

Resolving Twisted Surfaces within an Iterative Refinement Surface Reconstruction Approach

Hendrik Annuth and Christian-A. Bohn

Wedel University of Applied Sciences, Wedel, FR Germany

Abstract

We present a method which resolves twisted surface regions within a surface reconstruction approach that uses local refinement operations to iteratively fit a surface into an unorganized point cloud. We show that this local operation can be integrated reliably and efficiently, although resolving twisted surfaces is not a local operation since it may cause modifications up to one half of the entire surface. We introduce a novel data structure called the minimal edge front that enables efficiently retrieving topological information from the surface under investigation. Equipped with this operation the algorithm is able to robustly handle huge point-clouds of complex closed and also not closed objects like landscapes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; Geometric algorithms, languages, and systems; I.2 [Numerical Analysis]: Approximation—Approximation of surfaces and contours

Keywords: surface reconstruction, point-cloud, iterative refinement, growing cell structures

1. Introduction

Reconstruction of surfaces from unorganized point-cloud is a well known problem. For most practical computer graphics applications triangular meshes need to be oriented and fulfill the manifold criterion. These properties are essential for many mesh processing applications and can be used to optimize the rendering process.

Many common surface reconstruction algorithms require data points which are augmented with normal vectors which explicitly define the surface orientation [KBH06, OBA*03, ABCO*01, CBC*01]. In these cases normals are either estimated or derived from certain technical conditions. For example, in [SSZCO10] the direction to the scanner head from each sample point is utilized. Normal estimation is realized in [ABK98] by the determination of the farthest vertices of the Voronoi cell of a point, the poles. Sensibility to noise and non-uniform sample densities is accounted for in [MAVdF05].

In [HDD*92] a prominent normal estimation mechanism is described. With the k nearest neighbors to a data point p the plane with p as its pivot element and with the lowest squared distance to these k points is assumed to be the

tangent plane of p . The problem of this method is to find a suitable value for k which is, on the one hand, big enough to deal efficiently with noise and, on the other hand, small enough to include only points which are geometrically relevant concerning p . After the estimation of the normal vectors they need to be oriented. For densely scanned volumetric objects this can reliably be accomplished by defining it based on the inner and outer poles of a vertex [ABK98]. The assumption of having a volumetric object is made in many reconstruction algorithms [SSZCO10, KBH06, HK06, SLS*06, OBA*03, CL96]. These approaches, on the one hand, are able to deal with incomplete data and thus generate watertight surfaces. On the other hand, this is a serious limitation in case of, i.e., landscapes or partially retrieved objects which are not supposed to be watertight.

For non-solid objects or noisy and non-uniform sample data the orientation of the estimated normals need to be consistently propagated through the point-cloud. In [HDD*92] the orientation is propagated from one initial point over the edges of the minimal spanning tree of the absolute dot products of neighboring point normals. This causes the propagated to be done across the most parallel normals. The general problem of calculating normals on a local point level

