in the appendices for a detailed description of the parameter space.).

5.4. User evaluation

We implemented an interactive tool that allows users to explore the parameter space easily (seen in Fig. 12), calculate saliency, and perform saliency-weighted decimation. We found that we can enable an interactive, or close to interactive, experience for meshes of reasonable complexity. We observed sub-second execution times for Armadillo and all other less complicated models. For larger meshes, execution times are fast enough to avoid disturbing artist workflow (e.g. 3.44 s for Buddha—543k vertices, 29.75 s for Thai Statue—4.99m vertices, and 84.49 s for Lucy—14m vertices).

In order to validate that artists were able to navigate the parameter space with ease, and that this would be an appropriate tool for their day-to-day use cases, we performed remote user tests with 14 artists from three different companies in the entertainment industry. We cannot provide names and more specific feedback or images due to existing Non-disclosure agreements. Subjects were sent the tool with sample models, a short video on using it, and a questionnaire about the experience. We then requested artists to freely use the tool to calculate saliency in their private models to assess practicality in real-world production pipelines. The unsupervised tests had no time limit, giving users enough time to make several attempts on both simple and challenging models. Due to the wide variety in time availability by the artists in each company, the form of the provided feedback was varied; a few responded to every single question in a structured manner, but the majority provided short textual descriptions of their experience and several examples of their successes or difficulties. We requested feedback on the tool's ease of use (visualization, processing files) and specifically on navigating the parameter space. Other usability-type feedback (ability to undo, support for alternative file formats) will not be discussed here as they fall outside this paper's scope.

All users could easily navigate the parameter space regarding saliency computation, providing several successful results in their private datasets. Artists appreciated the ability to control the method. Moreover, they requested (1) allowing manual input to provide further detail of the results if non geometrically salient areas are considered essential. Additionally, (2) to provide presets for different model types (e.g., humans, buildings, animals, 3D scans) to speed-up processing of several models in a sequence

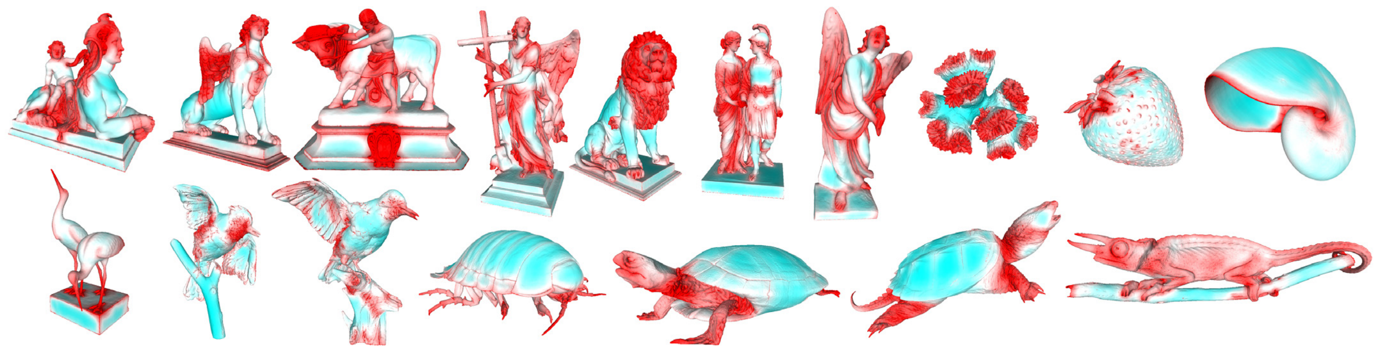
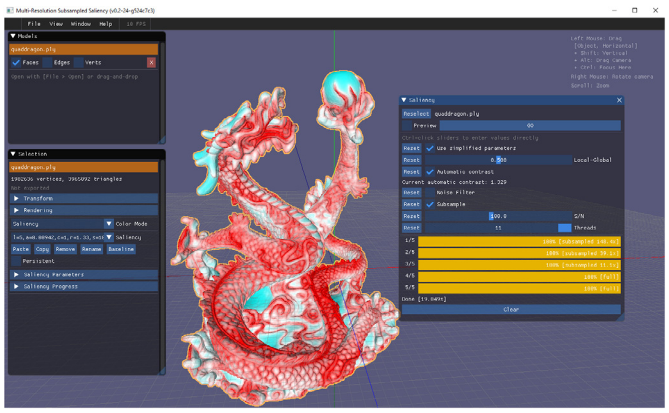


Fig. 11. Saliency results using the default parameters of our technique ($\Gamma = 0.5$, $\epsilon = 0$) on scanned models of statues and natural objects (Statues from noe-3d.atSketchFab and natural models from RISD Nature LabSketchFab). All models have 500k to 1M faces.



```
Multi-Resolution Subsampled Saliency (v0.2-24-g524c7c3)
10 FPS
quaddragon.ply
Open with [File > Open] or drag-and-drop
quaddragon.ply
1982636 vertices, 3965892 triangles
Not exported
Saliency                    Color Mode
l=5,a=0.00942,c=1,r=1.33,s=1  Saliency

Left Mouse: Drag
 [Object, Horizontal]
 + Shift: Vertical
 + Alt: Drag Camera
 + Ctrl: Focus Here
Right Mouse: Rotate camera
Scroll: Zoom

Saliency
Reselect quaddragon.ply
Ctrl-click sliders to enter values directly
Reset ✓ Use simplified parameters
Reset 0.500 Local-Global
Reset ✓ Automatic contrast
Current automatic contrast: 1.329
Reset   Noise Filter
Reset ✓ Subsample
Reset 100.0 S/N
Reset 11 Threads
1/5 100% [subsampled 148.4x]
2/5 100% [subsampled 39.1x]
3/5 100% [subsampled 11.1x]
4/5 100% [full]
5/5 100% [full]
Done [19.849s]
```

Fig. 12. A GUI allows artists to explore our method's parameters.

