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

Towards Globally Optimal Normal Orientations for Large Point Clouds

Nico Schertler, Bogdan Savchynskyy, and Stefan Gumhold

TU Dresden, Germany

Abstract

Various processing algorithms on point set surfaces rely on consistently oriented normals (e.g. Poisson surface reconstruction). While several approaches exist for the calculation of normal directions, in most cases their orientation has to be determined in a subsequent step. This paper generalizes propagation-based approaches by reformulating the task as a graph-based energy minimization problem. By applying global solvers, we can achieve more consistent orientations than simple greedy optimizations. Furthermore, we present a streaming-based framework for orienting large point clouds. This framework orients patches locally and generates a globally consistent patch orientation on a reduced neighbor graph, which achieves similar quality to orienting the full graph.

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

With the increasing availability of scene acquisition technology, point clouds have become an important way to represent surfaces. Assigning only a position to points is not enough for most use cases. Shaded renderings, most surface reconstruction algorithms, or scan registration are some examples that additionally rely on oriented point normals.

Given a set of point positions, the direction of point normals can be estimated by locally fitting a low-order surface whose normal can be calculated analytically, e.g. PCA is one simple means of fitting a plane [Pea01]. However, those fitting processes leave the orientation of a normal ambiguous, i.e. the correct normal could be the calculated one or its inverse. If the acquisition scenario is known and a robust normal estimation method is used, a consistent orientation can usually be achieved by setting all normals to point towards the sensor. If, however, only raw point cloud data are available, the orientation must be computed in a separate step. Furthermore, non-robust normal estimation methods can result in wrong orientations even if the sensor position is known, especially in the presence of noise.

In this article, we present a propagation-based method that finds approximately optimal orientations for raw unorganized point sets. Our main contributions are:

- Formal definition of the orientation problem as a discrete graph-based labeling problem and application of advanced solving techniques
- Presentation of a streaming-based out-of-core framework for large data sets

After formalizing the orientation problem, we analyze how this problem has been solved traditionally. We use the results of this analysis to introduce more involved solving techniques, which can achieve globally optimal orientations. The major part of this article focuses on our out-of-core framework, which integrates the presented solvers into a streaming-based approach. This approach can be easily applied to large data sets that do not fit in the computer's main memory. Such data are usually generated by combining registered scans into a single one. While the single scans may contain sparsely-sampled areas, their combination is usually sampled more densely, which makes normal propagation easier. Furthermore, the streaming approach can significantly speed up orientation for medium-sized data sets.

2. Related Work

Most existing normal orientation algorithms can be classified into propagation-based and volume-based approaches. The following section gives a short overview of these two

classes and some other approaches that do not fit in this system. For a detailed survey on surface reconstruction from point clouds, we refer the reader to [BTS*14]. Additionally, we review a few variants of point cloud streaming - the method we chose to use for orienting large point clouds.

2.1. Propagation-Based Methods

Propagation-based methods try to deduce the orientation of a point's normal from known orientations in its local neighborhood. To accomplish this, a measure must be calculated that defines for each pair of neighboring vertices the uncertainty with which one orientation can be calculated given the other. Based on this measure, a spanning tree of the neighbor graph is calculated that minimizes the overall uncertainty. Then, starting from one or more vertices with known normal orientation, the tree is traversed and the known orientations are propagated across the surface by flipping normals if necessary (according to a flip criterion).

One early approach is that of Hoppe [HDD*92]. This method uses the term $1 - |\langle n_i, n_j \rangle|$ as the uncertainty measure, where $n_{i,j}$ are the two normals of adjacent points. The flip decision is based on the dot product's sign. A single point with an extremal coordinate along one of the principal axes is chosen as the starting point of the propagation because its orientation can be inferred under the assumption that the point cloud represents a closed surface. The approach works well for smooth and densely sampled surfaces with low curvature but can fail at sharp edges and creases.

Since then, several improvements have been developed . Xie et al. [XWH*03] developed a flip criterion that is able to orient creases and close surface sheets. Furthermore, they propose to use several starting points and to propagate only along reliable edges. The resulting set of oriented patches is stitched together with a modified flip criterion. König's flip criterion [KG09] assumes low curvature by evaluating the complexity of Hermite splines defined by the point normals. In [HLZ*09], the authors propose to run an initial consolidation step (WLOP), which denoises the point cloud. The propagation is then done with a modified priority measure that can handle close-by surface sheets. An orientation-aware PCA is utilized to iteratively improve normal directions based on the orientation.

In [SBY11], Seversky et al. showed an approach that propagates gradients of harmonic functions instead of normal orientations across an MST to find a globally optimal orientation.

2.2. Volume-Based Methods

Volume-based methods interpret the input vertices as samples from the zero-level set of an implicit function (usually a signed distance function) that differentiates the inside from the outside. Consequently, these approaches can only be used for point clouds that represent manifold surfaces with no boundary. Point normals must always point to the outside, i.e. in a direction where the value of the implicit function increases. The function's gradient is usually a good approximation for the normal (both direction and orientation). However, more involved methods can be used to estimate the normal's direction more accurately, using the gradient only for orientation.

While the calculation of an unsigned distance function is relatively easy, introducing a sign, which defines the orientation, is quite complicated. Mullen et al. [MDGD*10] achieve this by shooting random rays from surface samples and evaluating their intersections with the entire point set. Giraudot et al. [GCSA13] reformulate the problem as an energy-minimization task based on a regular grid, which tries to find the best of a finite number of orientation hypotheses. Gong et al. [GPS12] use the original unsigned distance function to calculate a non-zero level set, which represents a band around the original point set. This level set can be divided into an inner and outer set (by connected component analysis). Distances to these bands are then used to reconstruct the sign of the unsigned distance function.

2.3. Other Orientation Methods

Normals can also be oriented based on visibility. Such methods have been studied primarily for polygonal meshes by Borodin et al. [BZK04], but it is possible to apply them to point clouds, too. The idea is to evaluate the visibility of faces or points (entities) from different view points and to maximize the front face visibility, preserving local consistency. Katz et al. [KTB07] propose a visibility algorithm for point clouds, which makes an adaptation of Borodin's approach for this domain imaginable.

The authors of [WYC12] show a variational model that finds normal directions and orientations in a single step using an energy minimization approach. The problem is expressed in matrix form that encodes constraints and penalties on the result based on the input point cloud and its neighborhood. The problem is then relaxed and an approximate solution is found via eigen decomposition. A similar variational framework, which unites normal estimation and orientation, is proposed in [ACSTD07].

In [LW10], the point cloud is first approximated with a coarse triangulation, which holds valuable topological information. Orientations on this mesh can be found comparatively easy and can thus be used to orient the point cloud normals.

2.4. Point Cloud Streaming

Streaming refers to an out-of-core strategy that can be used to process large data sets. The basic idea is to arrange the data points in a sequential stream. This input stream is then

used to read little amounts of data into main memory and process them. As soon as an entity is no longer needed, it is removed from main memory. During processing, results are written out to an output stream.

Pajarola [Paj05] proposes to create the stream by sorting the given points according to their x-coordinate. This results in a sweeping plane being moved through the data set during processing. The streaming process can be made more efficient by rotating the data to map its longest principal axis onto the x-axis before constructing the stream. Points are removed from main memory as soon as the k-neighborhood of subsequent points is complete.

