UNIVERSITY of York

This is a repository copy of *Symmetry-Aware Mesh Segmentation into Uniform Overlapping Patches*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/105622/

Version: Accepted Version

Article:
Dessein, Arnaud Fabien, Smith, William Alfred Peter orcid.org/0000-0002-6047-0413, Wilson, Richard Charles orcid.org/0000-0001-7265-3033 et al. (1 more author) (2017) Symmetry-Aware Mesh Segmentation into Uniform Overlapping Patches. Computer graphics forum. pp. 95-107. ISSN 0167-7055

https://doi.org/10.1111/cgf.12997

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Volume 0 (1981), Number 0 pp. 1–12 COMPUTER GRAPHICS *forum*

Symmetry-Aware Mesh Segmentation into Uniform Overlapping Patches

A. Dessein[1], W. A. P. Smith[2], R. C. Wilson[2], and E. R. Hancock[2]

[1]IMB / LaBRI, Université de Bordeaux, France [2]Department of Computer Science, University of York, UK

Abstract

We present intrinsic methods to address the fundamental problem of segmenting a mesh into a specified number of patches with a uniform size and a controllable overlap. Although never addressed in the literature, such a segmentation is useful for a wide range of processing operations where patches represent local regions and overlaps regularise solutions in neighbour patches. Further, we propose a symmetry-aware distance measure and symmetric modification to furthest-point sampling, so that our methods can operate on semantically symmetric meshes. We introduce quantitative measures of patch size uniformity and symmetry, and show that our segmentation outperforms state-of-the-art alternatives in experiments on a well-known dataset. We also use our segmentation in illustrative applications to texture stitching and synthesis where we improve results over state-of-the-art approaches.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Mesh structures have become a ubiquitous tool in computer graphics for 3D modelling of real-world and virtual objects. In this realm, a profusion of mesh processing techniques have arisen, being either designed according to inherent mesh specificities (e.g. rendering, remeshing, reconstruction) or borrowed from image processing ideas (e.g. filtering, denoising, segmentation). Adapting techniques from image processing, however, is not always straightforward and might require a conceptual rethinking of 2D notions for a meaningful extension to 3D models. In the context of segmentation, [SPDF14] generalise the idea of superpixels to superfacets. We follow here a complementary approach, and transfer the fundamental problem of segmenting an image into uniform overlapping patches to the domain of meshes.

1.1. Context and Motivations

In contrast to superpixels, which trade off the size uniformity against the content homogeneity, a uniform overlapping 2D patch structure is independent of the image content. Segmenting an image into such a structure is easy and can be specified by the height and width of rectangular (often square) patches, and the vertical and horizontal offsets between them. Methods that use 2D uniform overlapping patches are numerous and come from a wide range of problem domains including super-resolution [FJP02], sparse coding [MES08], denoising [DFKE07], restoration [ZW11], texturing [LLX*01], quilting [EF01], editing [CAF10], hybridisation [LW11], synthesis [WLKT09], face hallucination [LLT05] and generation [MPK09]. Some 3D mesh texturing schemes also exploit 2D uniform overlapping patches in the plane after surface flattening [PFH00, ZHW*06, WSBY06].

Given the ubiquity of 2D uniform overlapping patches in image processing, it is surprising that there are no existing methods for segmenting a 3D mesh similarly. Any operation that we would like to apply over the mesh surface to process texture, geometry, or any given function defined on the mesh, may benefit from such a segmentation. We could notably extend the above 2D methods to operate on 3D meshes, and process 3D meshes intrinsically without an arbitrary flattening, other extrinsic operations, and their inherent distortions.

Such a mesh segmentation, however, is not trivial to compute because one cannot rely on the simple translation of a given base shape. This is a result of emphasising the intrinsic mesh geometry compared to flat images. In addition, many objects and associated meshes possess symmetries as inherent components of their geometry. While most mesh pro-

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

cessing methods just ignore such symmetries for simplicity, it nonetheless seems natural to expect output solutions to be equivalent for symmetric parts. All in all, the desired segmentation should satisfy a certain number of properties:

- the patches be uniform in size, so that they encompass equal amounts of local data and exert a similar influence over global solutions;
- the number of patches or patch size, and hence the segmentation granularity, can be controlled;
- the degree of overlap, and hence the influence of neighbouring constraints, can be controlled;
- the methods be intrinsic, so that results are invariant under pose changes involving rigid deformations and robust against non-rigid ones;
- the symmetries can be preserved, so that symmetric regions of the mesh lie in symmetric patches.

1.2. Related Work

Mesh segmentation algorithms partition vertices, edges or faces to optimise a chosen objective. The survey of [Sha08] distinguishes between two types of objectives: part-type to produce semantically meaningful regions (typically, volumetric components such as arms, legs, body and head for a human body), and surface-type to create locally consistent patches under some criteria (often, disc-homeomorphic patches for the purposes of flattening). Segmentation is also related to sampling where samples determine patch centres.

Centroidal Voronoi tessellation (CVT) [CSAD04] use Lloyd's *k*-means clustering in Euclidean space to form a CVT (i.e. a Voronoi tessellation where centres are centroids) for surface-type segmentation. [YBZW14] locally enhance such an extrinsic CVT to ensure that Voronoi cells form single connected regions. [PC04] propose an intrinsic CVT based on geodesic distances, but centroids are not always well-defined, and clusters are not guaranteed to converge due to the manifold curvature. These algorithms provide a non-uniform segmentation. [VC04] approximate a CVT for uniform segmentation, but in the extrinsic setting. In addition, all these methods are not aware of any symmetry.

Poisson-disc sampling (PDS) [LLLF08] address PDS (i.e. uniform drawing of disjoint discs with a specified radius) for surfaces using an extrinsic algorithm. [FZ09] advocate the use of geodesic distances, but introduce an extrinsic relaxation step and make topological assumptions on the mesh. [CJW*09] develop a fully intrinsic PDS algorithm based on dart throwing. [CCS12] enhance PDS with drawing constraints such as importance sampling instead of uniform drawing. [YW13] show useful results for processing gaps in Poisson disc sets. Inherent to PDS, sampling is stochastic and disc centres can be spaced arbitrarily as long as they do not violate the minimum distance constraint (uniform drawing does not mean uniform spacing). Hence, it is difficult to specify the number of samples while ensuring an even spacing. Lastly, there is no obvious way to preserve symmetry.

Furthest-point sampling (FPS) [MD03] use FPS (i.e. iterative selection of the vertex that is furthest from previous ones) to sample surfaces evenly based on geodesic distances, and obtain a surface-type segmentation via the underlying Voronoi tessellation. [PC06] improve the fast marching scheme used to compute geodesic distances. FPS can be seen as a highly biased PDS where all samples are distant from a minimum distance but sampling gets deterministic after the first point is drawn. This allows selecting the number of samples while favouring an even spacing and thus uniform patches. Again, it is yet non-trivial to preserve symmetry.

Symmetry-aware segmentation [PSG*06] detect approximate planar symmetries and segment the mesh accordingly. [SKS06] further decompose the mesh into a hierarchy of symmetric parts. [MGP06] also find rotation, translation and scaling in addition to reflexion. [PMW*08] detect more general structural regularity under the same set of transformations. [LCDFh10] define a symmetry factored embedding (SFE) to detect complex symmetries and segment the mesh by extrinsic clustering in the SFE space. [XZJ*12] extend SFE for detection of multi-scale partial symmetries. These algorithms lead to part-type, non-uniform segmentations. They also do not necessarily seek a symmetric segmentation, but rather use symmetry as a cue for segmentation. [PGR07] introduce symmetry constraints in a CVT, but work in the extrinsic setting and build a non-uniform segmentation. In all approaches, the symmetry measure quantifies how symmetric different parts of the mesh are under a prescribed group of transformations. We are not aware of any work defining a symmetry-aware distance over the mesh vertices themselves.

Overlapping segmentation [TCY09] use an extrinsic patch dilation to build up overlapping segments that are required to be disc-homeomorphic, which prevents segmentation of meshes with a boundary and constrains the number of patches for meshes with a positive genus. [HGLT09] grow patches using a distance criterion based on the boundary length between neighbours, which allows an intrinsic dilation when using the geodesic distance, and puts no constraints on the mesh or patches. Both methods, however, produce non-uniform patches and do not preserve symmetries. Moreover, they do not provide intuitive controls on overlap.

Discussion None of these existing works addresses either a uniform overlapping, a symmetry-aware overlapping, or a symmetry-aware uniform segmentation. Many methods also neglect the intrinsic mesh geometry, or suffer from issues such as stochastic indeterminacy, topological constraints on the mesh or patches, and lack of algorithmic guarantees.