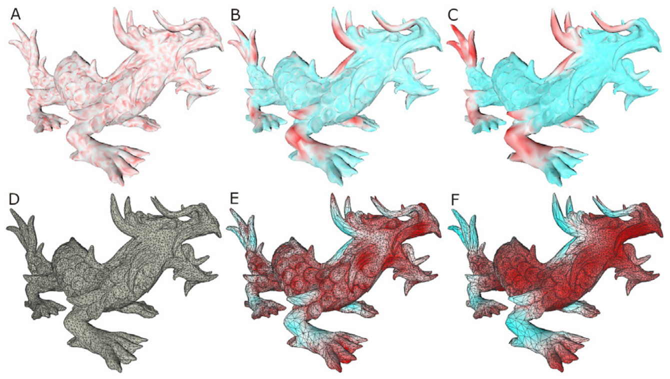


Fig. 13. Saliency weighted decimation comparison for the Asian Dragon. Top: colorized decimation error. Bottom: saliency map. From left to right: no saliency, $\Gamma = 0.25$, $\Gamma = 0.75$.

without requiring to re-set previously chosen parameters. These can be quickly addressed by improvements in the Tool, not the saliency approach itself. It is crucial to notice that no user has reported any case of complete failure in calculating saliency, attesting to the method's robustness in real-world scenarios. We attribute this to the more stable surface measurement that we proposed, the tropical angle of curvature (failure scenarios of curvature-based methods were discussed in Figs. 1 and 9).

Decimation: Using highly complex models, we evaluated the impact of using our saliency maps in mesh decimation. As discussed in Section 1, saliency-aware decimation has one main advantage when it comes to being applied in practical scenarios (e.g., entertainment industry); it gives the artist control of how the decimation is going to be performed. Typically the only parameter available is the target number of faces, with no indication of how the operation will affect a given model. The saliency map highlights how each model part will be affected before this lengthy operation.

Fig. 13 shows a comparison using saliency-weighted quadric edge collapse decimation. The top row shows the estimated quadrics error metric relative to the ground truth mesh, and the bottom row shows the used saliency map for the decimation (none in the leftmost scenario). Our findings are coherent with previous work that evaluated saliency-weighted decimation [2], where our approach will yield a higher global geometric error due to optimizing for saliency instead, as compared to traditional methods. However, when evaluated locally, our technique introduces less error in higher saliency regions (e.g., body, face) and higher on low saliency ones (e.g., paws, horns), effectively compensating decimation errors in less salient regions. As seen in this figure, using a more localized saliency map (Fig. 13E) will allow producing a mesh with quickly changing tessellation densities, allocating more triangles to sharper features (e.g., scales). A global map (Fig. 13F) will use more triangles in the area as a whole, which could be useful for high-quality animations of e.g., the dragon's body or face. Further examples can be seen in the appendices.

We also evaluated our approach with a quad-based decimation algorithm — ZRemesher by ZBrush [34] (we input our saliency map directly as the Poly Paint input for the ZRemesher tool, which is used to guide the extent to which decimation is applied across the mesh). Fig. 14A shows a saliency map produced by our method for the Asian Dragon model, and 14C presents the decimation result of ZRemesher when using this map. For comparison, ~14B shows the decimation result from ZRemesher adaptive mode without any saliency map applied. Both models were reduced to 5% of their original size. Our saliency map helps the decimation tool to preserve details relevant to shapes such as the eyes and individual scales. Furthermore, our map enables more aggressive decimation of less salient features, such as the cheeks and horns of the dragon. Although they do not preserve as much detail as the triangle decimation results, quad-meshes are the preferred format of several artists in the industry. Saliency-weighted decimation is perhaps more relevant in this case, as it allows this format to preserve features that would be smoothed by re-meshing.

Finally, Fig. 15 shows a practical example, rendering a saliency-aware-decimated mesh using path tracing. Even at very high compression (100:1) rates, details such as the crack on top of the lip and toes, marked as high saliency, are still clearly visible in the compressed render.

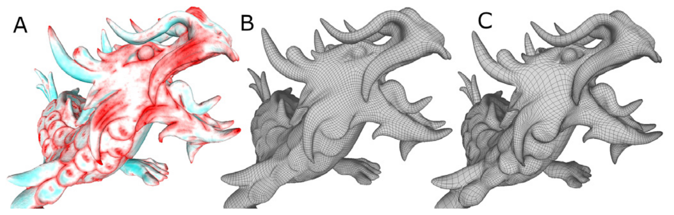


Fig. 14. Decimation Results using a quad mesh. (A) Saliency map for Asian Dragon. (B) 95% decimation without saliency map. (C) 95% decimation with saliency map. Decimation by ZRemesher.