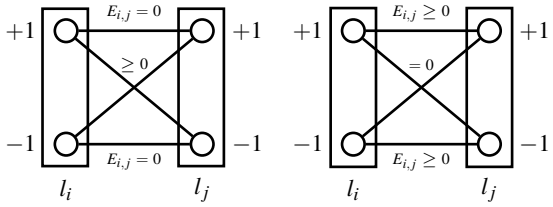
Isenburg et al. [ILSS06] construct the stream order using a grid. The input data set is buffered in the grid cell until all points of the cell have been read. When this is the case, these points are released to the stream. Due to their application to Delaunay triangulation, randomly chosen points are moved to the beginning of the stream to improve the triangulator's efficiency. Points are removed from main memory when they don't have an impact on future triangulations. This decision is based on cell-level finalization tags, which are inserted into the stream to express that no further point will lie in that cell.

3. Reformulation of the Orientation Problem

Hoppe formulated the orientation problem as a graph optimization problem. Keeping the main idea, we will define it as a maximum-likelihood problem on a *Markov Random Field* (MRF), which is a widely used model in e.g. Computer Vision. The MRF definition requires a graph and an energy function, which we will introduce in the following.

Given a set $\mathcal{P}$ of points with unoriented normals, an undirected neighbor graph $G = (\mathcal{P}, \mathcal{E} \subseteq \binom{\mathcal{P}}{2})$ serves as the basis of the MRF. In practice, the choice of the graph depends on the specific scenario. In the rest of this paper, we use a distance-truncated symmetrized k-nearest neighbor graph unless otherwise stated, i.e. two vertices are connected iff at least one is in the k-neighborhood of the other and their distance is at most r (the maximum neighbor search radius). A point's position and normal are denoted by p_i and n_i, respectively. Furthermore, each point is assigned one of two possible labels from the label set $\mathcal{L} = \{-1, +1\}$. This label serves as the point's normal factor. For a given labeling $L \in \mathcal{L}^{|\mathcal{P}|}$, a point's label is denoted by L_i. After the orientation process is finished, each point's normal is updated with $n_i \leftarrow L_i \cdot n_i$.

We base our energy definition on a flip criterion that specifies the normal relation of two neighbors. Previous approaches used criteria that consisted of two parts: a flip decision and an uncertainty measure. In order to integrate this flip criterion in our optimization framework, we fused both parts into a single real number. Thus, the flip criterion is a function $\phi : \mathcal{E} \to \mathbb{R}$. It yields a positive number if the normals



(a) $\phi(i,j) \geq 0$ (b) $\phi(i,j) < 0$

Figure 1: Visualization of (2) for positive and negative flip criterion output. The boxes represent nodes in the MRF, the circles in these boxes represent a specific label for the given MRF node.

are oriented consistently and a negative number otherwise. The result's absolute value specifies the certainty of the flip decision (note the difference to previous approaches, which used an uncertainty measure). We use certainty because this measure can be used directly as a weight for the resulting optimization problem. Hoppe's flip criterion is simply:

$$\phi_{Hoppe}(i,j) = \langle n_i, n_j \rangle \tag{1}$$

Xie's and König's more involved flip criteria can be expressed similarly. For details, we refer the reader to the supplementary material.

Based on this flip criterion, each edge is assigned a pairwise potential $E_{i,j} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$, which yields zero for consistently oriented normals and a positive penalty otherwise. In contrast to Hoppe, we additionally weight the flip criterion's output based on the neighbors' distance:

$$\begin{aligned} E_{i,j}(l_i, l_j) = & \\ & \begin{cases} |\phi(i,j)| \cdot \omega(p_i, p_j) & \phi(i,j) \geq 0 \oplus l_i = l_j \\ 0 & \text{otherwise} \end{cases} \\ \omega(p_i, p_j) = 1 - & \frac{\|p_i - p_j\|^2}{r^2} \quad , \end{aligned} \tag{2}$$

where $\oplus$ denotes the XOR operator and ω defines a weight based on the distance of two points, normalized by the maximum search radius. Figure 1 visualizes this potential definition.

The sum of all pairwise potentials forms the labeling's energy. The energy's minimizer is then the optimal orientation according to the chosen flip criterion:

$$\begin{aligned} E(L) &= \sum_{\{i,j\} \in \mathcal{E}} E_{i,j}(L_i, L_j) \\ L^* &= \arg\min_L E(L) \end{aligned} \tag{3}$$

For perfectly orientable data sets, the energy's minimum is 0 and the labeling that induces this energy is the desired result. If perfect orientation is not possible, the minimization

process tries to find a labeling, such that only edges with a small certainty are violated. The energy can be seen as the negative-log space equivalent of a labeling's probability. Thus, finding the energy's minimizer is equivalent to finding the most likely labeling.

4. Solving the Orientation Problem

In order to find the best orientation for each point normal, (3) has to be solved. The time complexity of this problem depends on the energy's submodularity [KZ04]. For binary labels, submodularity is defined as follows [KZ04, LRB07]:

$$\begin{aligned} &\textit{submodular} \iff \\ &E_{i,j}(1,1)+E_{i,j}(-1,-1) \leq E_{i,j}(1,-1)+E_{i,j}(-1,1) \\ &\qquad\qquad \forall\{i,j\} \in \mathcal{E} \end{aligned} \tag{4}$$

Comparing with (2) yields $\textit{submodular} \iff \phi(i,j) \geq 0$. Hence, the orientation problem is only submodular for the trivial case where all normals are already oriented consistently. In general, the problem is non-submodular, rendering it NP-hard [KZ04]. Therefore, most problem sizes make it infeasible to calculate a globally optimal orientation. Instead, approximate solvers can be used to find a good approximate solution.

Furthermore, due to the energy's symmetry with respect to the labeling, each configuration can be flipped entirely without affecting the total energy. Therefore, at least two global optima exist. This ambiguity does not pose a problem for the solvers presented in this paper. For others, it may be necessary to introduce a single unary term per connected component, which shifts the energy towards one of the two possible orientations.

4.1. Spanning Tree Solution (MST)

All propagation-based methods (cf. section 2.1) use spanning trees to solve (3). They all share the problem that they ignore edges that are not part of the spanning tree. In the simple case where all edges are conformant (i.e. there is a labeling that results in a zero-energy), these methods will find the optimal solution. However, they may fail if edges contradict each other. Figure 2 shows a small example where this is the case. Note that this figure is related to our adapted flip criterion and thus uses a maximum spanning tree (MST) with respect to the distance-weighted absolute flip criterion. The MST solver favors more prominent edges (with a high absolute value) over smaller ones. Consistently orienting a prominent edge may lead to inconsistencies of several smaller edges. The approach fails to find a global optimum if the sum of those smaller edges is larger than the single prominent edge.

The advanced methods we present in this paper do not come with this limitation because they respect all edges. However, the MST result provides a good initialization for some of these methods. Therefore, we developed an algorithmic improvement of the basic implementation. This improvement is based on a modification of the Union Find data structure, which allows to flip entire connected components in constant time. Through this modification, explicit calculation of the spanning tree is not necessary anymore, saving both run time and memory. We call this modified structure the *Signed Union Find*, which is about 2.2 times as fast as the traditional implementation. For details, please refer to the supplementary material.

An advantage of the MST solver is its quick execution. In a knn-graph, the time complexity of $\mathcal{O}(n_p \log n_p)$ (n_p being the number of points) is caused by the initial sorting step. The rest of the algorithm runs in effectively linear time for an optimized Union-Find data structure. This complexity makes it suitable even for large point clouds, as long as they do not exceed the computer's memory limit. For large data sets, we present our out-of-core strategy in section 5.