1.3. Overview and Contributions

We propose intrinsic methods for symmetry-aware segmentation of mesh vertices into a user-specified number of patches that are of uniform size and that overlap by a desired proportion. To do so, we revisit FPS on manifolds.

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

Although it allows for an intrinsic mesh segmentation into uniform patches, classical FPS fails both to produce overlapping patches and to preserve mesh symmetries.

A naive solution is to compute classical FPS [PC06] on a single symmetry orbit and then copy patches. However, results would not only depend on the arbitrary choice of the reference orbit, but also feature uniformity issues at borders between orbits. Another solution is to run FPS based on distances in the state-of-the-art SFE space [LCDFh10]. Nonetheless, this would hinder both uniformity and symmetry. Lastly, even with symmetric, uniform input segments, the existing patch growing [HGLT09] would degrade uniformity and break symmetry of the overlapping output patches. These issues are verified and discussed in our experiments.

We tackle these issues with three technical contributions:

1. We define a symmetry-aware distance over the mesh vertices which accounts for all symmetry orbits (Sect. 2)
2. We propose a symmetric FPS that fixes uniformity issues at borders between orbits and ensures symmetry (Sect. 3).
3. We develop a patch growing scheme that builds up uniform overlaps while preserving symmetry (Sect. 4).

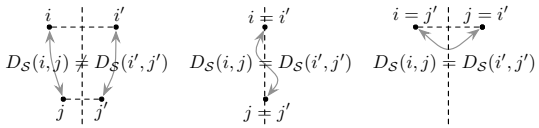
We also introduce quantitative measures of patch size uniformity and symmetry, and present experiments on a well-known dataset to show that only the coupling of all our three contributions together can provide a symmetry-aware, uniform and overlapping segmentation compared to the above-mentioned alternatives (Sect. 5). We finally motivate the utility of our segmentation on applications to texture stitching and texture synthesis problems, where we improve results over state-of-the-art methods (Sect. 6).

Our pipeline has a number of advantages. It is fully automatic, though the user can interact by specifying patch centres as seeds to control sampling and remove any stochastic indeterminacy. There are also no issues of algorithmic guarantees nor topological constraints on the mesh or patches.

1.4. Assumptions and Notations

We consider a triangular mesh $\mathcal{M} = (\mathcal{K}, \mathbf{S})$. The connectivity is given by the simplicial complex $\mathcal{K}$, whose elements can be vertices $\{i\}$, edges $\{i, j\}$, or faces $\{i, j, k\}$, with indices $i, j, k \in [1\,..\,N]$, where N is the number of vertices. The shape is given by the matrix $\mathbf{S} \in \mathbb{R}^{3\times N}$ which contains the 3D coordinates $\mathbf{s}_i \in \mathbb{R}^3$ of the respective vertices. We assume that the mesh surface $\mathcal{S}$ forms a 2D manifold with geodesic distance map $D_{\mathcal{S}}$. We often write a vertex $\{i\}$ simply as i and conflate i with $\mathbf{s}_i$ when the intended meaning is clear.

The mesh may also exhibit symmetries. We do not solve here the problem of finding these symmetries, but assume that they have been modelled directly with the mesh, or at least that they have been detected and that a remeshing has been applied accordingly. For simplicity, we focus on bilateral symmetry, though other symmetries can be handled as explained in our conclusions. This does not necessarily mean that the shape $\mathbf{S}$ is geometrically symmetric from extrinsic or intrinsic viewpoints, just that each vertex i has a semantically symmetric partner i'. We extend the symmetry operator to sets and simplicial complexes by applying it element-wise.



(a) No symmetry. (b) Self-symmetry. (c) Symmetry.

Figure 1: *Geodesic distance on a symmetric mesh with no exact extrinsic nor intrinsic symmetry. In general, the geodesic distance $D_{\mathcal{S}}$ does not preserve symmetry for arbitrary vertices i and j, except from self-symmetric vertices $i = i'$ and $j = j'$, or symmetric vertices $i = j'$ and $j = i'$.*

We define a patch $\mathcal{P}$ as a subset of vertices $i \in \mathcal{K}$. Two patches $\mathcal{P}_m$ and $\mathcal{P}_n$ are neighbours if they contain vertices $i \in \mathcal{P}_m$ and $j \in \mathcal{P}_n$ that are connected by an edge $\{i, j\} \in \mathcal{K}$. A standard segmentation covers $\mathcal{M}$ with M patches $\mathcal{P}_1, \ldots, \mathcal{P}_M \subset \mathcal{K}$, such that all vertices belong to exactly one patch. A segmentation thus defines a dual graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = [1\,..\,M]$, and $(m, n) \in \mathcal{E}$ if $\mathcal{P}_m$ and $\mathcal{P}_n$ are neighbours. We here loosen the hard partition constraint to allow vertices to be shared by adjacent symmetric patches along the symmetry line, or by overlapping patches $\mathcal{Q}_1, \ldots, \mathcal{Q}_M$ once the patches $\mathcal{P}_1 \subset \mathcal{Q}_1, \ldots, \mathcal{P}_M \subset \mathcal{Q}_M$ are grown.

2. Symmetry-Aware Distance

The geodesic distance $D_{\mathcal{S}}$ is not aware of symmetry as soon as the shape $\mathbf{S}$ is not perfectly symmetric (Fig. 1). Thus, $D_{\mathcal{S}}$ does not in general preserve the distances between two vertices and between their symmetric partners. Hence, any operation based on $D_{\mathcal{S}}$, such as point sampling or patch growing, will hinder symmetry. To tackle this, we define a symmetry-aware distance that preserves the mesh symmetries, if any.

2.1. Formulation

We seek to construct a symmetry-aware distance $\widetilde{D}_{\mathcal{S}}$ over the surface of a symmetric mesh, that naturally extends the geodesic distance $D_{\mathcal{S}}$ for non-symmetric meshes. With this aim in mind, it is relevant to consider the points by pairs, and hence to work on the Cartesian square $\mathcal{S} \times \mathcal{S}$ of the manifold.

We notice, however, that the symmetry operator being defined on the mesh vertices $\mathbf{s}_i \in \mathcal{S}$ only, we need a means to extend it to any point $\mathbf{s} \in \mathcal{S}$ on the whole mesh surface. This can be done by extending the symmetry inside symmetric faces using the barycentric coordinates within the triangles. We assume that such an extension has been chosen, and show hereafter that the arbitrary choice of this extension does not actually alter the symmetry-aware distance itself.

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

We now define the symmetry-aware distance $\widetilde{D}_{\mathcal{S}}$ as the geodesic distance $D_{\mathcal{S}\times\mathcal{S}}$ computed between pairs of symmetric points on the Cartesian square manifold:

$$\widetilde{D}_{\mathcal{S}}(\mathbf{r},\mathbf{s}) = D_{\mathcal{S}\times\mathcal{S}}((\mathbf{r},\mathbf{r}'),(\mathbf{s},\mathbf{s}')) \ , \tag{1}$$

where $\mathcal{S}\times\mathcal{S}$ is endowed with the canonical structure of product manifold induced by that of $\mathcal{S}$. For a general product manifold, the canonical structure is obtained by orthogonal copies of the respective manifolds (i.e. by direct product of their charts and direct sum of their tangent planes). As a result, the induced metric is given by the element-wise sum of the respective metrics, and hence geodesic distances are computed according to the Pythagorean theorem.

Thus, the symmetry-aware distance between two vertices is the quadratic mean (up to $1/\sqrt{2}$) of geodesic distances between those vertices and between their symmetric partners:

$$\widetilde{D}_{\mathcal{S}}(i,j) = \sqrt{D_{\mathcal{S}}(i,j)^2 + D_{\mathcal{S}}(i',j')^2} \ . \tag{2}$$

This expression is clearly independent of the choice of an extension for the symmetry operator on $\mathcal{S}$.

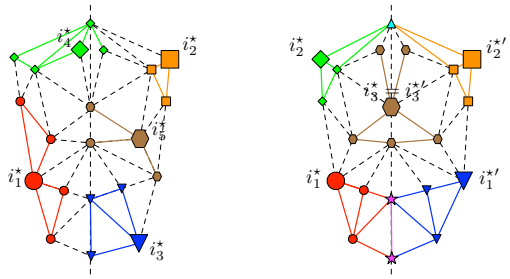
Interestingly, the symmetry-aware distance $\widetilde{D}_{\mathcal{S}}(i,i')$ between two symmetric vertices reduces (up to $\sqrt{2}$) to their geodesic distance $D_{\mathcal{S}}(i,i')$. The same holds for two self-symmetric vertices, so that we can use the convention that all vertices are self-symmetric for a non-symmetric mesh to make the symmetry-aware distance $\widetilde{D}_{\mathcal{S}}$ equivalent to the original geodesic distance $D_{\mathcal{S}}$. Hence, our framework naturally extends the non-symmetric case to handle symmetry.