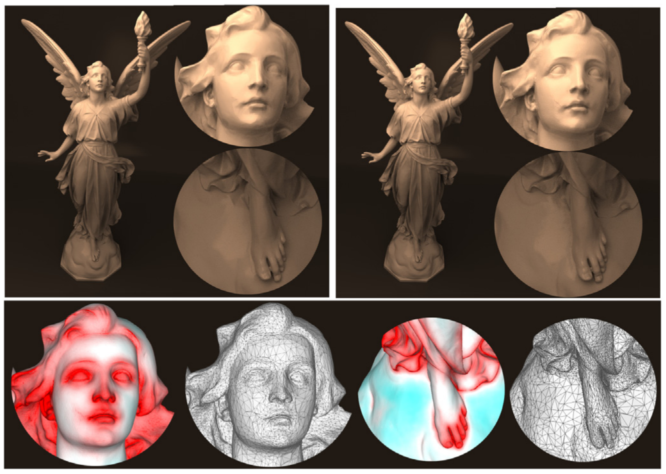


Fig. 15. Renders of Lucy. Left: After 99% decimation with our saliency map using a triangle mesh. Right: original with 14 million vertices. Bottom: Used saliency map and wire-frame output. Our method preserves salient features even under high compression levels, e.g., upper lip and feet details.

6. Conclusions, limitations and future work

We have presented a novel saliency measure and demonstrated that it captures local and global salient regions and creases and is resilient to mesh noise. We have also shown that our method can process large and dense meshes with distinct shapes and complexities while being more robust than state-of-the-art approaches concerning minimizing effects from spurious artifacts from varying tessellation density.

One limitation of our approach is how well it can be integrated into an existing production pipeline. While our approach fits well in a ZBrush-centered process, which is quite common, other tools for our GUI need to be developed to fit different processes (e.g., adding painting functionality to our program).

We also need to explore good saliency-weighted decimation approaches, to exploit our method to the fullest, specially in the quad-mesh format. This is because the decimated mesh quality depends not only on good saliency maps but also on the decimation algorithm. Unlike triangle-based decimation, quad mesh decimation remains a challenging topic [1].

In future work, we will examine how saliency values vary over time on animated meshes so that we can assign saliency to their range of movement. Pose-invariant saliency is a topic that has been brought up in recent neural-network based work for shape matching [35], and we wish to use this concept in mesh decimation. Additionally, we will examine alternative data structures and GPU implementations to enable our approach to scale to even larger models. As found in recent research, there is more considered than just geometry when it comes to visual saliency. We plan to investigate how these two aspects can compliment each other, so an efficient and scalable model of human attention can be developed for usage in production environments. Finally, we plan to study how this measurement can be combined with other visual information present in the mesh (e.g., textures, material maps), which may influence the final visual saliency.

Ultimately, our approach is the first to allow saliency detection to be practically applied to meshes containing millions of polygons. We believe our technique is so powerful as to guide a saliency-aware decimation process, meshing seamlessly with both the standardized tools and workflow of the entertainment industry.

CRediT authorship contribution statement

Rafael Kuffner dos Anjos: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization, Validation. **Richard Andrew Roberts:** Investigation, Methodology, Writing – original draft, Formal analysis. **Benjamin Allen:** Methodology, Software, Investigation, Data curation, Writing – original draft, Validation. **Joaquim Jorge:** Investigation, Writing – review & editing, Methodology, Validation. **Ken Anjyo:** Conceptualization, Supervision, Funding acquisition, Project administration, Writing – original draft, Writing – review & editing, Resources, Methodology, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgements

We would like to call special thanks to Ayumi Kimura for her continuous support. We are also grateful to Square Enix and OLM Digital for their thought-provoking discussions on our work. We also thank Warren Butcher and other peers from CMIC, for their support during the development of this publication. This project was partly supported by the Entrepreneurial University Program funded by TEC in New Zealand.

Appendix

A.1. Tropical angle of curvature

Motivation: We first recall the curvature of a planar curve and its discrete approximation. Let γ be a planar curve, which is defined on a 2D plane ($\approx R^2$). Suppose that this curve is parameterized with its arc-length s: $\gamma = \gamma(s)$. As is well known, we have $|\gamma'(s)| = 1$ for all s, where we denote $\gamma'(s) = \frac{d\gamma(s)}{ds}$. Assume that we have sample points $\gamma_i \equiv \gamma(s_i)$ of the curve uniformly across its arc length such that $\Delta s = s_i - s_{i-1}$ is constant. Then we can roughly estimate each (absolute) value of $\kappa_i \equiv \kappa(s_i)$ as being proportional to the turning angle $\Delta\varphi_i$, which is defined with the two adjacent tangent lines at γ_{i-1} and at γ_i, which means

$$|\kappa_i| = |\frac{d^2\gamma}{ds^2}(s_i)| \approx |\frac{\gamma'(s_i) - \gamma'(s_{i-1})}{s_i - s_{i-1}}| \approx |\frac{\Delta\varphi_i}{\Delta s}| \approx |\Delta\varphi_i|.$$

In this paper, we directly extend the above idea to measure "curviness" of a mesh surface M. For a moment we assume that the surface M is densely and uniformly sampled. While

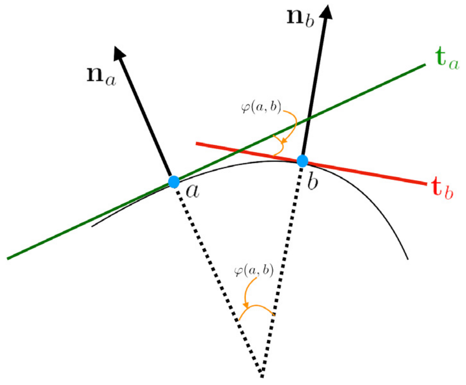


Fig. 16. Angular difference of two adjacent tangent planes.