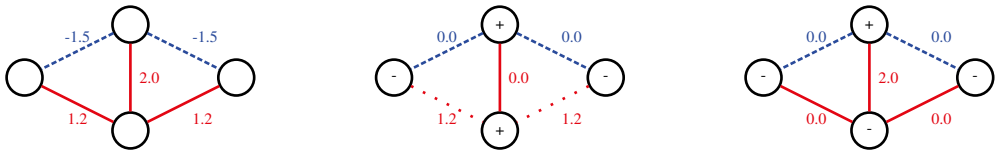
4.2. Advanced Solvers

As mentioned earlier, MRFs like those used to define the orientation problem (cf. (2)), are widely used in Computer Vision. As a consequence, a number of solvers for MRF-based energy minimization problems have been developed [KAH*15]. However, many of them focus on submodular functions or are only feasible for small models. In the following section, we introduce two solvers that do not come with these limitations, particularly QPBO and LSA-TR, and apply them to the orientation problem.

4.2.1. Quadratic Pseudo-Boolean Optimization (QPBO)

QPBO (quadratic pseudo-boolean optimization [BHS91]) is a graph-cut-based minimizer for MRFs with binary label space (as is the case for the orientation problem). However, its output labeling is not binary but pseudo-boolean, meaning that it can leave some nodes unlabeled.

For most of our test data sets, QPBO could not label any point. Therefore, we use the variant QPBO-I [RKLS07], where the *I* stands for *Improve*. For this, we first execute the MST solver to find an approximate solution. Then, after a first QPBO pass, all unlabeled nodes are iterated in a random order. If the iterated node is still unlabeled, this node's label is fixed to its MST solution and QPBO is executed again. This fixing of nodes is achieved by introducing unary terms, which eventually allows QPBO to compute a complete labeling. Since the iteration order is crucial, these steps are performed several times with different orders and the best result is returned. QPBO-I ensures that the resulting energy does not increase compared to the input labeling. We call this solver *MST+QPBO-I*. Figure 3 shows how *MST+QPBO-I* can improve the orientation quality over the traditional MST solver for one of our test data sets. A detailed analysis of results is given in section 6.



(a) Weighted output of flip criterion (b) MST solution, $E = 2.4$ (c) Optimal solution, $E = 2.0$

Figure 2: Example graph where the MST solver fails. a: Definition of the distance-weighted flip criterion. Red solid edges visualize positive values (aiming for equal labeling), blue dashed edges visualize negative values (aiming for contrasting labeling). b: Output labeling of the MST solver. Edges are labeled with the according potentials. Edges not belonging to the maximum spanning tree are represented as dotted lines. c: Optimal labeling.

4.2.2. Local Submodular Approximation (LSA)

Another way of solving the orientation problem is to approximate the energy with a submodular function, whose global minimum can be found in polynomial time. LSA-TR (local submodular approximation [GBV*14]) does this with the help of a Taylor expansion of the energy around a given initial solution. The approximation is chosen to be valid within a given distance to the initial labeling, which is referred to as the *trust region* (TR). The algorithm iteratively approximates and minimizes the energy. While iterating, the trust region is adapted based on the ratio of predicted energy reduction and actual energy reduction with respect to the previous step's solution. This way, a local optimum is found, whose energy is supposed to be near the global optimum's energy. Since the algorithm is designed to solve energies with few supermodular terms, we use the MST solution to permute the label semantics in order to turn non-submodular terms into submodular ones (such that $+1$ means to use the MST solution and -1 means to use its inverse).

5. Orienting Large Point Clouds

So far, we have presented methods to orient rather small point clouds. With high-end acquisition hardware, data sets reach sizes that make them unsuitable for the approaches presented above, especially when multiple scans are combined. Either because the data set does not fit in the computer's main memory or because the run time becomes unacceptably high (advanced solvers have beyond-linear polynomial time complexity). We, therefore, present an out-of-core orientation framework, which can handle nearly arbitrary point cloud sizes. The algorithm is based on the assumption that point set surfaces comprise patches whose points can be oriented easily within that patch. Global consistency must be respected only when stitching patches together. Our approach makes use of this assumption by executing the following steps:

1. Segment point cloud into orientable patches
2. Build a neighbor graph on these patches (segment graph)
3. Define a reduced MRF on the segment graph
4. Solve the orientation problem on the reduced MRF

Because the MRF must fit into main memory, this represents an upper bound of the input point cloud complexity. Note that the MRF size does not correlate directly with the data set size. For instance, a sphere tessellation of arbitrary resolution results in a single patch and with that in an MRF with a single node and no edges.

5.1. Segmentation

In this section, we explain the details of our segmentation method in the context of an in-core approach. Note that our final algorithm uses a streaming approach. The in-core variant is only explained for reasons of clarity. The segmentation step is followed by the orientation optimization, where the labels of points within a segment can change only if the labels of all other points within this segment change. This preserves the energy function but reduces the domain over which the energy is defined. The segmentation process aims at reducing the domain as much as possible without removing the energy's minimizer from it.

We consider a segment to be a continuous connected set of points that does not overlap with other segments. Starting with the bare point cloud, where all points are not assigned to any segment, the segmentation process is structured as follows: An arbitrary unassigned point is chosen (preferably one with neighbors that are already assigned to a segment). For this point, a list of candidate segments is retrieved from the neighboring points. The minimizer-preserving property of these candidates is then validated based on a heuristic. Afterwards, the best segment is chosen and a preliminary normal orientation is performed based on the neighbors in the segment, such that normals are consistent within their segment. If no valid segment is available, a new segment is created. This process is repeated until all points are assigned to a segment.

In this process, we rely on two measures: For any neighbor of the current point, the distance-weighted flip criterion output $\phi(i, j) \cdot \omega(p_i, p_j)$ specifies the **neighbor's vote** for the current point. The sum of neighbor votes from a specific segment defines the **segment's vote** for the current point. Figure 4 shows this relation.

(a) Hoppe, MST $E = 526.1$ (b) Hoppe, MST+QPBO-I $E = 380.0$ (c) Xie, MST $E = 121.9$ (d) Xie, MST+QPBO-I $E = 91.0$

Figure 3: Comparison of the MST solver (a, c) and QPBO (b, d) for the two flip criteria. A splatting-based renderer is used to visualize the point cloud. The backside of splats is rendered dark red. Apparently, Hoppe's flip criterion is not suited for this data set. Its ground truth's energy of 466.4 (calculated from the original mesh) is higher than that of the QPBO result, and thus does not represent the global minimizer. The number of wrong normals was calculated with respect to the ground truth data set.

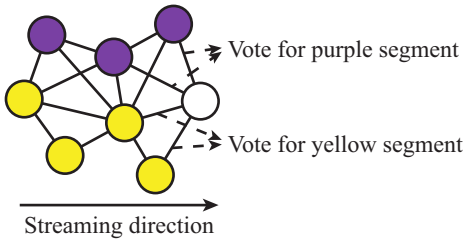
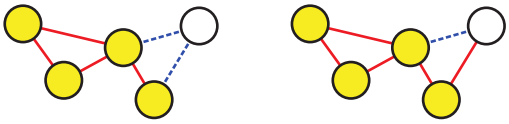


Figure 4: Visualization of candidate segments for the white (unsegmented) point. Segments are color-coded (purple and yellow). Both segments are candidates (represented by two neighbors each).

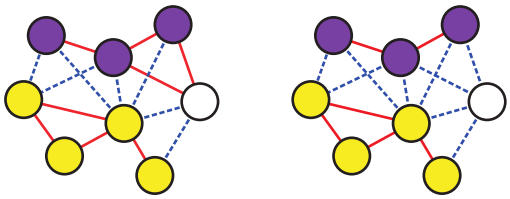


(a) Criterion fulfilled (b) Criterion not fulfilled

Figure 5: Example graph for the intra-segment segmentation criterion for the white node. Red edges visualize positive edge weights, blue edges visualize negative values. All nodes belong to the candidate segment.