2.2. Implementation

The symmetry-aware distance $\widetilde{D}_{\mathcal{S}}$ amounts to calculating two geodesic distances $D_{\mathcal{S}}$. In practice, this can be done with numerical methods such as fast marching. We use an existing and efficient MATLAB implementation [PC06]. This algorithm is designed for triangular meshes, and is based on directional derivatives with unfolding of obtuse triangles for numerical stability, as well as optimisation of the sorting-based priority queue with a Fibonacci heap.

With this algorithm, we can compute the symmetry-aware distance map to a vertex i, by calculating either one geodesic distance map if vertex i is self-symmetric, or two geodesic distance maps otherwise (although we need not always calculate the full maps when pruning vertices by front confinement or early abandon). Additionally, we can compute distance maps to an arbitrary set of vertices, instead of a single vertex, by propagation from this set as a starting front.

For practical considerations, we also use a slight variant of the symmetry-aware distance. When considering vertices that lie in a subset of interest, we confine front propagation within the induced surfaces $\mathcal{R}\subset\mathcal{S}$ and $\mathcal{R}'\subset\mathcal{S}$ for the first and second geodesic distance maps, respectively. This allows a symmetric sampling by separating propagations on the respective sides of the mesh. Not only does it provide a more efficient scheme, but it also prevents unwanted effects from occurring along the symmetry line by dimensional collapse from the product manifold to the mesh. This further proves useful when restricting propagation to grow a given patch within each of its respective neighbours only.



(a) Asymmetric sampling. (b) Symmetric sampling.

Figure 2: *Uniform mesh sampling. In this example, we sample 5 vertices on a symmetric mesh. We seed the asymmetric version with vertex $i_1^\star$, and select next samples $i_2^\star, i_3^\star, i_4^\star, i_5^\star$ successively by maximising the minimum geodesic distance to previous ones. In our symmetric version, we take symmetric partners $i_1^\star, i_1^{\star\prime}$ for seeding, and systematically sample vertices $i_2^\star, i_2^{\star\prime}, i_3^\star, i_3^{\star\prime}$ by pairs using the symmetry-aware distance and accounting for pairwise distances. The obtained segmentation preserves the original mesh symmetries with the symmetric version in contrast to the asymmetric one.*

3. Symmetry-Aware Furthest-Point Sampling

We here enhance classical FPS [PC06] with a symmetric version (Fig. 2). First, we use our symmetry-aware distance $\widetilde{D}_{\mathcal{S}}$ rather than geodesic distance $D_{\mathcal{S}}$. Second, we systematically sample vertices by symmetric pairs. Third, when sampling a new pair of vertices $i^\star \neq i^{\star\prime}$, the pairwise distance $\widetilde{D}_{\mathcal{S}}(i^\star, i^{\star\prime})$ is also considered to avoid uniformity issues along the symmetry line, by favouring self-symmetric vertices rather than proper symmetric vertices that are close.

3.1. Algorithm

We consider one side of the mesh without lack of generality, recalling that as soon as vertex $i^\star$ is sampled, its symmetric partner $i^{\star\prime}$ is sampled too. Since our symmetry-aware distance $\widetilde{D}_{\mathcal{S}}$ accounts for both sides evenly, subsequent developments do not depend on the arbitrary choice of a side. We denote by $\widetilde{\mathcal{K}}$ the connectivity induced by $\mathcal{K}$ on that side.

We grow the sample set iteratively by adding a new sample one at a time. We denote by $\widetilde{\mathcal{I}}_l = \{i_1^\star, \ldots, i_l^\star\} \subset \widetilde{\mathcal{K}}$ the set of first l selected samples. We also define $\widetilde{D}_l$ as the symmetry-aware distance map to $\widetilde{\mathcal{I}}_l$:

$$\widetilde{D}_l(i) = \min_{i^\star\in\widetilde{\mathcal{I}}_l} \widetilde{D}_{\mathcal{S}}(i,i^\star) \ . \tag{3}$$

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

To select the next sample $i^\star_{l+1}$, we look for the vertex which is the furthest from the current samples, and from its symmetric partner if proper. To formalise this, we pose the convention that $\widetilde{D}_0(i)$ is the symmetry-aware distance from i to i', except from self-symmetric vertices where it is infinite:

$$\widetilde{D}_0(i) = \begin{cases} \widetilde{D}_{\mathcal{S}}(i,i') & \text{if } i \neq i'; \\ +\infty & \text{otherwise.} \end{cases} \tag{4}$$

The next sample $i^\star_{l+1}$ is then chosen as follows:

$$i^\star_{l+1} = \underset{i \in \widetilde{\mathcal{K}}}{\arg\max}\ \min\left\{\widetilde{D}_0(i), \widetilde{D}_l(i)\right\} \ . \tag{5}$$

For the next iteration, the new distance map $\widetilde{D}_{l+1}$ can simply be updated as the minimum between the previous distance map $\widetilde{D}_l$ and that to the new sample $i^\star_{l+1}$:

$$\widetilde{D}_{l+1}(i) = \min\left\{\widetilde{D}_l(i), \widetilde{D}_{\mathcal{S}}(i, i^\star_{l+1})\right\} \ . \tag{6}$$

We repeat the process until M' pairs of symmetric vertices are sampled, where M' is such that we obtain M distinct samples in total (up to a tolerance of one sample more because of selecting one or two distinct points at each iteration).

In the end, patches $\mathcal{P}_1, \ldots, \mathcal{P}_{M'}$ are constructed via the Voronoi tessellation based on symmetry-aware distances:

$$\mathcal{P}_m = \left\{ i \in \widetilde{\mathcal{K}} \colon \widetilde{D}_{\mathcal{S}}(i, i^\star_m) = \min_{i^\star \in \widetilde{I}_{M'}} \widetilde{D}_{\mathcal{S}}(i, i^\star) \right\} \ . \tag{7}$$

After copying patches across the second side, we may obtain self-symmetric patches that span both sides of the mesh. Other patches might also share some self-symmetric vertices as soon as they are adjacent along the symmetry line. We finally note that our symmetric FPS is a natural extension of classical FPS for non-symmetric meshes, where all vertices are considered as self-symmetric, distances $D_{\mathcal{S}}$ and $\widetilde{D}_{\mathcal{S}}$ coincide and no pairwise distances $\widetilde{D}_0$ alter sampling.

3.2. Implementation

Typically, classical FPS is seeded by sampling randomly the first vertex $i^\star_1$. As a refinement, one can replace $i^\star_1$ with its furthest vertex. When doing so for symmetric FPS we must again include pairwise distances in the distance criterion. Optionally, we also propose to initialise sampling with a set $\{i^\star_1, \ldots, i^\star_L\}$ of fiducial keypoints that represent semantically meaningful regions and can be selected manually or automatically. A by-product advantage is to make sampling deterministic since seeding with at least one keypoint removes the stochastic indeterminacy due to random initialisation.

Although unlikely in practice, there may exist multiple vertices that reach the maximum distance to previous samples at a given iteration. If this happens, we randomly select one of them. It also allows using the algorithm on disconnected meshes, where some maximum distances are infinite until a vertex is sampled in each connected component.

For efficiency, we also assume that each vertex is assigned to a single patch (except from shared vertices on the symmetry line). Whenever the minimum distance to centres is reached for several patches, which is again quite unlikely, we keep the patch whose centre was sampled first. A single vector is thus required along iterations to store and update distances $\widetilde{D}_l$. The respective closest samples, and hence the associated Voronoi tessellation, can also be updated iteratively with a single vector of sample indices.

To update $\widetilde{D}_{l+1}$, we need to compute the distance map to the new sample $i^\star_{l+1}$. This can be done efficiently by starting respective fronts from $i^\star_{l+1}$ and $i^\star_{l+1}{}'$ via fast marching. For initialisation, we need to compute the distances $\widetilde{D}_1, \ldots, \widetilde{D}_L$ from each seed keypoint. This is also done via fast marching from the seeds $i^\star_1, \ldots, i^\star_L$, and their symmetric partners $i^{\star\prime}_1, \ldots, i^{\star\prime}_L$. We thus propagate two fronts for each of the M' sample pairs, requiring $O(MN \log N)$ operations in the worst case. We can improve this by confining the fronts for $\widetilde{D}_l$ before step l to the set $\{i \in \widetilde{\mathcal{K}} \colon \widetilde{D}_{\mathcal{S}}(i, i^\star_{l-1}) \leq \widetilde{D}_{l-1}(i)\}$.