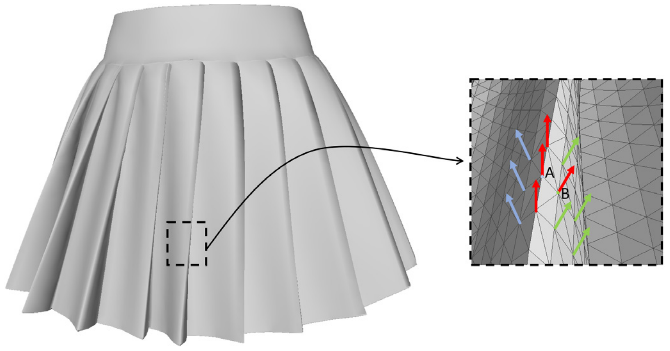


Fig. 17. Sharp edge detection by the tropical angle of curvature. Figure shows the normal vectors of vertices in the 1 ring neighborhood of A (blue) and B (green). Shared vectors are drawn in red.

the curvature for a planar curve can be thought of as being (proportional to) the turning angle mentioned above, we first consider the angular difference between the two adjacent tangent planes at the closest sample points (i.e., at the two vertices, each of which belongs to 1-ring neighborhood of the other vertex), as shown in Fig. 16 where the tangent planes $\mathbf{t}_a$ at a and $\mathbf{t}_b$ at b are defined with the given normal vectors $\mathbf{n}_a$ and $\mathbf{n}_b$, respectively. Then we refer to the angular difference of the two tangent planes as $\varphi(a,b)$, which is obtained as $\cos^{-1}(n_a \cdot n_b)$. We may assume that the angular difference $\varphi(a,b)$ ranges from 0 to π. Unlike a planar curve case, our curviness measure at vertex v of M should consider the tangent planes near the vertex v. Therefore we define the curviness measure, denoted by φ_{max}, as the maximum value of the difference angle $\varphi(a,b)$, where (a,b) is any pair of vertices in the 1-ring neighborhood of v:

$$\varphi_{max}(v) := \max_{a,b\in adj(v)} \varphi(a,b). \tag{A.1.1}$$

Further, in our formulation and implementation, we introduce the following geometric measurement θ, instead of φ_{max}:

$$\theta(v) = \max_{a,b\in adj(v)} \left\{ \frac{1-(n_a\cdot n_b)}{2} \right\}. \tag{A.1.2}$$

As explained in 3.1, Eq. (A.1.2) simply gives normalization of the parameter space φ of $[0,\pi]$ to the parameter space θ of $[0,1]$, without going through arc-cosine function. We expect that the tropical angle of curvature θ (Eq. (1) or (A.1.2)) could give us rough yet useful information as a curviness measure, like a kind of analogy with the absolute value of the principal curvatures for a smooth manifold. Usefulness of the new geometric measure θ is clarified in this paper. We then note that, in practice, we use the tropical angle of curvature under the assumption that the mesh vertices are densely but not necessarily uniformly distributed.

Validity: As an alternative of θ in Eq. (1), we could consider:

$$\psi(v) = \max_{a\in adj(v)} \left\{ \frac{1-(n_a\cdot n_v)}{2} \right\}. \tag{A.1.3}$$

However, as illustrated in Fig. 17, we see that the tropical angle θ works better than ψ for crease or sharp edge detection. Consider the vertices marked A and B in Fig. 17. If we consider the vertex itself and the vertices in its 1-ring neighborhood, then both A and B exhibit the same maximum difference of normals, and hence produce the same value of ψ. This means the entire region appears uniform from a curvature perspective. We wanted to capture that A was much "curvier" than B, so we came up with θ, which looks for the max difference of normals over all adjacent vertices. This produces a large value for A, and a much smaller value for B.

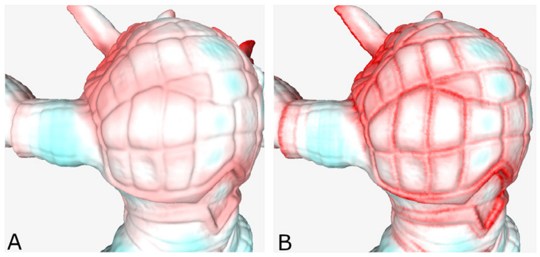


Fig. 18. Effect of adding $\theta(v)$ for contour detection on the armadillo model.

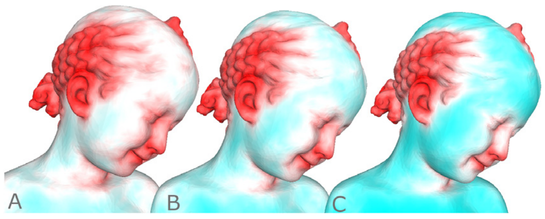


Fig. 19. Manually changing β to change the perceived "contrast" of the saliency map.

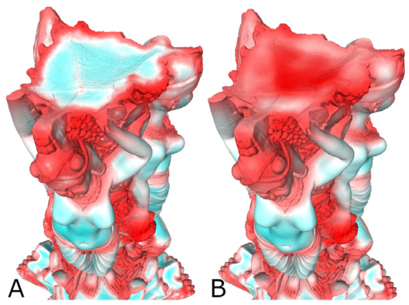


Fig. 20. Example saliency maps of a model created from a scan (Thai Statue from Stanford Repository) where local noise is filtered out by using $\epsilon = 0.01\%$ of the mesh area, as seen in the top of the model (A). (B) features a large value of ϵ in order not to apply this operation.