The segment candidate search is based on the current point's neighbors that are already assigned to a segment. The set of all neighbors' segments is the preliminary set of candidate segments for the current point. This set is reduced further to avoid interleaving segments. Only the closest neighbors (empirically, neighbors closer than 130% of the closest neighbor's distance yield good results) are considered.

The minimizer-preserving properties of the candidate segments are then evaluated based on our segmentation heuristic. This heuristic consists of two criteria, which must be fulfilled by a segment. If a segment does not fulfill both criteria, it is removed from the candidate list. The heuristic cannot guarantee the preservation of minimizers. However, it can be evaluated efficiently and yielded good results in our experiments.



(a) Criterion fulfilled (b) Criterion not fulfilled

Figure 6: Example graph for the inter-segment segmentation criterion for the white node and the yellow segment. Nodes are colored by their segment.

The **intra-segment criterion** constrains neighbor votes within the candidate segment. The potentials of the according edges cannot be changed when the segment is established. Consequently, it must be possible to find an orientation for the processed point that yields a zero potential for all those edges and any segment orientation. This is the case iff all votes have the same sign. To reduce the number of segments, we allow contradicting edges as long as their sum does not exceed θ_S. Figure 5 shows two examples of a fulfilled and an unfulfilled intra-segment criterion. Additionally, the segment vote's absolute value must be greater than the threshold θ_{acc}. We assume that the local orientation of the point's normal cannot be deduced reliably from segments, where this is not the case. The formal definition of the criterion can be found in the supplementary material.

The **inter-segment criterion** constrains neighbor votes that pass a segment border. In order to preserve the energy's minimizer, all edges between two segments should have the same sign. Therefore, the processed point can only be added to a segment if its edges to points of the neighboring segment have the same sign as all other points of the candidate seg-

ment, taking into account possible preliminary flips during the segmentation process (cf. figure 6). We allow contradicting edges whose sum does not exceed θ_S.

Together with the distance constraint, these criteria define a segment's validity. A segment is valid iff it is closer than 130 % of the closest neighbor's distance, if it fulfills the intra-segment criterion, and if it fulfills the inter-segment criterion. From the set of valid segment candidates, the one with the greatest absolute vote is chosen as the segment for the current point because it allows the most certain orientation. If this vote is negative, the normal is flipped in order to create consistent patches. If no valid segment exists, a new segment is created.

5.2. MRF Generation and Global Orientation

Given the segmentation of the input points, the original neighbor graph is reduced by contracting all nodes of a segment to a single node. Consequently, edges that do not cross segment borders will be removed. Multiple edges between two segments are merged into a single edge whose edge weight (distance-weighted output of ϕ) is given by the sum of contributing edge weights. This results in a segment graph whose edges describe the orientation relation of two neighboring segments.

The segment graph generated by contracting nodes is not a k-nearest neighbor graph, but an adjacency graph of patches, which can be regarded as another form of neighbor graphs. With the energy definition from (2), this graph can be transformed into an MRF, whose minimizer is the optimal segment orientation. Because this MRF is also an instance of the orientation problem, the solvers presented in section 4 are suited equally well for this minimization problem. The final orientation can be found by multiplying every point's normal with their respective segment label.

5.3. Streaming Processing

Additionally to segmentation, we use Pajarola's out-of-core streaming approach [Paj05] as explained in section 2.4. However, instead of the completion of the k-neighborhood, we employ the maximum neighbor search radius r to determine the points that can be released from main memory (which are then all points that are farther than r behind the sweeping plane). In this section, we outline the differences to the in-core variant explained before.

During the streaming process, every iteration reads a new chunk of data into main memory. We refer to this piece of data as the read chunk. This chunk is added to the set of active data and then processed.

The streaming approach induces a natural order, in which points are assigned to a segment (according to the stream order). All points with a smaller x-coordinate than the current point have been processed earlier and thus already have a segment. Therefore, the candidate segments are based on the neighbors with a smaller x-coordinate. These neighbors are approximated with a knn-search with respect to the current point. The contraction procedure can be executed on-the-fly while streaming the point cloud. Therefore, it is necessary to store the neighbor graph only for points in the read chunk. Figure 7 visualizes the segmentation result in combination with the streaming processing.

(a) (b)

Figure 7: Shaded visualization of the segmentation result for Xie's flip criterion. Points of the same segment have the same color. $\theta_S = 0; \theta_{acc} = 0.5$; 34 segments have been created. a: Axis-aligned view, streaming direction from left to right. b: Different view point.

When points are released from active memory, their position, preliminary normal, and segment index are written to a temporary file. After optimization, the points are re-streamed from the temporary file and their segment's sign is applied to their normals. Additionally, we chose to group data into connected components by writing them to per-component files. This allows an easy filtering of small components which are likely to be caused by noise.

Algorithm 1 outlines the major steps of the orientation process.

5.4. Additional Parameters

The streaming approach requires two additional parameters, which are used during evaluation of the segmentation criteria: θ_S and θ_{acc}. In this section, we review their significance.

The threshold $\theta_S \in \mathbb{R}_{\geq 0}$ defines the tolerance for contradicting segment votes (either intra-segment or inter-segment). Contradicting votes with a low certainty are often generated by noise or shortcomings of the flip criterion. To prevent this from generating very many small segments, θ_S can be increased, which generally results in fewer segments. This reduces the optimization domain and with that the execution time of the optimization. Although this thresholding

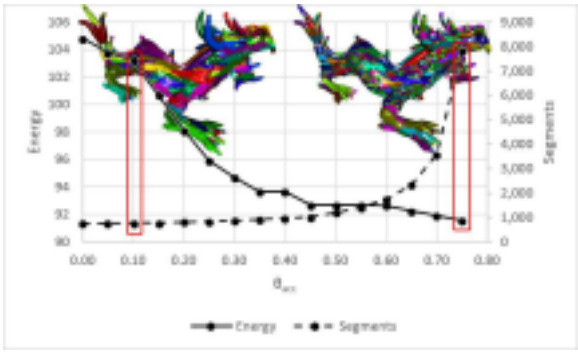
Algorithm 1 Streaming Normal Orientation

```
1:  function SOLVE(fileIn, fileOut)
2:      fileTemp ← temporary file; mrf ← new MRF
3:      while readChunk ← read data from fileIn do
4:          for all v ∈ readChunk do                    ▷ in parallel
5:              N_v ← knn(v)                            ▷ with a smaller index
6:          for all v ∈ readChunk do
7:              SEGMENT(v, N_v, fileTemp, mrf)
8:          clean active memory
9:      L ← solve mrf
10:     while readChunk ← read data from fileTemp do
11:         for all (v, s) ∈ readChunk do
12:             normal_v ← normal_v · L_s
13:             Write v to fileOut_connected component of v
14: function SEGMENT(v, N_v, fileTemp, mrf)
15:     votes ← per-segment votes from N_v
16:     bestSeg ← ∅
17:     for all s ∈ votes do                            ▷ iterate segments
18:         if s is valid ∧ |votes_s| > |votes_bestSeg| then
19:             bestSeg ← s
20:     if bestSeg = ∅ then
21:         bestSeg ← create new segment in mrf
22:     else if votes_bestSeg < 0 then
23:         flip normal and incident edges
24:     for all s ∈ votes \ bestSeg do
25:         mrf.addEdge(bestSeg, s)
26:     write (v, bestSeg) to fileTemp
```

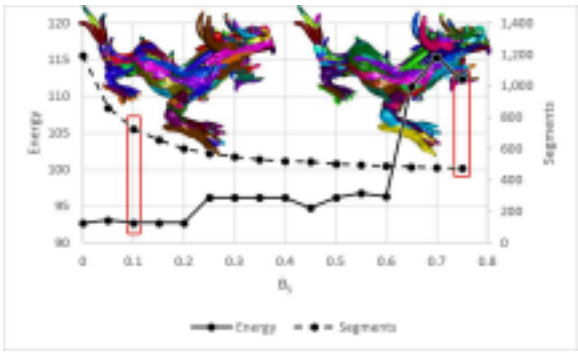
introduces a bias into the energy, we assume that the minimizers are preserved because the votes have only little impact on the energy due to their small certainty. If, however, this bias is not desired, θ_S can be set to 0.