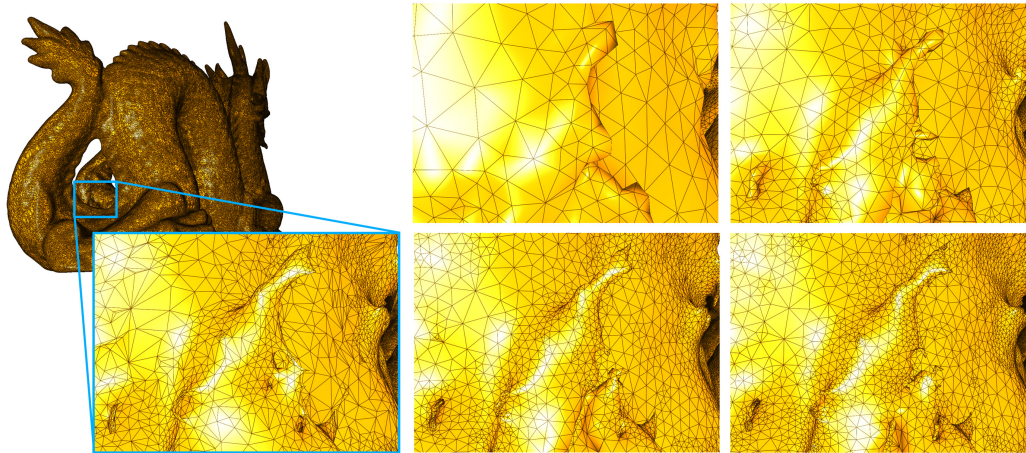


Figure 1: An iterative refinement approach for surface reconstruction. Challenging regions are easily handled through the incremental adaptation of solutions at successive stages. See the region at the dragon's leg (left) and the improvement levels of the reconstruction process (right).

is the assumption that a local point subset exists that allows for drawing a conclusion about the unknown global surface. This assumption does not hold in case of noise, non-uniform sample densities, outliers, corners, creases — generally for insufficiently sampled thin or tiny structures [LCOLTE07]. In [HLZ*09] principal component analysis is used to estimate normals after the point-cloud is released from noise, outliers are removed, and the sample distribution is thinned out in order to achieve evenly distributed point samples.

Region growing approaches [GK02, BMR*99] propagate an initial orientation thru the point-cloud. This propagation often fails if the orientation is not locally recognisable. Some approaches operate on non-manifold, non-oriented meshes [DRADLN10, CLK09, HF08] and fix the generated meshes in a post-processing step. To propagate an orientation thru a non-oriented triangular mesh is a very complex and ambiguous problem.

The moving least square (MLS) approach [Lev03] exposes good capabilities concerning challenging scanned data. Some approaches start with the precomputation of oriented tangent planes as a projection domain for higher degree polynomials which approximate the point data. In these approaches, the contribution of a single polynomial at a given surface point is defined by a non-negative weight function. [ABCO*01] presents a practical implementation and [LCOLTE07] suggest an alternative projection method.

[IJS03b] propose an iterative refinement approach for surface reconstruction which extends a general neural computing approach from Fritzke [Fri93] by a smoothing operation presented by Taubin [Tau95]. Here, a surface is represented by an explicit mesh that consists of triangles with a given

orientation and which incrementally grows during a training process (see Fig. 1).

One important problem of this approach is that objects to be modeled adhere to topology which has to be defined initially — the mesh homeomorphism is static. Cutting and coalescing the mesh during the iterative growing process are imperative to match arbitrary homeomorphisms. In [IJS03a] the triangle size is suggested as an indicator for a cut operation and in [AB10] cutting the mesh is triggered by high vertex valences. Although these approaches are promising in creating acceptable surfaces even for above mentioned problems and unclosed objects, they may produce inconsistent overall surface orientations in cases where differently oriented surface pieces melt together. The arising surface twists can not be resolved by local refinement procedures (see Fig. 2).

In the following we give a brief introduction into the underlying iterative surface reconstruction algorithm and present a general solution to the problem of twisted surface orientations. Finally we prove validity, reliability, and efficiency of the presented approach by applying several test cases.

2. Iterative Refinement Approach for Reconstruction

The basic concept of an iterative refinement process is to fit an initial surface into an unorganized point-cloud. The initial surface is commonly represented by a very simple mesh of oriented triangles, like a tetrahedron. The refinement process randomly selects a samples of the point-cloud and then deforms the current surface in order to progressively minimize its distance to the given sample. This basic step is repeated constantly throughout the algorithm. During this iteration