A.2. Parameters and evaluation results

The execution speed of our algorithm allows us to create a user interface where these can be easily changed and explored easily, and at interactive speeds for simpler models. There are five free variables in our saliency definition, each one affecting a different part of the calculation. A Summary of their individual effects can be seen in Table 1. Figs. 18,19, 20 visually exemplify

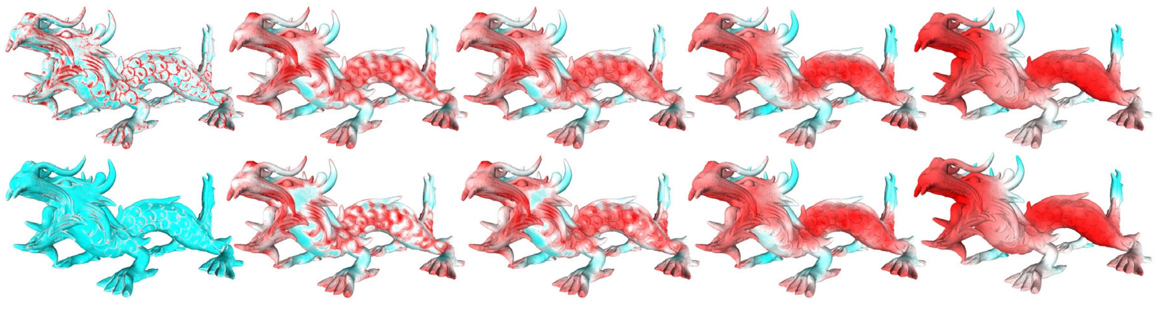


Fig. 21. Left to right: globality increasing from 0 to 1 in steps of 0.25. Top row has $\epsilon = 0$ meaning no noise reduction applied. Bottom has $\epsilon = 0.02$.

Table 1
Individual Parameters of our approach. Each parameter independently controls a separate dimension. Given some configurations overlap, we proposed a simplified version with one parameter.

Par	Description	Effect
ℓ	Number of levels to calculate $R(v)$	Allows less or more visibility of local features in $S(v)$
r	Radius of the topmost level where $R(v)$ will be calculated	Controls the size of the most global salient elements that will be detected in the input model,
β	Power applied to the calculation of $\theta(v)$	Works as a contrast control for $R(v)$, as it affects the binning operation.
α	Weight parameter applied to the combination of $\theta(v)$ in $S(v)$	Increases the perceived saliency of immediately local features such as creases and sharp edges.
ϵ	Noise control parameter of $\eta(v, r)$	Controls what is the expected level of noise in the data so it can be minimized.

some the effects of these parameters. An interactive exploration can be seen in the accompanying video. As for execution time tests with the examples in this paper, please see Tables 2 and 3.

While an experienced artist could manually tweak each of these parameters, we propose a simplified version for easier control of the saliency map, estimating two of the parameters, and using a hyper-parameter to define the other three.

First, the value ϵ can be set per model based on the estimated noise of the used 3D scanner. This value may either be provided by the manufacturer, or procedurally estimated accurately for off-the-shelf sensors [36,37]. A simple approach to estimating this noise level is to observe the standard deviation of the depth of a certain pixel in a sequence of frames observing a static planar object, parallel to the viewing plane. The user can then set ϵ to a multiple of this. For the use case of a sculpted model, which is common in the entertainment industry, this value will be zero, unless one desires to smooth out small details from the model. The effect of this parameter can be seen in Fig. 20

Secondly, we can automatically determine a reasonable value for β from the observation that the entropy of the global curvature distribution approximates the mean entropy of all neighborhood curvature distributions. We solve for a value of β such that the global entropy is equal to some desired max histogram. We aim for mean saliency equal to the saliency midpoint (white in our figures), but do not know the saliency range beforehand as it depends on β. We evaluated the deviation of the mean saliency from the midpoint $^{(R_\mu - R_{\text{mid}})}/_{(R_{\max} - R_{\min})}$ for various models. These came from a larger collection than the ones included in the paper, selected from public repositories, created by various artists, varying in complexity and content and having medium to high polygon count, meaning they would be candidates for decimation. We found that, on average, a target entropy factor of 0.453 yields 0 deviation with variance <0.1. The only models that deviate from this trend are those with a meager polygon count (e.g., wolf). The low number of vertices makes skewing the distribution in any direction due to outliers considerably easier. For those cases, however, our algorithm runs interactively, and it becomes trivial to adjust the value of β manually.

The values for ℓ, r and α are controlled by a *global hyper-parameter* (Γ), which is the only user controlled value. For $\Gamma = 0$ we set $\alpha = 1$, $r = 0.0001$, and $\ell = 3$. A local map will be mindful of creases ($\alpha = 1$) and highlight only local structures

Table 2
Breakdown of execution time tests for our results in seconds. These times are included in those reported in Table 3. θ is the time to calculate $\theta(v)$ for all vertices. Layout is the time to prepare our optimized data structure.

Model	θ	Layout
Wolf	0.001	0.004
Bimba	0.006	0.011
Max Planck	0.011	0.020
Bunny	0.015	0.019
Horse	0.022	0.027
Armadillo	0.112	0.112
Buddha	0.348	0.265
Dragon	1.581	0.834
Asian Dragon	2.704	1.485
Thai Statue	4.751	2.296
Lucy	8.605	6.178

($r = 0.0001$). While we could use a single level here, using multiple levels ensures that while we increase r through Γ, we still capture the same local details as before.