More precisely, the first front from $i^\star_{l-1}$ is restricted to vertices $i \in \widetilde{K}$ such that $D_{\mathcal{S}}(i, i^\star_{l-1}) \leq \widetilde{D}_{l-1}(i)$. The second front from $i^\star_{l-1}{}'$ is then restricted to vertices i' such that $D_{\mathcal{S}}(i', i^\star_{l-1}{}')^2 \leq \widetilde{D}_{l-1}(i)^2 - D_{\mathcal{S}}(i, i^\star_{l-1})^2$. This makes the computation more efficient. Indeed, assuming that vertices are evenly spaced, which approximately holds for a sufficiently fine meshing, we expect the new patch to contain about $N/2l$ vertices. Thus, the complexity for confined propagation is roughly bounded by $\sum_{l=1}^{M'} (N/l) \log(N/l)$, implying an asymptotic complexity of $O(N \log N \log M)$.

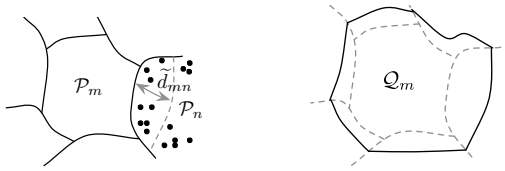
Lastly, we need to initialise pairwise distances $\widetilde{D}_0$ between symmetric vertices. This can be done in batch before segmentation. It leads to propagating about $N/2$ fronts (slightly less due to self-symmetric vertices) for a pessimistic complexity of $O(N^2 \log N)$. Although this computation is the bottleneck of our whole segmentation pipeline, there are several heuristics that result in a large speed-up.

3.3. Heuristics

To begin with, we stop the respective fronts as soon as the symmetric partners have been reached. Even if effective in practice, this heuristic improvement is hard to quantify.

Another improvement is to calculate pairwise distances greedily along iterations. Before sampling at iteration l, we sort the vertices in decreasing order according to the following distance criterion: $\widetilde{D}_l(i)$ if $\widetilde{D}_0(i)$ has not been computed yet, or minimum between $\widetilde{D}_l(i)$ and $\widetilde{D}_0(i)$ otherwise. We then compute $\widetilde{D}_0(i)$ and stop when reaching a vertex for which it has already been computed, or for which the minimum between $\widetilde{D}_l(i)$ and $\widetilde{D}_0(i)$ is greater than the distance criterion of the next vertex. We know that all subsequent vertices are not candidates for symmetric FPS at iteration l, and that we need not compute their pairwise distances now if not

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.



(a) Before projection. (b) After projection.

Figure 3: *Patch agglomeration. A patch $\mathcal{P}_m$ is grown by projecting vertices from surrounding patches $\mathcal{P}_n$. The vertices that are aggregated to the grown patch $\mathcal{Q}_m$ are such that their distance to $\mathcal{P}_m$ is less than a controlled threshold $\widetilde{d}_{mn}$.*

already done. We finally sample among the first vertices, for which all required quantities are now available.

A further optimisation is the early abandon of required distances $\widetilde{D}_0(i)$ during front propagation. Because the front distance to i increases monotonically along fast marching iterations until reaching i', we stop propagation as soon as the front distance exceeds $\widetilde{D}_{l-1}(i)$.

Lastly, we also test early abandon of $\widetilde{D}_0(i)$ by first using an inexpensive lower bound. If this bound is greater than $\widetilde{D}_{l-1}(i)$, we need not compute $\widetilde{D}_0(i)$. Moreover, we will not need it for subsequent iterations $l' > l$ either. This reveals effective for proper symmetric vertices $i \neq i'$ that are far from the symmetry line. A convenient bound is the 3D Euclidean distance $\|\mathbf{s}_i - \mathbf{s}_{i'}\|_2 \leq D_{\mathcal{S}}(i,i') = \widetilde{D}_{\mathcal{S}}(i,i')/\sqrt{2}$, which can be computed in $O(1)$ instead of the worst-case $O(N \log N)$.

4. Symmetry-Aware Uniform Patch Growing

To grow patches and make them overlap, we proceed by agglomeration (Fig. 3). The existing patch growing [HGLT09] is not designed so that the grown patches will have uniform areas nor be symmetric. Moreover, it fails to provide natural controls on overlap. Symmetry can be handled easily by plugging in our symmetry-aware distance for projection of surrounding vertices. We grow $\mathcal{P}_m$ by computing distances to the vertices of neighbour patches $\mathcal{P}_n$. The grown patch $\mathcal{Q}_m$ is then obtained by adding vertices whose distance is less than a threshold $\widetilde{d}_{mn}$. We choose different thresholds $\widetilde{d}_{mn}$ for local adaptation to the approximate uniformity. These thresholds are derived from a single global parameter σ that controls overlap intuitively over the whole mesh.

4.1. Algorithm

To grow a given patch $\mathcal{P}_m$, we consider separately each of its neighbours $\mathcal{P}_n$ with $(m,n) \in \mathcal{E}$, and define thresholds as:

$$\widetilde{d}_{mn} = \sigma \times \widetilde{D}_{\mathcal{S}}(i_m^\star, i_n^\star) \ , \tag{8}$$

where the overlap ratio $\sigma \geq 0$ is set by the user. The threshold $\widetilde{d}_{mn}$ is thus proportional to the distance between the two patch centres. Hence, the threshold $\widetilde{d}_{nm}$ of $\mathcal{P}_n$ into $\mathcal{P}_m$ will equal the threshold $\widetilde{d}_{mn}$ of $\mathcal{P}_m$ into $\mathcal{P}_n$, so that both patches interpenetrate in each other within the same proportion. The overlap $\mathcal{O}_{mn}$ of $\mathcal{P}_m$ onto $\mathcal{P}_n$ is then given by:

$$\mathcal{O}_{mn} = \left\{ i \in \mathcal{P}_n \colon \min_{j \in P_m} \widetilde{D}_{\mathcal{S}}(i,j) \leq \widetilde{d}_{mn} \right\} \ . \tag{9}$$

The parameter σ controls the overlap size intuitively over the mesh. For $\sigma = 0$, no growing occurs at all. When $\sigma = 0.5$, the grown patch $\mathcal{Q}_m$ reaches the centres $i_n^\star$ of its neighbours $\mathcal{P}_n$, independently of their distances $\widetilde{D}_{\mathcal{S}}(i_m^\star, i_n^\star)$ to the reference patch centre $i_m^\star$. Indeed, the patch boundaries are exactly halfway from the neighbour patch centres. Assuming the vertices are evenly spaced over the mesh, we would thus expect about half of the vertices in $\mathcal{P}_n$ to be included in $\mathcal{O}_{mn}$. For σ increasing up to 1 or greater, the overlap $\mathcal{O}_{mn}$ spreads after $i_n^\star$ and ends up spanning the whole neighbour patch $\mathcal{P}_n$.

A grown patch $\mathcal{Q}_m$ is eventually constructed by concatenation of $\mathcal{P}_m$ and of the respective overlaps:

$$\mathcal{Q}_m = \mathcal{P}_m \cup \bigcup_{n|(m,n)\in\mathcal{E}} \mathcal{O}_{mn} \ . \tag{10}$$

By symmetry, we only need to grow half (slightly more because of self-symmetric patches) of the patches $\mathcal{P}_1, \ldots, \mathcal{P}_{M'}$ into $\mathcal{Q}_1, \ldots, \mathcal{Q}_{M'}$. We finally note that the proposed growing scheme again generalises naturally the non-symmetric case, where projections according to the geodesic distance $D_{\mathcal{S}}$ instead of the symmetry-aware distance $\widetilde{D}_{\mathcal{S}}$ are used.

4.2. Implementation

On the one hand, computing a given distance threshold $\widetilde{d}_{mn}$ requires propagating one or two fronts. If both patches are self-symmetric, we just need to start a single front from the reference centre $i_m^\star = i_m^{\star\prime}$ to the neighbouring centre $i_n^\star = i_n^{\star\prime}$, with restriction to $\mathcal{P}_m \cup \mathcal{P}_n$. Similarly, if both patches are symmetric, we start a single front from $i_n^\star = i_m^{\star\prime}$ to $i_n^\star = i_m^{\star\prime}$. Otherwise, we need to start two fronts: one from $i_m^\star$ to $i_n^\star$, and one from $i_m^{\star\prime}$ to $i_n^{\star\prime}$, where the former is restricted to $\mathcal{P}_m \cup \mathcal{P}_n$, while the latter is restricted to $\mathcal{P}'_m \cup \mathcal{P}'_n$. Since thresholds $\widetilde{d}_{mn} = \widetilde{d}_{nm}$ are symmetric, we actually only need to compute half of them. On the other hand, growing a reference patch $\mathcal{P}_m$ within a given neighbour $\mathcal{P}_n$ systematically requires two front propagations starting from the whole patches $\mathcal{P}_m$ and $\mathcal{P}'_m$, with restriction to $\mathcal{P}_n$ and $\mathcal{P}'_n$, respectively.