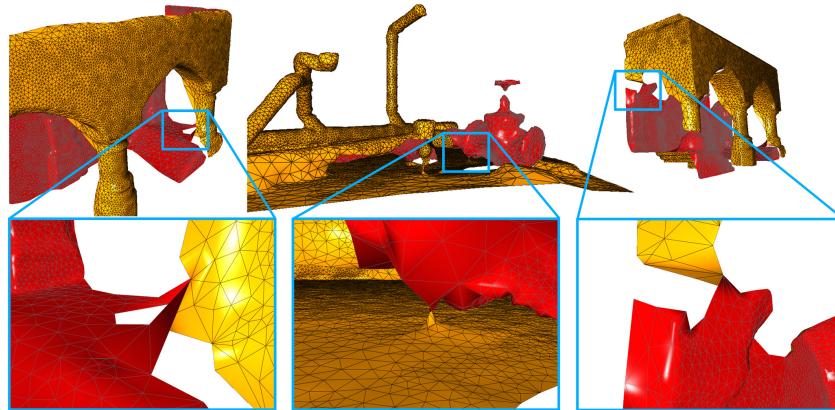


Figure 2: Several examples of twisted connections of surface pieces. Twists at boundary vertices (right and left), and a self intersection (middle). Note that contrary oriented triangles are never connected directly.

the process keeps track of the local surface error which is based on the point samples' distance and which can be measured in different ways. These errors are used to determine surface areas which need to be refined by local subdivision processes. Subdivision and further iterations lead to a better match of the mesh to the sample distribution and the process is stopped when the representation error reaches a certain minimum. Iterative refinement approaches compete with the state of the art in surface reconstruction but they expose advantages in robustness and flexibility. Due to its incremental kind of training and the locality of mesh modifications they are able to match nearly any given surface. Nevertheless the *local* type of mesh adaption makes it difficult to solve *global* mesh problems like twists of the orientation of subsets of triangles. In the following we describe an algorithm which rises that challenge.

A typical iterative refinement process for mesh reconstruction has been presented by the *Growing Cells Meshing (GCM)* algorithm [AB10]. It is the base for the following discussions, but our approach is generally suitable for any kind of iterative local reconstruction schemes. A comprehensive outline of the GCM algorithm is exposed in algorithm 1. Here a mesh based surface is repeatedly deformed by moving the closest vertex and its neighbors toward the randomly selected samples. Using a vertex-local signal counter the process keeps track of how often vertices have been selected during the iterations. When a certain number of deformations at a vertex took place, the surface there is refined by adding a new vertex through a vertex split operation. Vertices which have been selected rarely are removed by edge collapse operations. If a vertex valence indicates a misplaced surface the vertex is removed together with the surrounding faces. The latter operation allows for opening the surface. Coalescing operations then seal and rearranging the surface. Iterations are repeated until accuracy ex-

ceeds a certain threshold. For further details on the algorithm see [AB10].

3. The Emergence of Global Twists

An iterative refinement approach is very capable in creating sound orientated surfaces. Surfaces always evolve from former surface stages. This kind of inertia avoids local twists caused by ambiguous point constellations. At any stage of the iterative process the current surface can be seen as a guess, based on the sample information that has been encountered so far. The more information is processed, the more precise this guess becomes. If however the topology has been guessed incorrectly, this might lead to a surface twist. If for example an s-shaped plane is reconstructed with little sample information processed so far the structure might be recognized as being box-shaped leaving the plane ends with opposing orientations. When the process refines the surface further and the correct shape of the surface becomes evident the different orientations will collide at some point and can not be resolved by local refinement operations.

4. Resolving Twisted Surfaces in Iterative Refinement

Our approach involves three basic steps to solve the twisted surface problem. First, a twist within the current surface is identified, second, the twisted surface is separated from the mesh, and third, the twisted surface is turned around.

The third step is fairly easy. Depending on the underlying data structure, explicit normals are inverted, or vertex orders are reversed if a mesh partition's orientation needs to be changed.

The first step is invoked at every coalescing operation (line 9 at algorithm 1). This is a convenient location to attach this function since it already involves a boundary vertex \mathbf{v}_x ,

Algorithm 1 The complete iterative refinement algorithm without the surface twist operation. Conditions 1 and 2 can be defined as simple counters or depending on a certain approximation error. Overall performance does not significantly depend on them.