The threshold $\theta_{acc} \in \mathbb{R}_{\geq 0}$ defines a minimum certainty of segment votes. Increasing θ_{acc} generally increases the number of segments. This can be desirable if the preliminary orientation during segmentation cannot find a reasonable orientation, thus leaving the orientation to global optimization. Therefore, a moderate choice of θ_{acc} helps to preserve the minimizers.

The range of both thresholds is mainly defined by the flip criterion. The criteria presented in this paper yield values in the range $[-1, 1]$ for unit normals. Therefore, small thresholds (usually smaller than 1) are preferable. Figure 8 visualizes how changing the parameters affects the optimization result. In these charts, the final energy is calculated on the original MRF instead of the reduced segment graph. It can be seen that the energy decreases (i.e. orientation quality increases) as the number of segments increases, which is caused by a larger number of degrees of freedom for the optimizer. Note how large θ_{acc} produces more small segments. This is even amplified by decreasing θ_S. The best parameter configuration depends on the concrete scenario. In general,



(a) Varying θ_{acc} and constant $\theta_S = 0$



(b) Varying θ_S and constant $\theta_{acc} = 0.5$

Figure 8: Impact of segmentation parameters on the number of segments and the resulting orientation quality for the dragon data set, Xie's flip criterion, and the MST+QPBO-I solver. The energy has been calculated on the unreduced neighbor graph.

the number of segments should be chosen as small as possible without having a strong negative impact on the energy.

5.5. Implementation Details

The streaming orientation requires acceleration data structures that allow fast neighbor queries. Since the active data set has a very small extent in the x-direction (the maximum neighbor search radius), it is sufficient to build spatial structures in the two other dimensions. Additionally, the chosen structure must support parallel read and write operations of points.

Our implementation uses a data structure based on a sparse 2d grid. We found that a cell size equal to the maximum neighbor search radius performs best, which is obviously a good compromise between the number of searched cells and the number of points within a cell. The cells contain a temporary list for newly added points and a kd-tree, which accelerates neighbor searches within the cell (a single cell may contain several thousand points). New points from

the read chunk are first inserted into the list, which is significantly faster than inserting them into the kd-tree. They are removed from the list and inserted into the kd-tree when their first neighbor query is executed (the first path taken by the knn-query is equal to the path taken by insertion). Because neighbor queries are parallelized, all relevant points must be inserted into the structure before the query can be executed, which motivates the usage of a quickly modifiable structure such as the temporary list. Neighbor queries include both the kd-tree and the temporary list. They return only neighbors with a smaller index.

The neighbor search is executed in parallel for all points in the read chunk. This process is protected by a cell-level lock strategy that allows multiple concurrent reads from the same cell. Modifying access of a cell is protected by an exclusive lock. An exclusive lock is acquired when a point is removed from the temporary set and is released once it is inserted in the according kd-tree. This two-level approach allows a high level of parallelism, especially if the points of the read chunk lie in separate grid cells.

For numerical optimization of the MRF, we rely on the implementation provided by OpenGM [ATK12].

Our implementation is available at `http://github.com/NSchertler/StreamingNormalOrientation`.

6. Results

We tested the presented algorithms with both real-world 3D scans and re-sampled polygonal meshes. Before executing the orientation algorithm, we randomized each point's normal orientation. The following section gives an overview of the test results. First, the achieved quality of the different solvers is compared, followed by an analysis of run time and memory consumption.

6.1. Quality

By construction, the advanced solvers always generate results that are at least as good as those of the traditional MST solver (in terms of energy). As shown in figure 9 and table 1, using QPBO is especially valuable when the MST solver fails for isolated parts of the data set due to fine details, which result in contradicting edges in the neighbor graph (e.g. parts of the foot in the Ramses data set or the right elephant's ear and parts near the left woman's hand in the Thai Statue data set). In practice, QPBO-I performs at least as good as LSA-TR or slightly better for all of our test data sets (with respect to both run time and quality). Therefore, QPBO-I seems to be the better choice for the orientation problem. Although combinatorial solvers like Multicut [KSA*11, KSRS13] can find global optima, their long run time and high memory consumption make them unsuitable for most data sets.

without streaming	Dragon	Ramses	Thai Statue
Flip Criterion	Xie	Xie	Hoppe
Number of points	28,885	60,910	723,583
E. of MST Solution	121.9	79.7	3,296.5
MST + QPBO-I	91.0	30.1	2,689.3
Energy Reduction	25.3 %	62.2 %	18.4 %
Overall Orientation	716 ms	1,020 ms	118.0 s
Neighbor Search	85 ms	214 ms	3.4 s
Solve MST	24 ms	54 ms	0.7 s
Solve QPBO	581 ms	691 ms	113.0 s
Energy (LSA)	94.7	44.9	2729.12
Optimization (LSA)	8.1 s	18.9 s	49.7 min
with streaming	**Fig. 10a**	**Fig. 10b**	**Fig. 10c**
Flip Criterion	Hoppe	Hoppe	Hoppe
Number of points	50.5 M	186,8 M	260.8 M
Generated Segments	52,322	64,628	860,966
Overall Orientation	11.7 min	15.9 min	70.6 min
Neighbor Search	6.2 min	3.6 min	35.6 min
Segmentation	1.0 min	3.3 min	6.1 min
Optimization	1.0 s	0.3 min	0.8 min
Finalization	0.4 min	0.9 min	5.7 min

Table 1: Statistics of the orientation process using a knn graph. $k = 6, \theta_S = 0$, $\theta_{acc} = 0.5$; r has been chosen manually based on average point density. Intel Core i7 (3.4 GHz). Portions of the total time that are not accounted for in any of the sub steps are I/O and minor management steps.

Like the simple orientation approach, the streaming version heavily relies on a valid flip criterion. Once this condition is met, a reliable segmentation can be calculated and the reduced orientation problem can be solved with similar quality as the original problem. Due to the approximate nature of the reduced energy, streaming may result in a slightly worse orientation. E.g. the dragon data set results in an energy of 92.7 when oriented with the streaming approach (energy evaluated on the original neighbor graph) with four points oriented differently than the unstreamed orientation ($E = 91.0$). Figure 10 shows the results of streamed orientation of some large data sets.

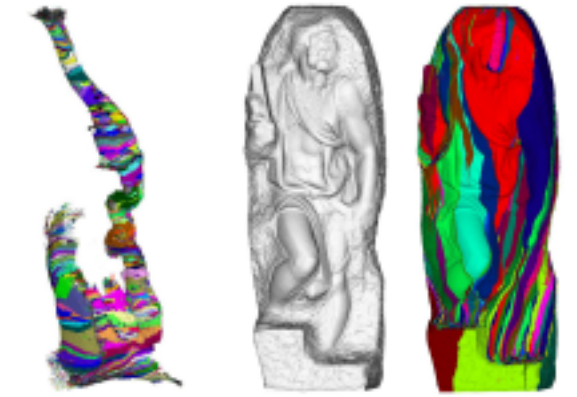
6.2. Performance

In order to evaluate the algorithm's run time, we re-sampled the dragon data set with different numbers of points. This approach allows us to compare differently sized data sets with nearly the same complexity. This generates a similar number of segments, which is essential for a fair comparison of the streaming approach.