The expected number of patch neighbours is asymptotically bounded (see supplementary material for a proof). Hence, we need to propagate a bounded number of fronts for each patch. The overall pessimistic complexity is thus in $O(MN \log N)$, though we have a lower practical complexity. Since each patch contains about N/M vertices, for a sufficiently fine meshing, growing a patch requires $O((N/M)\log(N/M))$ computations. We thus expect an overall complexity of $O(N \log(N/M))$.

To go further, we can also confine the fronts, with respect to local thresholds $\widetilde{d}_{mn}$ in neighbouring patches, in the respective sets $\{i \in \mathcal{P}_n \colon \widetilde{D}_{\mathcal{S}}(i, \mathcal{P}_m) \leq \widetilde{d}_{mn}\}$. More precisely,

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

the first front from $\mathcal{P}_m$ can be restricted to vertices $i \in \mathcal{P}_n$ such that $D_{\mathcal{S}}(i,\mathcal{P}_m) \leq \widetilde{d}_{mn}$. The second front from $\mathcal{P}'_m$ can further be restricted to vertices i' such that $D_{\mathcal{S}}(i',\mathcal{P}'_m) \leq \widetilde{d}^2_{mn} - D_{\mathcal{S}}(i,\mathcal{P}_m)^2$. As a result, the number of required operations decreases with the overlap ratio σ, even if the overall complexity is similar because of computing thresholds $\widetilde{d}_{mn}$.

5. Experimental Results

We now consider a well-known dataset and introduce quantitative measures of patch size uniformity and symmetry to evaluate our methods for symmetry-aware segmentation into uniform overlapping patches. We also provide experimental evidence that intuitive alternatives adapted from classical FPS [PC06], state-of-the-art SFE [LCDFh10] and patch growing [HGLT09], cannot achieve such a segmentation.

5.1. Evaluation Setup

We use a subset of the TOSCA dataset [BBK08], comprising 4 objects (Cat, Dog, Horse, Victoria) with resolution ranging from 19k to 46k vertices, a null genus but positive number of boundary components, and varying poses for a total of 40 meshes. All objects possess bilateral symmetry, the correspondence for which we get via the perfect mirror symmetry of the neutral pose per object. Since pose changes are asymmetric and non-isometric, the meshes have neither extrinsic nor exact intrinsic symmetry (apart from neutral poses). We compare different segmentation methods by using two metrics we introduce (all scores will be given in percentage).

We first propose a quantitative measure of size uniformity with the normalised coefficient of variation for patch sizes:

$$\mathrm{SU}(\mathcal{P}_1,\ldots,\mathcal{P}_M) = \frac{1}{\sqrt{M-1}} \times \frac{\mathrm{stdev}_i(\mathrm{size}(\mathcal{P}_i))}{\mathrm{mean}_i(\mathrm{size}(\mathcal{P}_i))} \quad , \qquad (11)$$

where the patch size is the sum of areas for all faces contained within the patch. The coefficient of variation is a standardised measure of dispersion and thus provides a scale-invariant uniformity index. Here, a value of 0% means a perfectly uniform, best-case segmentation (i.e. all patches have identical size), while higher values indicate the degree to which patch sizes vary up to 100% for a worst-case segmentation (i.e. all patches but one have null size).

We also use a quantitative measure of segmentation symmetry with the ratio of vertices that violate symmetry:

$$\mathrm{SS}(\mathcal{P}_1,\ldots,\mathcal{P}_M) = \frac{1}{N} \times \mathrm{card}(\cup_i \{\mathcal{P}_i \setminus \widetilde{\mathcal{P}}'_i\}) \quad , \qquad (12)$$

where $\widetilde{\mathcal{P}}'_i$ is obtained by applying the symmetry operator to the patch that is paired with $\mathcal{P}_i$ as sharing symmetric centres. A value of 0% indicates a perfectly symmetric, best-case segmentation (i.e. all paired patches are symmetric), while higher values mean less symmetry up to 100% for a worst-case segmentation (i.e. each vertex belongs at least to a patch whose paired patch does not contain its symmetric partner).

Patch number (M)		25	50	75	100	150	200
SFE-GTS	SU(%)	95.1	82.1	58.3	48.0	21.0	18.4
SFE-GIS	SU(%)	81.8	40.8	20.4	15.8	12.1	10.5
FPS-NS	SU(%)	8.5	4.8	3.7	3.1	2.5	2.2
FPS-SA	SU(%)	8.0	4.3	3.3	2.8	2.2	2.0

Table 1: *Quantitative evaluation of mesh segmentation uniformity. Using the proposed measure of size uniformity* SU*, our symmetry-aware FPS method* FPS-SA *outperforms the naive FPS adaptation* FPS-NS *and both variants* SFE-GTS/GIS *of classical FPS in state-of-the-art SFE space.*

5.2. Mesh Segmentation

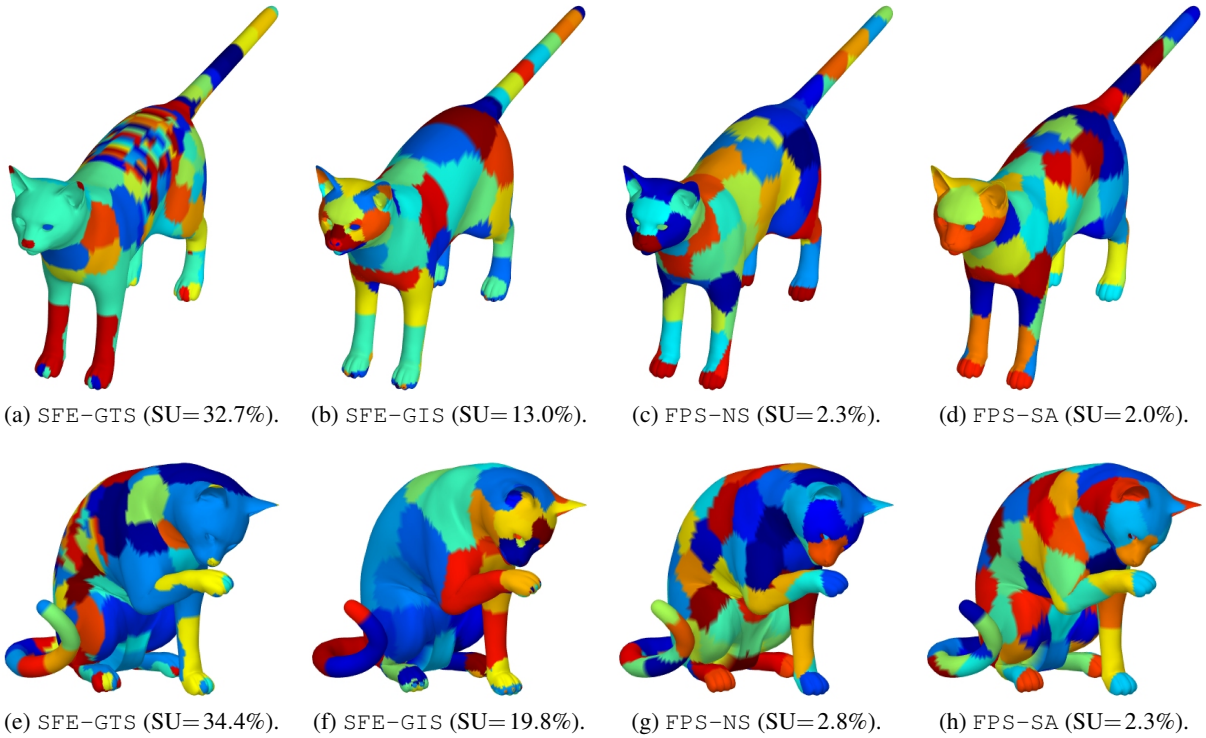
We begin by evaluating the uniformity of segmentation before building overlaps. We do not show the non-symmetric version here since this reduces to classical FPS (see supplementary material for additional results and failure cases).

We compare our symmetry-aware FPS (referred to as FPS-SA) with two main alternatives. First, we test a naive symmetric adaptation of classical FPS [PC06]. Half of the mesh is discarded (self-symmetric vertices are retained), the remaining half is segmented using FPS and the segmentation is copied over to the discarded half (FPS-NS). Second, we perform extrinsic FPS in SFE space [LCDFh10]. For a fair comparison, we build the correspondence matrix with the ground-truth symmetry, by computing dissimilarity on the neutral pose and sampling the known planar mirror symmetry as the only transformation (SFE-GTS). In a variant, we construct the correspondence matrix using their suggested dissimilarity measure of global intrinsic symmetry (SFE-GIS). We set the time parameter to 20, the localisation parameter to 1% and retain 100 eigenvectors of the correspondence matrix in both versions. We also keep 100 eigenvectors of the cotangent Laplacian in the latter version.