- 1: Given a point-cloud $\mathcal{P} = \{p_1 \dots p_n\}$ and an initial Mesh $\mathcal{M} = \{v_1 \dots v_n\}$.
- 2: **repeat**
- 3: **repeat**
- 4: **repeat**
- 5: Select random sample \mathbf{p}_x of \mathcal{P} and search vertex \mathbf{v}_x with smallest Euclidian distance to \mathbf{p}_x .
- 6: Move \mathbf{v}_x towards \mathbf{p}_x .
- 7: Smoothen all direct neighbors of \mathbf{v}_x .
- 8: Increase signal counter of \mathbf{v}_x and decrease all others.
- 9: If \mathbf{v}_x is a boundary vertex then search for opposing boundary vertex \mathbf{v}_{opp} and coalesce both if their normals have the same orientation.
- 10: **until** condition 1 holds.
- 11: Add new vertex at the vertex with highest signal counter through a vertex split operation.
- 12: **until** condition 2 holds.
- 13: **repeat**
- 14: Search for vertex \mathbf{v}_{null} with the lowest signal counter.
- 15: **if** valence of \mathbf{v}_{null} indicates a misplaced surface **then**
- 16: Remove \mathbf{v}_{null} and its surrounding triangles.
- 17: **else**
- 18: Remove \mathbf{v}_{null} by an edge collapse operation.
- 19: **end if**
- 20: **until** no vertices with low signal counters exist.
- 21: **until** accuracy exceeds a certain threshold.

the search for an opposing boundary vertex \mathbf{v}_{opp} , and testing of their orientations.

4.1. Twist Detection

A twist between two boundaries is determined by opposite normal orientations at the same planar domain. This is the case if, first, the size of the angle α between the normals \mathbf{n}_x and \mathbf{n}_{opp} of the opposing vertices \mathbf{v}_x and \mathbf{v}_{opp} exceeds 170° , and second, if \mathbf{v}_x and \mathbf{v}_{opp} lie at the same planar domain, i.e., if the angle β between \mathbf{n}_x and $\overrightarrow{\mathbf{v}_x \mathbf{v}_{opp}}$ differs not more than between 80° and 100° . Since \mathbf{n}_{opp} and \mathbf{n}_x are more or less parallel the latter also includes the test for \mathbf{v}_x (see Fig. 3).

Since resolving a twist is very costly, superfluous processes should be avoided. Thus, the above test is repeated for both neighbors of \mathbf{v}_x and \mathbf{v}_{opp} , and only if all three tests expose a twist a valid detection is assumed.

On the one hand, this kind of detection is chosen arbitrary

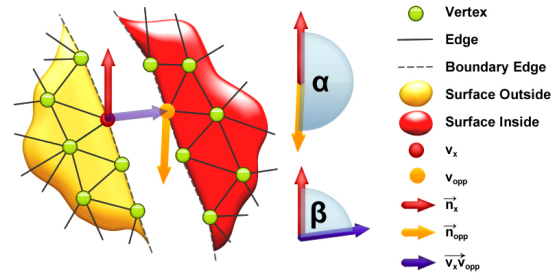


Figure 3: Detection of a twisted surface through angles α between the normals and β between a normal and the vector connecting the two vertices under consideration.

and one can think of several alternatives. On the other hand, the specific kind of test is not that significant in the overall algorithm's validness or performance, since it is not required to detect a twist at a very early stage as long as it is detected at some time. The latter is guaranteed by the refinement process of the GCM. The only possible case of a fail of this approximate detection arises if the twist is not exposed at any boundary or if the twisted surface has not been clearly exposed before the process ends.