Figure 11 compares the performance of the direct and the streaming approach. In the direct approach, most time (beyond linear) is consumed by the QPBO-I step. The peak memory consumption scales linearly and is mainly caused

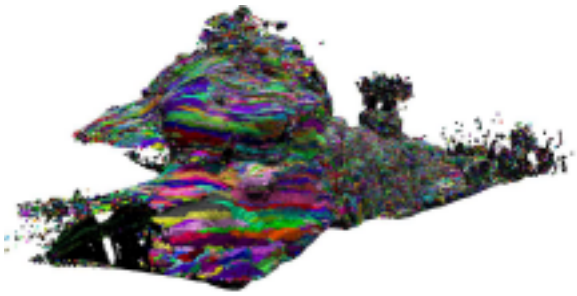
(a) Ramses - MST (b) Ramses - MST + QPBO-I (c) Thai Statue - MST (d) Thai Statue - MST + QPBO-I

Figure 9: Orientation results for two more data sets using the MST solver and MST + QPBO-I



(a) Gomantong side cave. Streaming direction from left to right.

(b) St. Matthew. Streaming direction from bottom to top.



(c) Gomantong caves. Streaming direction from left (foreground) to right (background). The black areas at the left are the cave's correctly oriented inside. The correctness has been validated visually.

Figure 10: Results of streamed orientation of large data sets.

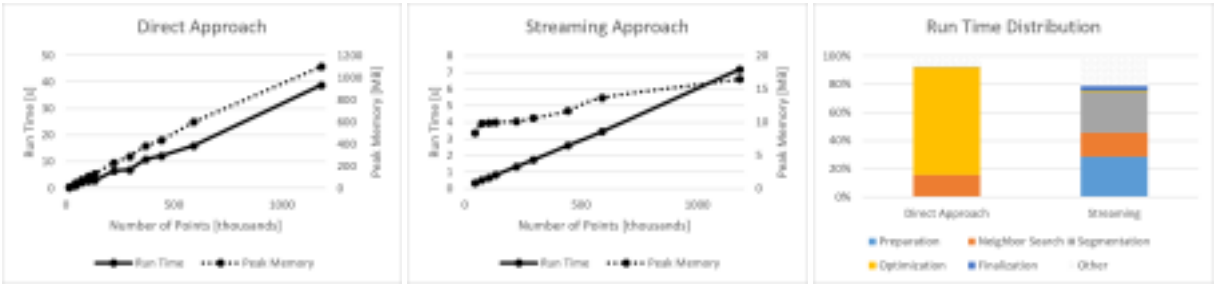
by the QPBO solver. The streaming approach greatly reduces the optimization domain. Due to this, the time needed to solve the MRF is neglectable compared to the other steps. Another effect of the segmentation is the reduced peak memory consumption, which stays almost constant and is only influenced by the number of generated segments. If data is already available in the streaming format, the preparation step, which sorts the points along the x-axis, can be skipped, reducing the total run time even further. Even if only the fast MST solver is used, the streaming approach is superior to the direct approach (in terms of performance) as soon as the overhead of preparation is overcome (which is the case at about 130,000 points in this experiment). The MST solver is equivalent to Hoppe's approach, which is the fastest of the presented propagation-based methods [KG09, HLZ*09, SBY11]. Therefore, our streaming approach is faster than any of the in-core approaches.

6.3. Limitations

Our method depends strongly on the chosen flip criterion. Therefore, we can achieve only results as good as the flip criteria allow. Especially, denser data sets lead to more robust flip criteria. What flip criterion should be used, depends on the characteristics of the data set. If there are a lot of sharp creases, Xie's criterion should be used. Hoppe's criterion is preferable for mainly smooth data sets. Furthermore, Hoppe's criterion is more robust against noise.

An orientation process that updates normal directions was presented in [HLZ*09]. Since our optimization approach relies on fixed edge weights, we cannot update normal directions easily. However, an alternating approach is imaginable.

The orientation approach itself is not affected by noise (as long as the neighborhood does not change too much). However, this may not be true for normal estimation, which in



(a) Without streaming (b) With streaming, $\theta_S = 0$, $\theta_{acc} = 0.5$ (c) Relative Run Time Distribution

Figure 11: Comparison of run time and peak memory consumption of the streamed and unstreamed version of the algorithm for different sizes of the dragon data set and the MST+QPBO-I solver.

turn affects the orientation process. Therefore, our orientation approach inherits the robustness against noise from the normal estimation method and the flip criterion. Outliers do not have a negative impact on normal orientation in general, as long as the normal estimation of inliers is not affected. This is due to the flip criterion's distance-weighting, which assigns a very small certainty to edges from outliers.

7. Conclusion and Future Work

In this article, we have consolidated existing propagation-based normal orientation approaches into a single energy minimization model. Several ways to solve this problem have been presented, concluding that the traditional MST approach can be improved with non-greedy state-of-the-art solvers, which are widely applied in the domain of Computer Vision. Furthermore, we presented a segmentation-based framework for orienting large data sets, which reduces the problem and then uses the same model and solvers like the original approach.

A main problem, which became obvious, is the choice of a reasonable flip criterion (see Hoppe's flip criterion for the dragon data set, figure 3b). Although one of both criteria works for most data sets, there are point clouds which cannot be solved with either one (ground truth energy is not a global minimizer). With the definition of an MRF, it is possible to define flip criteria not only on a pair-wise basis but in a larger neighborhood, which can make the flip criterion more robust. However, solving those higher-order MRFs is quite difficult and still subject to current research [AFG08, FGBZ11, OV12, FWZ14].

Although QPBO-I yielded good results in our experiments, other data sets may require different solvers and possibly solver-specific adaptations of the problem definition. Most notably, Cut-Glue-Cut [BKK*14] and CC-Fusion [BHK15] seem to be viable candidates.

In the supplementary material, we outlined that QPBO-I needs to fix a subset of orientations in order to calculate a complete labeling. This suggests a semi-automatic, user-guided orientation process, where the subset of fixed nodes is defined by the user. This avoids that nodes with wrong orientations are fixed, which inevitably leads to a wrong orientation of other nodes.

Some measures to counter data set noise that have been presented in other papers can be incorporated with our approach. For example, the flip criteria could be evaluated on a denoised or smoothed version (cf. [HLZ*09]).

8. Acknowledgments

We thank the respective authors for providing the Dragon and Thai Statue data (Stanford 3D Scanning Repository), the St. Matthew data (Digital Michelangelo Project), the Pegasus and Ramses model (AIM@SHAPE-VISIONAIR Shape Repository), and the Gomantong data (3D cave mapping group, Institute of Cartography, TU Dresden (Manfred Buchroithner), International Union of Speleology (Donald McFarlane)). Data provision is thankfully acknowledged.

This work is partially funded by the European Social Fund and the Free State of Saxony (ESF project number 100226943, "ADFEX").