We vary the number M of patches and display size uniformity (SU) averaged over all meshes (Tab. 1). Naive symmetric FPS FPS-NS and our symmetry-aware FPS FPS-SA clearly outperform SFE versions SFE-GTS/GIS. This is not surprising because distance in SFE space is related to the degree of symmetry between points, there is no notion of geodesic distance over the mesh surface. Hence, uniformity in SFE space does not produce uniform patches. FPS-SA also outperforms FPS-NS. In addition, FPS-NS has some practical drawbacks which make it unattractive. Segmentation results depend on which half of the mesh is discarded (we compute both alternatives and show the average), and there is no obvious way to fix uniformity issues that arise along the symmetry line when the segmentation is copied.

We visualise examples on the Cat model for $M = 100$ patches (Fig. 4). In the top row, we show the neutral pose with exact extrinsic symmetry. In this case, our symmetry-aware distance reduces to the geodesic distance. Nevertheless, FPS-SA provides a more uniform segmentation than FPS-NS because of our symmetric modification to FPS.

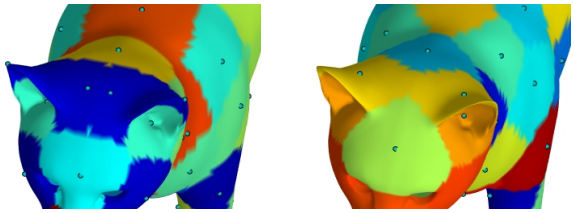
© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.



(a) SFE-GTS (SU=32.7%). (b) SFE-GIS (SU=13.0%). (c) FPS-NS (SU=2.3%). (d) FPS-SA (SU=2.0%).

(e) SFE-GTS (SU=34.4%). (f) SFE-GIS (SU=19.8%). (g) FPS-NS (SU=2.8%). (h) FPS-SA (SU=2.3%).

Figure 4: *Visualisation of mesh segmentation examples. The Cat model with exact extrinsic symmetry (top row) and approximate intrinsic symmetry (bottom row) is segmented using the four different algorithms (size uniformity is shown in brackets).*



(a) FPS-NS (SU=2.3%). (b) FPS-SA (SU=2.0%).

Figure 5: *Effect of symmetric modification. The naive FPS adaptation FPS-NS fails to produce uniform patches by placing symmetric keypoints close to the symmetry line in contrast to our symmetry-aware FPS modification FPS-SA.*

This effect can be seen by overlaying the samples (Fig. 5). FPS-NS sometimes places proper symmetric samples close to, but not on, the symmetry line (e.g. between ears). With FPS-SA, the same region is segmented by a proper symmetric pair, but the distance between centres is accounted for by FPS leading to a more uniform sampling. In the bottom row, the model exhibits only approximate intrinsic symmetry. Here, the advantage of our symmetry-aware distance becomes apparent as the size uniformity only slightly degrades, whereas results are significantly less uniform for other methods. Besides the clear non-uniformity of SFE-GTS/GIS, there are also some practical drawbacks. The method has a number of parameters that need tuning. In addition, the patches are not necessarily connected and there is no distinction between self-symmetric and proper symmetric patches. These would need to be detected as a post-processing step.

5.3. Patch Growing

We now evaluate the uniformity of our symmetry-aware patch growing based on the distance between patch centres (DPC-SA), and test the non-symmetric version (DPC-GD) using the geodesic distance instead of our symmetry-aware distance (see supplementary material for additional results). We also compare to the existing patch growing [HGLT09] based on the length of patch boundaries (LPB-GD).

We fix a number of $M = 100$ patches, vary the overlap ratio σ and display size uniformity (SU) as well as segmentation symmetry (SS) averaged over all meshes (Tab. 2). Using the distance between patch centres in DPC-GD/SA outperforms the criterion based on the length of patch boundaries in LPB-GD with respect to uniformity. Moreover, LPB-GD does not provide intuitive controls on overlap (we grow patches with a binary search on the overlap parameter per mesh until their total size is within 1% of that for our method). It is also unclear how to define the boundary lengths (we define two boundaries per neighbours, consisting of all edges from one patch that belong to a face such that the remaining vertex is in the other patch, and average

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

Overlap ratio (σ)		0	0.05	0.1	0.15	0.2	0.25
LPB-GD	SU(%)	2.8	2.9	3.2	3.3	3.3	3.2
	SS(%)	0.0	0.4	1.2	1.9	2.6	3.2
DPC-GD	SU(%)	2.8	2.9	3.1	3.0	2.9	2.9
	SS(%)	0.0	0.5	1.1	1.6	2.3	2.9
DPC-SA	SU(%)	2.8	2.9	3.1	3.0	2.9	2.9
	SS(%)	0.0	0.0	0.0	0.0	0.0	0.0

Table 2: *Quantitative evaluation of patch growing uniformity and symmetry. Using the proposed measures of size uniformity* SU *and segmentation symmetry* SS*, our symmetry-aware patch growing method* DPC-SA *outperforms the existing strategy* LPB-GD*, while preserving the uniformity of the asymmetric version* DPC-GD*, as well as the symmetry constraints in contrast to both methods* DPC/LPB-GD*.*

their lengths). Lastly, only DPC-SA inherently provides a perfectly symmetric segmentation, while both DPC/LPB-GD violate symmetry constraints to a significant extent.

We visualise examples on the Victoria model for an overlap ratio $\sigma = 0.2$ (Fig. 6). In addition to maintaining uniformity, only our approach DPC-SA is perfectly symmetric, whereas the grown patches of DPC-GD and LPB-GD contain respectively 1,333 and 1,057 vertices that violate symmetry.

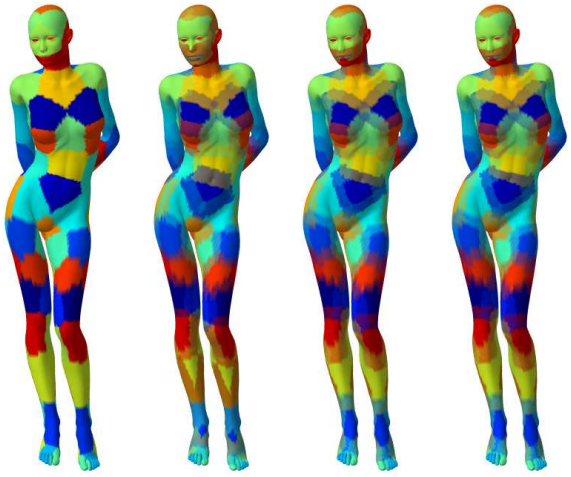
6. Segmentation Applications

We finally show the utility of our segmentation methods on applications to texture stitching and texture synthesis, where we improve results over state-of-the-art approaches.

6.1. Texture Stitching

The first application we consider is stitching a texture from multiple partial views, where we exploit a uniform overlapping (but non-symmetric) segmentation. Standard techniques either average the available textures or select textures from the best view per vertex. Such baselines, however, are likely to contain seams and blurs. A state-of-the-art technique is rather to blend the best textures per vertex by solving a screened Poisson equation [CLB*09]. Nevertheless, some artefacts may still appear due to numerous noise sources in the acquisition pipeline. We propose to tackle this with a patch-based rather than vertex-based approach.

We use the Rooster dataset of [CLB*09], which consists of a model reconstructed from 8 depth maps, aka range scans, by zippering. The mesh has 68,612 vertices, a genus of 1 due to a visible handle, and 1 boundary component as a result of a small hole in the top. We first sample views on the mesh via depth buffering and bilinear interpolation in the pixel grid. We then segment the mesh into uniform overlapping patches. For each view, we compute the mean angle between the viewer and vertex normals in the different patches. Unobserved vertices, due either to occlusion or



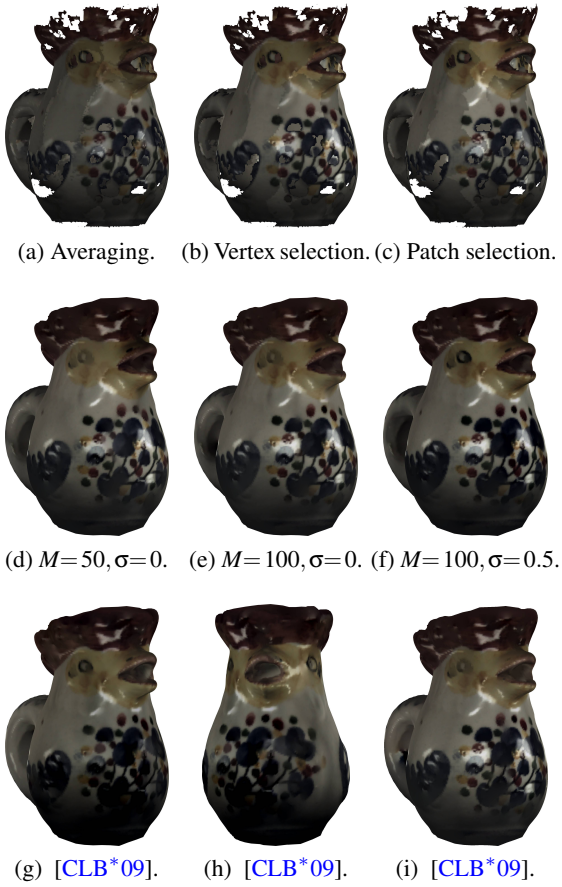
(a) FPS-SA (SU = 3.7%, SS = 0.0%). (b) LPB-GD (SU = 4.2%, SS = 2.3%). (c) DPC-GD (SU = 3.6%, SS = 2.9%). (d) DPC-SA (SU = 3.6%, SS = 0.0%).