At the middle of the spectrum ($\Gamma = 0.5$), globally salient regions are identified, but are still less salient than local elements, which are still detected. For this, we just increase r to halfway the maximum globality point (for the examples in this paper, $r = 0.0555$ for $\Gamma = 1$, so $r = 0.2775$ for $\Gamma = 0.5$ as it interpolates linearly). Both α and ℓ keep the same values.

Next we show that a global map ($\Gamma = 1$) will just roughly identify larger regions of saliency in the model ($r = 0.0555$, $\ell = 1$, $\alpha = 0$). For this, we decrease alpha linearly to zero to slowly remove the detection of creases which are purely local elements, we keep increasing the r to look at larger areas, and remove levels at $\Gamma = 0.67$ and $\Gamma = 0.84$ to continue removing the presence of local elements.

Fig. 21 shows the results of varying this parameter. As seen in the bottom row, the noise filtering has higher effect when calculating local maps, where noise is widely considered as detail. When $\Gamma = 0$, without noise filtering, lots of noisy regions are highlighted in white (mid saliency point). These maps however pick up details that are way too local, being dominated by what is detected by alpha (the crease detection), being not very useful for a decimation task.

As we increase the global parameter Γ, the maps become more expressive and useful for decimating. With $\Gamma = 0.25$ we

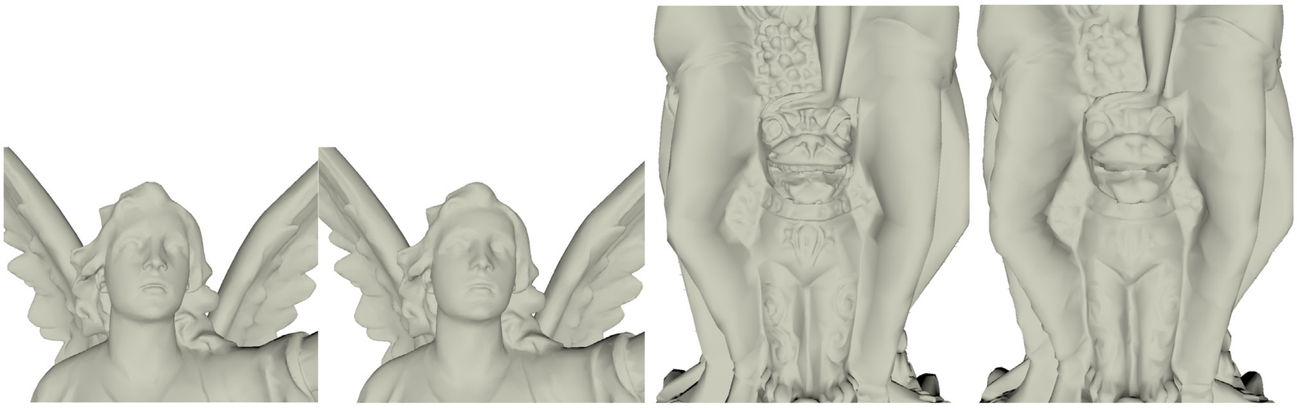


Fig. 22. Comparison between decimation with (left) and without (right) saliency weighting for the Lucy (left) and Hindu (right) models simplified at 99%. Both models have the same vertex count, with the saliency allowing more details to be preserved.

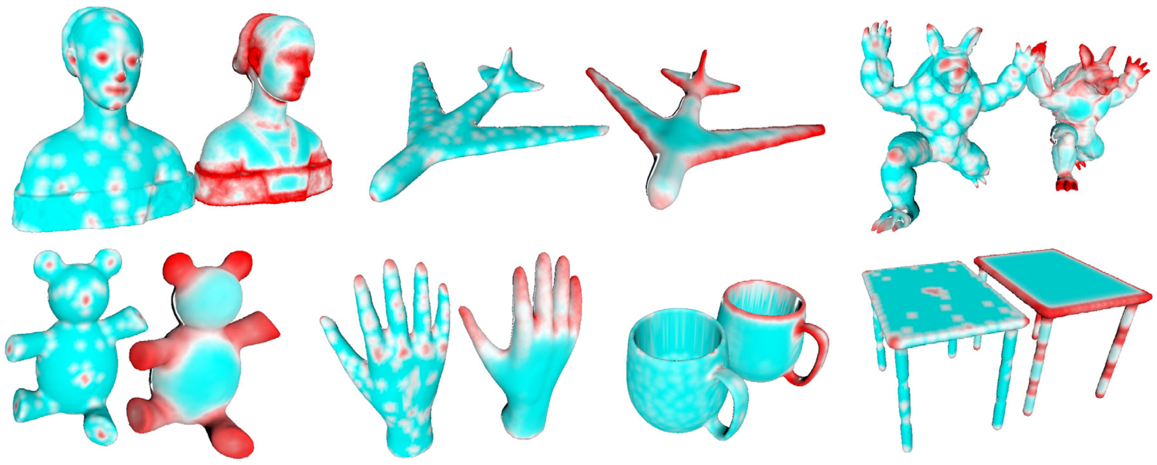


Fig. 23. Comparison between Schelling points [26] probability distribution and our method. Schelling points tend to be discretized around individual areas of interest that may have contextual meaning, which is not always ideal for decimation (example, point at the center of the table). On the other hand, our approach works better at detecting creases and edges at different scales (e.g. see cup, table and airplane models). We argue that for mesh decimation ours is the best performing approach.

Table 3
Full results of execution time tests in hh:mm:ss. Cutoff time is 48 h. [13] could only process the first two meshes, running out of memory (OOM) on the remaining cases. Our approach is the only one that shows enough scalability for practical use, taking less than three minutes to complete the most complex model (Lucy).