4.2. Twist Separation

When the surfaces which \mathbf{v}_x and \mathbf{v}_{opp} belong to are detected as being twisted, one of them needs to be turned around. As long as the surfaces are connected, the turning operation would fail since both surfaces would be turned which would not solve the twist. It seems to be surprising that differently orientated surface areas could be connected since the coalescing process would avoid such a connection. Nevertheless, this case actually may happen if these surfaces are connected through an earlier refinement stage where the difference of the normals have not yet been developed that far. Fig. 2 gives an impression of the various kinds of connections that may exist between twisted surfaces. To single out these different structures by heuristics using geometric properties like vertex positions and normals is extremely difficult and unreliable. To avoid this, we rely on the mesh topology only for detecting these cases. It is achieved by a data structure that can reveal different kinds of mesh topology based information — a minimal edge front. It consists of multiple closed circular bands of mesh edges that enclose all vertices of a certain edge-wise distance to an initial starting point. For example, if a minimal edge front is expanded two times from an initial vertex it will enclose all vertices that can be reached from this initial vertex in a distance of two mesh edges (see *a*) in Fig. 4). A minimal edge front is minimal in the way that the vertices it surrounds cannot be enclosed by a smaller number of mesh edges. Every expansion level is unique and therefore reversible.

To find out if \mathbf{v}_x and \mathbf{v}_{opp} are connected we initialize two

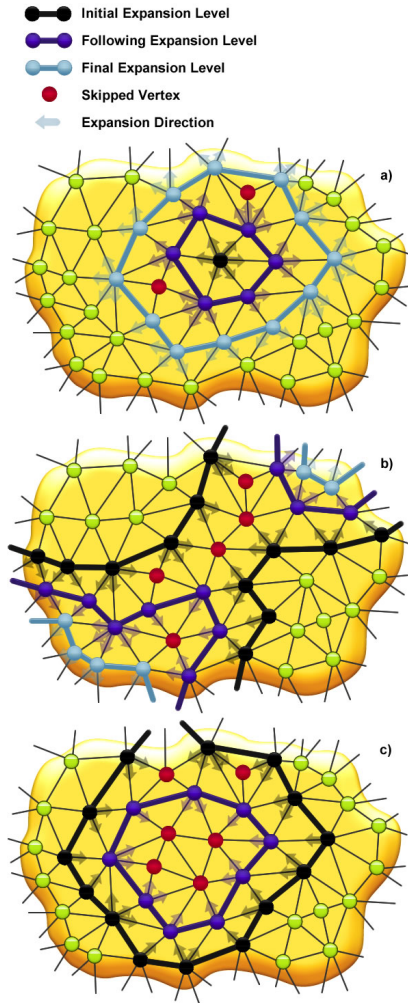


Figure 4: Different cases while expanding a minimal edge front. a) Expansion from an initial stage from one vertex to the second expansion level, b) Collision and merging while expanding an edge front, c) Annihilation of an edge front that can not be expanded further.

minimal edge fronts at each vertex and propagate them until they collide (see *b*) in Fig. 4) or until one of them is annihilated (see *c*) in Fig. 4), which means that it can not be expanded anymore and did not yet reach the other vertex. The latter means that \mathbf{v}_x and \mathbf{v}_{opp} are unconnected and since the extension of both fronts have been run concurrently, the vertex of the annihilated front belongs to the smaller surface partition. At this point the separateness of \mathbf{v}_x and \mathbf{v}_{opp} is proven and the orientation of the smaller partition is switched. If \mathbf{v}_x and \mathbf{v}_{opp} are connected the according mesh parts must be separated before turning one of them around. To separate them, we calculate the edge-wise shortest con-

nection path between \mathbf{v}_x and \mathbf{v}_{opp} (see *a*) in Fig. 5) and then search the shortest cut to interrupt this connection path.

In a first stage we search for the shortest cut on the path that starts and ends at a boundary vertex. The search starts with the cut length zero involving a vertex that has more than two boundary edges (see *b*) in Fig. 5). The search is accomplished for all vertices on the path. If a zero length cut can not be found the search length is increased by one. To search for a higher cut length than zero along the path, we determine the two closest boundary vertices to a given vertex of the connection path. The search is performed with the minimal edge front from above. The connection between these two boundary vertices defines a cutting path. In *c*) Fig. 5 a cutting path of length one is shown. It needs to be tested if this cut would actually interrupt the path between \mathbf{v}_x and \mathbf{v}_{opp} . *e*) Fig. 5 shows a cutting path of length one that fails to interrupt the connection path. It also needs to be checked if the cut would detach \mathbf{v}_x or \mathbf{v}_{opp} from the surface like shown in *f*) Fig. 5.