Figure 6: *Visualisation of patch growing examples. The segmentation of the Victoria model (left) is grown using three different algorithms (size uniformity and segmentation symmetry are shown in brackets).*

missing information, are assumed to have a viewing angle of $\pi/2$. Hence, patches with unobserved data are penalised, and no difference on the nature of non-observability is made. For each patch now, we select texture from the view with smallest average viewing angle. Unobserved vertices are filled in with subsequent sorted views. We finally blend all selected textures by solving a screened Poisson equation via a global least-square problem (see supplementary material for further explanations), where texture gradients are selected by least-angle per face. We also fill in unobserved vertices by setting their gradients to zero. Regularisation in the screened Poisson equation is done via a coarse texture obtained by averaging the selected textures in overlaps, and using a small penalty $\lambda = 10^{-6}$ to remove colour offset indeterminacies.

The results demonstrate the benefits of our methods on this difficult dataset (Fig. 7). The two baseline vertex-based strategies suffer from variations in illumination across scans. Averaging textures produces multiple seams where some scans become or cease to be observable, and a general colour offset resulting in a darker texture due to accounting for shadowed vertices (Fig. 7a). Selecting the less foreshortened vertices addresses these issues, but introduces even more marked seams at transitions between selected scans (Fig. 7b). The coarse texture obtained by selecting patches ($M = 100, \sigma = 0.5$) and averaging textures in overlaps reduces this latter problem, but some seams are still visible (Fig. 7c). Moreover, inherent to all three approaches, unobserved vertices are not textured at all. Our approach tackles all these issues. To assess the effect of the patch number and overlap, we compute stitching for different values of M, σ. For a small number of patches ($M = 50, \sigma = 0.0$), we observe slight

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.



(a) Averaging. (b) Vertex selection. (c) Patch selection.

(d) $M=50, \sigma=0$. (e) $M=100, \sigma=0$. (f) $M=100, \sigma=0.5$.

(g) [CLB*09]. (h) [CLB*09]. (i) [CLB*09].

Figure 7: *Texture stitching of multiple partial views. We compare our patch-based approach to a simplified one, as well as two baseline and one state-of-the-art vertex-based schemes. Merging vertex information is not sufficient on this dataset, whereas patch blending improves quality and removes seams when using enough patches and overlap.*

seams, a ghosting effect around the eye and blurred specularities (Fig. 7d). Increasing the patch number ($M=100, \sigma=0.0$), specularities are improved, yet we still observe slight seams and the ghosting artefact (Fig. 7e). The best result is when further growing the patches to overlap ($M=100, \sigma=0.5$), where the obtained stitching is seamless (Fig. 7f). In contrast, the state-of-the-art texture stitcher [CLB*09] blurs the specularities with the painted pattern (Fig. 7g). This is even more marked on the frontal view, where the central specularity is missed and the details in the bottom half of the painted pattern are lost due to a noticeable colour bleed from the base (Fig. 7h). However, this is not necessarily due to using a vertex-based approach. Indeed, their pipeline differs from ours in several stages (e.g. selection by least-distance, different sampling, screening term and Poisson solver). For a fair comparison, we thus reimplemented their algorithm in our pipeline, so that results only depend on whether we use patches or not for robustness to noise. In this case, the specularities are preserved and the colour bleed is removed, but a ghosting artefact still appears around the eye (Fig. 7i).

6.2. Texture Synthesis

As a second application, we employ our symmetric, uniform, overlapping segmentation for texture synthesis. Specifically, we address 3D face texture synthesis. A state-of-the-art approach is to generate random realisations of a 3D morphable model (3DMM), such as the Basel Face Model [PKA*09]. A drawback is that details are averaged out due to statistical regularisation. We tackle this by importing texture from a patch database to hallucinate missing details.

We use the 3DMM [PKA*09]. The model has 53,490 vertices, a null genus, 1 boundary component, and has been symmetrically remeshed via template fitting. We segment the mean face with symmetric eye centres as seeds ($M=200, \sigma=0.5$), and build a patch database with images from the CMU PIE face database [SBB02]. The fitted 3DMM provides a model for each subject and a dense correspondence to each image. We stitch textures from the 3 available views under same illumination per subject to provide a complete texture. As the 3DMM does not account for glasses or facial hair, we exclude corresponding subjects, leading to a total of 770 faces representing 35 subjects under 22 lightings.

We obtain the reference face by randomly generating shape and texture coefficients from the 3DMM and arbitrary lighting parameters for rendering. We then replace patches from this target face with patches from the database, by selecting them so as to minimise the Euclidean error. We finally apply Poisson blending (see supplementary material for further explanations), with least-error selection of gradients in overlaps. To illustrate the importance of overlap, symmetry and seeding, we also try out reconstruction by removing one of these features at a time. The asymmetric version uses both asymmetric segmentation and growing, whereas symmetric variants employ a symmetric segmentation and growing, and pick patches by symmetric pairs.

Overall, the results prove the usefulness of our methods (Fig. 8). The state-of-the-art 3DMM [PKA*09] provides a globally coherent texture but locally lacks of realistic asperities (Fig. 8a). In contrast, we import plausible details and produce a coherent face both globally and locally (Fig. 8b). Moreover, the synthesised face represents an original individual as a hybrid between several training subjects, which can be seen from assigning a colour to each patch based on the selected identity (Fig. 8c). Importantly, having no overlap at all makes some artefacts appear, notably below the right eye (Fig. 8d). Even more dramatic, the asymmetric version leads to an unrealistic face where the left eye is much darker than the right one (Fig. 8e). We also point out a subtle colour difference between two halves of the left iris when not having eye centres as seeds (Fig. 8f). Finally, hallucinating other random face realisations provides as many different but still plausible hybrids (Fig. 8g–i).

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

(a) [PKA*09]. (b) Hallucination. (c) Identities.

(d) No overlap. (e) No symmetry. (f) No seeds.

(g) Hallucination. (h) Hallucination. (i) Hallucination.

Figure 8: *Texture synthesis of random faces. A random face is generated from a state-of-the-art 3DMM and improved by importing texture patches. This provides a hybrid, realistic face with global consistency and local details, where artefacts are reduced via overlap, and colour issues of the eyes are avoided via semantic constraints of symmetry and seeds.*

7. Conclusion

We presented methods for symmetry-aware mesh segmentation into uniform overlapping patches. We now discuss some limitations and possible extensions for future work.

Mesh modeling We restricted to triangular meshes, due to the fast marching code used here for convenience. By plugging in a more elaborate scheme for geodesic computation, our methods could further be applied to polygonal meshes.

Segmentation criteria We considered a specified number of patches and overlap ratio. However, it is straightforward to choose other patch criteria (e.g. minimum distance between patch centres, average patch size or boundary length). We can also enforce alternate overlapping criteria (e.g. growing patches up to next neighbouring depths), though care should be taken to maintain patch uniformity. We could even obtain almost (due to discretisation) perfectly uniform patches by growing them until they reach a given size, although this obviously removes any intuitive controls on overlap.

Symmetry types We focused on a single and full, reflexive symmetry. Translational or rotational symmetry can be handled easily by including the previous samples on both sides instead of that on one side only plus the virtual symmetric sample. For multiple symmetries, we duplicate the manifold as necessary and define the symmetry-aware distance on the product manifold accordingly. If a symmetry is partial, then uninvolved vertices can be set to self-symmetric for this symmetry. In all cases, we must sample simultaneously all vertices that lie in the same symmetry orbit, and account for the minimum pairwise distance within orbits.

Texture stitching We assumed a constant appearance across the views to be stitched. A potential improvement is to include colour calibration and correction steps for camera and light parameters. In addition, preserving specularity for stitching is not always desired, and an interesting perspective to address this is to incorporate an inverse rendering model.