Model	#Vertices	Spectral	CNN	LCE	Ours ST	Ours MT	Ours $n_s = 100$	Ours n_s25
Wolf	4,344	00:00:40.25	00:00:38.53	00:00:00.34	00:00:00.19	00:00:00.18	00:00:00.01	00:00:00.01
Bimba	15,516	00:26:13.29	00:02:38.59	00:00:01.84	00:00:00.51	00:00:00.40	00:00:00.06	00:00:00.04
M. Planck	27,726	OOM	00:03:34.02	00:00:02.96	00:00:00.81	00:00:00.59	00:00:00.08	00:00:00.07
Bunny	34,834	OOM	00:04:57.18	00:00:04.98	00:00:01.03	00:00:00.69	00:00:00.10	00:00:00.09
Horse	48,485	OOM	00:07:01.02	00:00:07.84	00:00:01.62	00:00:01.00	00:00:00.15	00:00:00.12
Armadillo	172,974	OOM	01:38:13.23	00:02:22.47	00:00:13.59	00:00:03.54	00:00:00.87	00:00:00.73
Buddha	543,671	OOM	13:32:02.25	00:19:12.67	00:03:28.12	00:00:30.29	00:00:03.44	00:00:02.28
Dragon	1,982,636	OOM	Timeout	05:14:32.96	00:52:59.20	00:06:47.64	00:00:15.57	00:00:10.46
A. Dragon	3,609,455	OOM	Timeout	16:55:39.02	04:25:20.16	00:40:31.54	00:00:29.34	00:00:19.40
T. Statue	4,999,996	OOM	Timeout	45:50:46.40	12:44:37.87	01:43:57.17	00:00:56.94	00:00:29.75
Lucy	14,028,019	OOM	Timeout	Timeout	Timeout	31:56:48.50	00:02:58.32	00:01:24.49

are able to differentiate between the edges of individual scales, and the insides that are less salient. On $\Gamma = 0.5$ each individual scale is now considered a whole salient element. With $\Gamma = 0.75$ each piece of the dragon that has detail is grouped with similar saliency (e.g. whiskers, horns, feet, fangs, tail, etc.). Finally, the global map ($\Gamma = 1$) groups even larger elements (e.g. head, body).

When applied in industry settings, the choice between which Γ to use will depend on available animations, textures, and other effects that the model has, which can guide artists in choosing which elements should keep similar tessellation levels, and what is the desired level of compression. From our experiments, a map with a wider distribution in the saliency spectrum ($\Gamma = 0.25$ in the case depicted in Fig. 21) will provide an optimal compression/quality trade-off. More details on decimation results can be seen in Section 5.4.

Finally we show two more evaluation results. The first one demonstrates effectiveness of our method by comparing the decimated models with and without saliency weighting. In Fig. 22, top row shows decimated results for Lucy's model, while bottom row shows decimated Hindu models. While the results at each row have the same number of vertices, it is clearly to see that the result with saliency on the left successfully keeps the details to be preserved, compared with the result on the right without saliency.

The second one is comparison of our method and the Schelling points method [26]. Schelling points tend to be discretized around individual areas of interest that may have contextual meaning, rather than geometric saliency for mesh decimation. While there is some overlap between both methods to support the theory behind surface-driven saliency capturing visually interesting points, our saliency method is better much suited for mesh decimation, as illustrated in Fig. 23.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cag.2023.01.012.

References

[1] Talton JO. A short survey of mesh simplification algorithms. University of Illinois at Urbana-Champaign; 2004.

[2] Limper M, Arjan K, Fellner DW. Mesh saliency analysis via local curvature entropy. In: Eurographics 2016, Vol. 2016. Switzerland: Eurographics - European Association for Computer Graphics; 2016, p. 13.

[3] Liu X, Liu L, Song W, Liu Y, Ma L. Shape context based mesh saliency detection and its applications: A survey. Comput Graph 2016;57:12–30.

[4] Schmidt T-W, Pellacini F, Nowrouzezahrai D, Jarosz W, Dachsbacher C. State of the art in artistic editing of appearance, lighting and material. Comput Graph Forum 2016;35(1):216–33. http://dx.doi.org/10.1111/cgf.12721, URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12721.

[5] Song R, Liu Y, Martin RR, Echavarria KR. Local-to-global mesh saliency. Vis Comput 2018;34(3):323–36.

[6] Garland M. Quadric-based polygonal surface simplification. Tech. Rep., Pittsburgh PA: Carnegie-Mellon Univ School of Computer Science; 1999.

[7] Koch C, Ullman S. Shifts in selective visual attention: Towards the underlying neural circuitry. In: Vaina LM, editor. Matters of intelligence: Conceptual structures in cognitive neuroscience. Dordrecht: Springer Netherlands; 1987, p. 115–41.

[8] Lee CH, Varshney A, Jacobs DW. Mesh saliency. In: ACM SIGGRAPH 2005 papers. SIGGRAPH '05, New York, NY, USA: Association for Computing Machinery; 2005, p. 659–66.

[9] Miao Y, Feng J. Perceptual-saliency extremum lines for 3D shape illustration. Vis Comput 2010;26(6–8):433–43.

[10] Gal R, Cohen-Or D. Salient geometric features for partial shape matching and similarity. ACM Trans Graph 2006;25(1):130–50.

[11] Tasse FP, Kosinka J, Dodgson N. Cluster-based point set saliency. In: 2015 IEEE international conference on computer vision. ICCV, 2015, p. 163–71. http://dx.doi.org/10.1109/ICCV.2015.27.