References

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry processing* (2007), vol. 7, pp. 39–48. 2

[AFG08] ALI A., FARAG A., GIMEL'FARB G.: Optimizing binary MRFs with higher order cliques. In *Computer Vision, ECCV 2008*, Forsyth D., Torr P., Zisserman A., (Eds.), vol. 5304 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 98–111. doi:10.1007/978-3-540-88690-7_8. 11

[ATK12] ANDRES B., T. B., KAPPES J. H.: OpenGM: A C++ library for discrete graphical models. *ArXiv e-prints* (2012). URL: http://arxiv.org/abs/1206.0111, arXiv:1206.0111. 9

[BHK15] BEIER T., HAMPRECHT F. A., KAPPES J. H.: Fusion moves for correlation clustering. In *IEEE Conference on Computer Vision an Pattern Recognition (CVPR)* (2015). 11

[BHS91] BOROS E., HAMMER P., SUN X.: *Network flows and minimization of quadratic pseudo-boolean functions*. Tech. rep., Technical Report RRR 17-1991, RUTCOR, 1991. 4

[BKK*14] BEIER T., KRÖGER T., KAPPES J. H., ET AL.: Cut, glue & cut: A fast, approximate solver for multicut partitioning. In *CVPR. Proceedings, in press* (2014). 11

[BTS*14] BERGER M., TAGLIASACCHI A., SEVERSKY L. M., ET AL.: State of the Art in Surface Reconstruction from Point Clouds. In *Eurographics 2014 - State of the Art Reports* (2014), Lefebvre S., Spagnuolo M., (Eds.), The Eurographics Association. doi:10.2312/egst.20141040. 2

[BZK04] BORODIN P., ZACHMANN G., KLEIN R.: Consistent normal orientation for polygonal meshes. In *Computer Graphics International 2004 (CGI 2004)* (June 2004), IEEE Computer Society, pp. 18–25. 2

[FGBZ11] FIX A., GRUBER A., BOROS E., ZABIH R.: A graph cut algorithm for higher-order Markov random fields. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 1020–1027. 11

[FWZ14] FIX A., WANG C., ZABIH R.: A primal-dual algorithm for higher-order multilabel markov random fields. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on* (2014), IEEE, pp. 1138–1145. 11

[GBV*14] GORELICK L., BOYKOV Y., VEKSLER O., ET AL.: Submodularization for binary pairwise energies. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on* (June 2014), pp. 1154–1161. doi:10.1109/CVPR.2014.151. 5

[GCSA13] GIRAUDOT S., COHEN-STEINER D., ALLIEZ P.: Noise-Adaptive Shape Reconstruction from Raw Point Sets. *Computer Graphics Forum 32*, 5 (2013), 229–238. doi:10.1111/cgf.12189. 2

[GPS12] GONG Y., PAUL G., SBALZARINI I.: Coupled signed-distance functions for implicit surface reconstruction. In *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on* (May 2012), pp. 1000–1003. doi:10.1109/ISBI.2012.6235726. 2

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., ET AL.: Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), SIGGRAPH '92, ACM, pp. 71–78. doi:10.1145/133994.134011. 2

[HLZ*09] HUANG H., LI D., ZHANG H., ET AL.: Consolidation of unorganized point clouds for surface reconstruction. In *ACM SIGGRAPH Asia 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH Asia '09, ACM, pp. 176:1–176:7. doi:10.1145/1661412.1618522. 2, 10, 11

[ILSS06] ISENBURG M., LIU Y., SHEWCHUK J., SNOEYINK J.: Streaming computation of delaunay triangulations. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 1049–1056. doi:10.1145/1179352.1141992. 3

[KAH*15] KAPPES J. H., ANDRES B., HAMPRECHT F. A., ET AL.: A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision* (2015), 1–30. doi:10.1007/s11263-015-0809-x. 4

[KG09] KÖNIG S., GUMHOLD S.: Consistent propagation of normal orientations in point clouds. In *VMV* (2009), pp. 83–92. 2, 10

[KSA*11] KAPPES J. H., SPETH M., ANDRES B., ET AL.: Globally optimal image partitioning by multicuts. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, Boykov Y., Kahl F., Lempitsky V., Schmidt F., (Eds.), vol. 6819 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pp. 31–44. doi:10.1007/978-3-642-23094-3_3. 9

[KSRS13] KAPPES J. H., SPETH M., REINELT G., SCHNÖRR C.: Higher-order segmentation via multicuts. *arXiv:1305.6387* (2013). 9

[KTB07] KATZ S., TAL A., BASRI R.: Direct visibility of point sets. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. doi:10.1145/1275808.1276407. 2

[KZ04] KOLMOGOROV V., ZABIN R.: What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 2 (Feb 2004), 147–159. doi:10.1109/TPAMI.2004.1262177. 4

[LRB07] LEMPITSKY V., ROTHER C., BLAKE A.: Logcut - efficient graph cut optimization for markov random fields. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (Oct 2007), pp. 1–8. doi:10.1109/ICCV.2007.4408907. 4

[LW10] LIU S., WANG C. C.: Orienting unorganized points for surface reconstruction. *Computers & Graphics 34*, 3 (2010), 209 – 218. Shape Modelling International (SMI) Conference 2010. doi:10.1016/j.cag.2010.03.003. 2

[MDGD*10] MULLEN P., DE GOES F., DESBRUN M., ET AL.: Signing the unsigned: Robust surface reconstruction from raw pointsets. *Computer Graphics Forum 29*, 5 (2010), 1733–1741. doi:10.1111/j.1467-8659.2010.01782.x. 2

[OV12] OSOKIN A., VETROV D.: Submodular relaxation for MRFs with high-order potentials. In *Computer Vision–ECCV 2012. Workshops and Demonstrations* (2012), Springer, pp. 305–314. 11

[Paj05] PAJAROLA R.: Stream-processing points. In *Visualization, 2005. VIS 05. IEEE* (Oct 2005), pp. 239–246. doi:10.1109/VISUAL.2005.1532801. 3, 7

[Pea01] PEARSON K.: LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6 2*, 11 (1901), 559–572. doi:10.1080/14786440109462720. 1

[RKLS07] ROTHER C., KOLMOGOROV V., LEMPITSKY V., SZUMMER M.: Optimizing binary MRFs via extended roof duality. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (2007), IEEE, pp. 1–8. 4

[SBY11] SEVERSKY L. M., BERGER M. S., YIN L.: Harmonic point cloud orientation. *Computers & Graphics 35*, 3 (2011), 492 – 499. Shape Modeling International (SMI) Conference 2011. doi:10.1016/j.cag.2011.03.012. 2, 10

[WYC12] WANG J., YANG Z., CHEN F.: A variational model for normal computation of point clouds. *The Visual Computer 28*, 2 (2012), 163–174. doi:10.1007/s00371-011-0607-6. 2

[XWH*03] XIE H., WANG J., HUA J., ET AL.: Piecewise c1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. In *Visualization, 2003. VIS 2003. IEEE* (Oct 2003), pp. 91–98. doi:10.1109/VISUAL.2003.1250359. 2

Supplementary Material: Towards Globally Optimal Normal Orientations for Large Point Clouds

Nico Schertler, Bogdan Savchynskyy, and Stefan Gumhold

TU Dresden, Germany

Abstract

This supplementary material gives more information about Xie's flip criterion and the Signed Union Find data structure. Additionally the two segmentation criteria are defined formally.

1. Xie's flip criterion

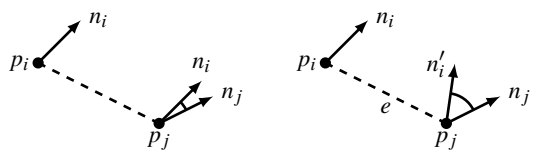
Xie's flip criterion can handle sharp creases and close surface sheets by reflecting one normal along the direction vector between the two points (see figure 1 for details). The criterion can be expressed as:

$$\begin{aligned} \phi_{Xie}(i,j) &= \langle n_i', n_j \rangle \\ n_i' &= n_i - 2e\langle e, n_i \rangle \\ e &= \frac{p_i - p_j}{\|p_i - p_j\|} \end{aligned} \tag{1}$$

2. Signed Union Find for MST Solutions

In our paper, we use the Signed Union Find data structure to find the Maximum Spanning Tree solution for an orientation problem. In this section, we explain the details of this data structure.