Texture synthesis We observed limitations in the synthesis variability and quality. Variability could be addressed either by adding more training face examples to the database, or by building a database of intrinsic textures and optimising for camera and illumination parameters during synthesis. Quality could be enhanced by improving the database quality itself. Our stitching procedure from 2D images also leads to inherent artefacts on training faces that are inevitably transferred to synthesised faces. Exploiting 3D capture for training would tackle this issue. Lastly, we could favour the neighbouring compatibility of selected patches to reduce seams, via belief propagation instead of our greedy strategy where patches are selected independently of each other.

Further applications Other potential applications of our segmentation methods include mesh restoration tasks such as super-resolution, denoising, deblurring and inpainting. Our symmetry-aware distance may also find different applications than segmentation in flattening, matching or classifying surfaces with symmetries. A thorough consideration of each task is however needed to assess whether our methods can improve results compared to traditional approaches.

Additional Resources

We provide supplementary material and MATLAB source code on a companion website: `http://www-users.cs.york.ac.uk/wsmith/segmentation`.

Acknowledgements

This work was supported by grant DSTLX1000070369 from the Defence Science & Technology Laboratory. We are grateful to P. Simari for providing us with symmetric mesh data, to H. Elgabou for his help to process some models, and to G. Peyré for his assistance with the fast marching code.

References

[BBK08] BRONSTEIN A. M., BRONSTEIN M. M., KIMMEL R.: *Numerical Geometry of Non-Rigid Shapes*. Springer, 2008. 7

[CAF10] CHO T. S., AVIDAN S., FREEMAN W. T.: The patch transform. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 8 (2010), 1489–1501. 1

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

[CCS12] CORSINI M., CIGNONI P., SCOPIGNO R.: Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. Vis. Comput. Graphics 18*, 6 (2012), 914–924. 2

[CJW*09] CLINE D., JESCHKE S., WHITE K., RAZDAN A., WONKA P.: Dart throwing on surfaces. *Comput. Graph. Forum 28*, 4 (2009), 1217–1226. 2

[CLB*09] CHUANG M., LUO L., BROWN B. J., RUSINKIEWICZ S., KAZHDAN M.: Estimating the Laplace-Beltrami operator by restricting 3D functions. *Comput. Graph. Forum 28*, 5 (2009), 1475–1484. 9, 10

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. Graph. 23*, 3 (2004), 905–914. 2

[DFKE07] DABOV K., FOI A., KATKOVNIK V., EGIAZARIAN K.: Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. Image Process. 16*, 8 (2007), 2080–2095. 1

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Siggraph* (2001), pp. 341–346. 1

[FJP02] FREEMAN W. T., JONES T. R., PASZTOR E. C.: Example-based super-resolution. *IEEE Comput. Graph. Appl. 22*, 2 (2002), 56–65. 1

[FZ09] FU Y., ZHOU B.: Direct sampling on surfaces for high quality remeshing. *Comput. Aided Geom. D. 26*, 6 (2009), 711–723. 2

[HGLT09] HÉTROY F., GÉROT C., LU L., THIBERT B.: Simple flexible skinning based on manifold modeling. In *Int. Conf. on Computer Graphics Theory and Applications* (2009), pp. 259–265. 2, 3, 6, 7, 8

[LCDFh10] LIPMAN Y., CHEN X., DAUBECHIES I., FUNKHOUSER T.: Symmetry factored embedding and distance. *ACM Trans. Graph. 29*, 4 (2010), 103:1–103:12. 2, 3, 7

[LLLF08] LI H., LO K.-Y., LEUNG M.-K., FU C.-W.: Dual Poisson-disk tiling: An efficient method for distributing features on arbitrary surfaces. *IEEE Trans. Vis. Comput. Graphics 14*, 5 (2008), 982–998. 2

[LLT05] LIU W., LIN D., TANG X.: Neighbor combination and transformation for hallucinating faces. In *IEEE Int. Conf. on Multimedia and Expo* (2005), pp. 145–148. 1

[LLX*01] LIANG L., LIU C., XU Y.-Q., GUO B., SHUM H.-Y.: Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph. 20*, 3 (2001), 127–150. 1

[LW11] LIU S., WU J.: Fast patch-based image hybrids synthesis. In *IEEE Int. Conf. on Computer-Aided Design and Computer Graphics* (2011), pp. 191–197. 1

[MD03] MOENNING C., DODGSON N. A.: Fast marching farthest point sampling. In *Eurographics* (2003). 2

[MES08] MAIRAL J., ELAD M., SAPIRO G.: Sparse representation for color image restoration. *IEEE Trans. Image Process. 17*, 1 (2008), 53–69. 1

[MGP06] MITRA N. J., GUIBAS L. J., PAULY M.: Partial and approximate symmetry detection for 3D geometry. *ACM Trans. Graph. 25*, 3 (2006), 560–568. 2

[MPK09] MOHAMMED U., PRINCE S. J. D., KAUTZ J.: Visiolization: Generating novel facial images. *ACM Trans. Graph. 28*, 3 (2009), 57:1–57:8. 1

[PC04] PEYRÉ G., COHEN L. D.: Surface segmentation using geodesic centroidal tesselation. In *Int. Symp. on 3D Data Processing, Visualization and Transmission* (2004), pp. 995–1002. 2

[PC06] PEYRÉ G., COHEN L. D.: Geodesic remeshing using front propagation. *Int. J. Comput. Vis. 69*, 1 (2006), 145–156. 2, 3, 4, 7

[PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: Lapped textures. In *Siggraph* (2000), pp. 465–470. 1

[PGR07] PODOLAK J., GOLOVINSKIY A., RUSINKIEWICZ S.: Symmetry-enhanced remeshing of surfaces. In *Eurographics Symp. on Geometry Processing* (2007), pp. 235–242. 2

[PKA*09] PAYSAN P., KNOTHE R., AMBERG B., ROMDHANI S., VETTER T.: A 3D face model for pose and illumination invariant face recognition. In *IEEE Int. Conf. on Advanced Video and Signal Based Surveillance* (2009), pp. 296–301. 10, 11

[PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L. J.: Discovering structural regularity in 3D geometry. *ACM Trans. Graph. 27*, 3 (2008), 43:1–43:11. 2

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3D shapes. *ACM Trans. Graph. 25*, 3 (2006), 549–559. 2

[SBB02] SIM T., BAKER S., BSAT M.: The CMU Pose, Illumination, and Expression (PIE) database. In *IEEE Int. Conf. on Automatic Face and Gesture Recognition* (2002), pp. 46–51. 10

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. *Comput. Graph. Forum 27*, 6 (2008), 1539–1556. 2

[SKS06] SIMARI P., KALOGERAKIS E., SINGH K.: Folding meshes: Hierarchical mesh segmentation based on planar symmetry. In *Eurographics Symp. on Geometry Processing* (2006), pp. 111–119. 2

[SPDF14] SIMARI P., PICCIAU G., DE FLORIANI L.: Fast and scalable mesh superfacets. *Comput. Graph. Forum 33*, 7 (2014), 181–190. 1

[TCY09] TOTZ J., CHUNG A. J., YANG G.-Z.: Patient-specific texture blending on surfaces of arbitrary topology. In *Workshop on Augmented environments for Medical Imaging and Computer-aided Surgery* (2009), pp. 78–85. 2

[VC04] VALETTE S., CHASSERY J.-M.: Approximated centroidal Voronoi diagrams for uniform polygonal mesh coarsening. *Comput. Graph. Forum 23*, 3 (2004), 381–389. 2

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the art in example-based texture synthesis. In *Eurographics* (2009), pp. 93–117. 1

[WSBY06] WU Q., SHI L., BOND S., YU Y.: Laplacian texture synthesis and mixing on surfaces. In *Pacific Conf. on Computer Graphics and Applications* (2006), pp. 50–59. 1

[XZJ*12] XU K., ZHANG H., JIANG W., DYER R., CHENG Z., LIU L., CHEN B.: Multi-scale partial intrinsic symmetry detection. *ACM Trans. Graph. 31*, 6 (2012), 181:1–181:11. 2

[YBZW14] YAN D.-M., BAO G., ZHANG X., WONKA P.: Low-resolution remeshing using the localized restricted Voronoi diagram. *IEEE Trans. Vis. Comput. Graphics 20*, 10 (2014), 1418–1427. 2

[YW13] YAN D.-M., WONKA P.: Gap processing for adaptive maximal Poisson-disk sampling. *ACM Trans. Graph. 32*, 5 (2013), 148:1–148:15. 2

[ZHW*06] ZHOU K., HUANG X., WANG X., TONG Y., DESBRUN M., GUO B., SHUM H.-Y.: Mesh quilting for geometric texture synthesis. *ACM Trans. Graph. 25*, 3 (2006), 690–697. 1

[ZW11] ZORAN D., WEISS Y.: From learning models of natural image patches to whole image restoration. In *IEEE Int. Conf. on Computer Vision* (2011), pp. 479–486. 1

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.