[12] Miao Y-w, Hu F-x, Chen M-y, Liu Z, Shou H-h. Visual salience guided feature-aware shape simplification. J Zhejiang Univ Sci C 2014;15(9):744–53.

[13] Song R, Liu Y, Martin RR, Rosin PL. Mesh saliency via spectral processing. ACM Trans Graph 2014;33(1).

[14] Sipiran I, Bustos B. Key-components: detection of salient regions on 3D meshes. Vis Comput 2013;29(12):1319–32.

[15] Wu J, Shen X, Zhu W, Liu L. Mesh saliency with global rarity. Graph Models 2013;75(5):255–64.

[16] Zernike vF. Beugungstheorie des schneidenver-fahrens und seiner verbesserten form, der phasenkontrastmethode. Physica 1934;1(7):689–704. http://dx.doi.org/10.1016/S0031-8914(34)80259-5, URL: http://www.sciencedirect.com/science/article/pii/S0031891434802595.

[17] Lakshminarayanan V, Fleck A. Zernike polynomials: a guide. J Modern Opt 2011;58(18):1678. http://dx.doi.org/10.1080/09500340.2011.633763, arXiv:https://doi.org/10.1080/09500340.2011.633763.

[18] Page DL, Koschan AF, Sukumar SR, Roui-Abidi B, Abidi MA. Shape analysis algorithm based on information theory. In: Proceedings 2003 international conference on image processing (Cat. No.03CH37429), Vol. 1. 2003, p. 1–229.

[19] Feng X, Wan W, Xu RYD, Perry S, Zhu S, Liu Z. A new mesh visual quality metric using saliency weighting-based pooling strategy. Graph Models 2018;99:1–12.

[20] Lavoué G, Cordier F, Seo H, Larabi M-C. Visual attention for rendered 3D shapes. Comput Graph Forum 2018;37(2):191–203. http://dx.doi.org/10.1111/cgf.13353, URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13353.

[21] Wang X, Koch S, Holmqvist K, Alexa M. Tracking the Gaze on objects in 3D: How do people really look at the Bunny? ACM Trans Graph 2018;37(6). http://dx.doi.org/10.1145/3272127.3275094.

[22] Wang X, Lindlbauer D, Lessig C, Maertens M, Alexa M. Measuring the visual salience of 3D printed objects. IEEE Comput Graph Appl 2016;36(4):46–55. http://dx.doi.org/10.1109/MCG.2016.47.

[23] Hu S, Liang X, Shum HP, Li FW, Aslam N. Sparse metric-based mesh saliency. Neurocomputing 2020;400:11–23.

[24] Song R, Liu Y, Rosin P. Mesh saliency via weakly supervised classification-for-saliency CNN. IEEE Trans Vis Comput Graphics 2019. http://dx.doi.org/10.1109/TVCG.2019.2928794.

[25] Song R, Zhang W, Zhao Y, Liu Y, Rosin PL. Mesh saliency: An independent perceptual measure or a derivative of image saliency? In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR). 2021, p. 8853–62.

[26] Chen X, Saparov A, Pang B, Funkhouser T. Schelling points on 3D surface meshes. ACM Trans Graph 2012;31(4). http://dx.doi.org/10.1145/2185520.2185525.

[27] Nousias S, Arvanitis G, Lalos AS, Moustakas K. Mesh saliency detection using convolutional neural networks. In: 2020 IEEE international conference on multimedia and expo. ICME, IEEE; 2020, p. 1–6.

[28] Taubin G. Estimating the tensor of curvature of a surface from a polyhedral approximation. In: Proceedings of IEEE international conference on computer vision. IEEE; 1995, p. 902–7.

[29] DeCarlo D, Finkelstein A, Rusinkiewicz S, Santella A. Suggestive contours for conveying shape. In: ACM SIGGRAPH 2003 papers. SIGGRAPH '03, New York, NY, USA: Association for Computing Machinery; 2003, p. 848–55. http://dx.doi.org/10.1145/1201775.882354.

[30] Corsini M, Cignoni P, Scopigno R. Efficient and flexible sampling with blue noise properties of triangular meshes. IEEE Trans Vis Comput Graphics 2012;18(6):914–24. http://dx.doi.org/10.1109/TVCG.2012.34.

[31] Lagae A, Dutré P. A comparison of methods for generating Poisson disk distributions. In: Computer graphics forum, Vol. 27. Wiley Online Library; 2008, p. 114–29.

[32] Stanford 3D scanning repository. Stanford Computer Graphics Laboratory, URL: http://graphics.stanford.edu/data/3Dscanrep/gamma-corrected/.

[33] Giorgi D, Biasotti S, Paraboschi L. Shape retrieval contest 2007: Watertight models track. 2007.

[34] Zremesher by zbrush. URL: https://pixologic.com.

[35] Hu S, Shum HP, Aslam N, Li FW, Liang X. A unified deep metric representation for mesh saliency detection and non-rigid shape matching. IEEE Trans Multimed 2019;22(9):2278–92.

[36] Nguyen CV, Izadi S, Lovell D. Modeling kinect sensor noise for improved 3D reconstruction and tracking. In: 2012 second international conference on 3D imaging, modeling, processing, visualization transmission. 2012, p. 524–30. http://dx.doi.org/10.1109/3DIMPVT.2012.84.

[37] Mallick T, Das PP, Majumdar AK. Characterizations of noise in kinect depth images: A review. IEEE Sens J 2014;14(6):1731–40. http://dx.doi.org/10.1109/JSEN.2014.2309987.