Hoppe's original algorithm [HDD*92] first calculates the spanning tree and in a second step performs the propagation. This requires traversing the graph twice (once completely and once the spanning tree) as well as storing the spanning tree explicitly. Our improved version merges both steps into a single traversal, calculating the spanning tree on-the-fly without explicitly storing it.



(a) Hoppe's flip criterion (b) Xie's flip criterion

Figure 1: Visualization of Hoppe's and Xie's flip criterion.

The basis of this improvement is Kruskal's minimum spanning tree algorithm [Kru56], which is usually implemented using a Union-Find data structure [Knu69]. A possible application of normal orientation using Kruskal's algorithm can be found in [XD11].

The Union-Find data structure is a forest of rooted trees where each entry corresponds to a node in the graph. Each node maintains a pointer to its parent node. The entire structure is implemented with a list of indices. Initially, each node is a separate tree. A connected component's representative (operation *find*) can be found by following the path of parent pointers up to the root. Two entries belong two the same connected component iff they share the same representative. Merging two connected components (operation *union*) is achieved by updating the parent pointer of one component's root to point to the other component's root. Due to some acceleration techniques like path compression, both operations can be executed in effectively constant time (more precisely, it grows very slowly in order of the inverse Ackermann function).

We augment this structure with a sign bit s_i for each node ($0 \equiv +$, $1 \equiv -$) and refer to it as *Signed Union-Find*. The idea is to enable quick sign flips for entire connected components, which is achieved by using the sign bits on the path from a node to its root as XOR summands of the node's actual sign:

$$sign(i) = \bigoplus_{j \in \text{path from i to root}} s_j \tag{2}$$

Path compression will ensure that this chained XOR is kept short, enabling effectively constant time complexity.

If the sign of an entire connected component must be changed (method *flipConnectedComponentSign* in algorithm 1), it is sufficient to flip the sign bit of the component's root (also effectively constant time). Since this bit is included in the sign formulas for every child node, this essentially flips the signs of all nodes within this connected component.

Care must be taken when uniting two root nodes. Without loss of generality we assume that node i will be the new root for node j. Then, in order to preserve the sign in the connected component of node j, the sign bit has to be updated with $s_j \leftarrow s_i \oplus s_j$. This works because $\oplus$ is its own inverse operation.

Path compression has to adapt the sign bits in a similar way. If node i with path to its root consisting of nodes $j,\dots,q,r$ (node r being the root node) is re-linked to be a direct child of its root r, the sign calculation of node i changes from $s_i \oplus s_j \oplus \dots \oplus s_q \oplus s_r$ to $s_i \oplus s_r$. It is obvious that the update must be $s_i \leftarrow s_i \oplus s_j \oplus \dots \oplus s_q$. An entire path can be compressed (i.e. re-linking all nodes to the root) in $\mathcal{O}(k)$ time, where k denotes the number of nodes on that path.

This Signed Union-Find data structure enables on-the-fly computation of the spanning tree. Algorithm 1 shows its application to solving the orientation problem.

Algorithm 1 MST Solver Using Signed Union-Find

```
1:  function SOLVE(P, E)        ▷ Calculates optimal labeling
2:      uf ← init Signed Union-Find with |P| nodes
3:      sort edges in descending order with respect to |ϕ|
4:      for {i, j} ∈ E do
5:          r_i ← uf.find(i)
6:          r_j ← uf.find(j)
7:          if r_i ≠ r_j then
8:              diffSigns ← uf.getSign(i) ≠ uf.getSign(j)
9:              if diffSigns ⊕ (ϕ(i, j) < 0) then
10:                 uf.flipConnectedComponentSign(i)
11:             uf.merge(i, j)
12:     return signs of uf
```

Although this algorithm does not directly propagate orientations along edges, the general idea is the same as before. One difference though is the absence of a starting point with known orientation. If desired, the orientation for a single point (or more non-contradicting points) can be forced after the execution of the algorithm by checking the according sign and flipping the component if necessary.

Figure 2 compares the performance of the MRF solver with the Signed Union Find data structure and with our implementation of the traditional one. Both data structures scale nearly linearly, while the Signed Union Find is in average 2.2 times as fast as the traditional data structure because it avoids the second traversal.

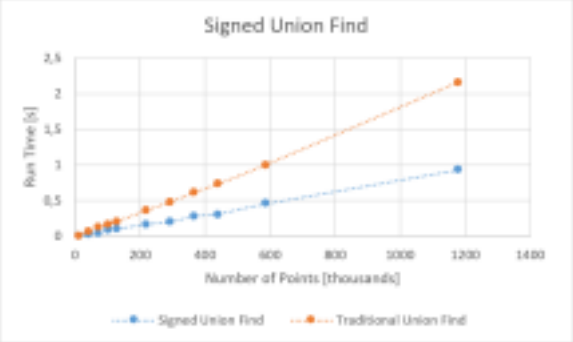


Figure 2: Run time comparison of the MST solver with the traditional Union Find data structure and our Signed Union Find data structure for different sizes of the dragon data set.

3. Segmentation Criteria

The segmentation criteria are used to evaluate if a segment can be assigned to a point. Given the neighbor vote of two neighboring points $vote(p_1,p_2)$, we derive the segment vote of a point p with respect to segment s as

$$vote(p,s) := \sum_{q\in s} vote(p,q), \tag{3}$$

where $vote(p,q)$ is zero for non-neighboring points. Similarly, the vote between two segments s_1 and s_2 is

$$vote(s_1,s_2) := \sum_{p\in s_1}\sum_{q\in s_2} vote(p,q) \tag{4}$$

The intra-segment criterion is defined as follows: The assignment of segment s to point p is valid iff, given the set of considered neighbor points $\mathcal{N}^{pt}(p,s)$ of point p from segment s, i.e. all neighbors that are already assigned to this segment:

$$\left(\sum_{n\in\mathcal{N}^{pt}(p,s)} vote_{>0}(p,n) \le \theta_S \;\vee\; -\sum_{n\in\mathcal{N}^{pt}(p,s)} vote_{<0}(p,n) \le \theta_S\right) \wedge |vote(p,s)| \ge \theta_{acc} \tag{5}$$

The symbols $vote_{>0}(p_1,p_2)$ and $vote_{<0}(p_1,p_2)$ denote the positive and negative part of the scalar neighbor vote between two neighboring points p_1 and p_2. I.e.

$$vote_{>0}(p_1,p_2) = \begin{cases} vote(p_1,p_2) & \text{if } vote(p_1,p_2) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The inter-segment criterion specifies that a segment is valid iff

$$\forall t \in \mathcal{N}^{seg}(p) \setminus s : \\ (sgn(vote(s,t)) = sgn(vote(p,t)) \cdot flip) \vee |vote(p,t)| < \theta_S, \tag{7}$$

where $\mathcal{N}^{seg}(p)$ denotes the set of neighbor segments of point p and $flip$ is the flip decision $= sgn(vote(p,s))$.

References

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., ET AL.: Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), SIGGRAPH '92, ACM, pp. 71–78. doi:10.1145/133994.134011. 1

[Knu69] KNUTH D. E.: *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. Reading, Mass.: Addison-Wesley, 1969. 1

[Kru56] KRUSKAL J. B.: On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society 7*, 1 (1956), 48–50. 1

[XD11] XI Y., DUAN Y.: A new integrated depth fusion algorithm for multi-view stereo. *Computer Graphics International* (2011). 1