If no sufficient cut could be found in the first stage, the second search stage will be started. Here, circular cuts are investigated, which do not necessarily involve boundary vertices, and which are defined by a cutting path with circular connected edges. The search for circular cuts starts with the length of three as shown in *d*) in Fig. 5. When a vertex on the path is tested, a minimal edge front is initialized at that vertex and the front is expanded until a collision of the front takes place (see *b*) in Fig. 4), or the current search length is exceeded. If this path interrupts the connection path between \mathbf{v}_x and \mathbf{v}_{opp} and the triangles intersect each other, like in *d*) Fig. 5, it is a valid cutting path.

If a cutting path has been found which satisfies the above criteria, a cut will be performed. All edges of the side of the cutting path that includes less triangles are deleted. Now \mathbf{v}_x and \mathbf{v}_{opp} are tested again for a connection and the process repeats until \mathbf{v}_x and \mathbf{v}_{opp} are separated.

5. Results

Twist resolving in an iterative refinement algorithm is a novel addition to the approach and in this section we focus on the reliability and efficiency of the presented algorithm. Since our approach has no influence on the general surface reconstruction mechanism, we refer to [AB10, IJS03b] for quality analysis of the algorithm.

The runtime complexity of the GCM algorithm is $O(n \log n)$ where n is the number of vertices in the final mesh [AB10]. In the following we will approximately determine the complexity of the presented operation. If we assume that the twist detection works correctly and the unknown surface is orientable, then — once detected — twists will be resolved again and again, which strongly limits the number of twists that can realistically occur in a reconstruction process. The worst case concerning memory and run-

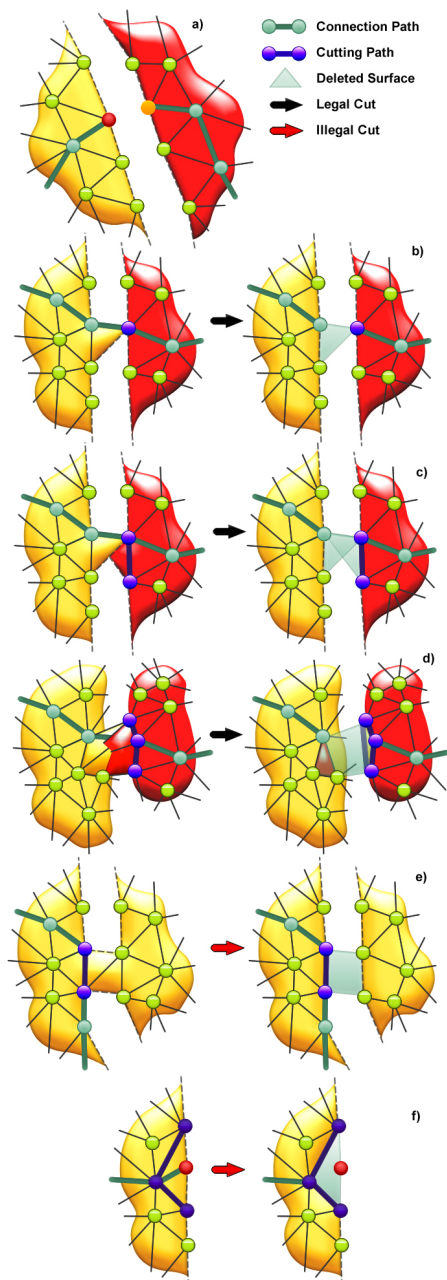


Figure 5: v_x and v_{opp} with their connection path a). A cut of length zero b) and one c) from boundary to boundary. A circular cut of length three through a self intersecting surface d). An illegal cutting path that would not interrupt the connection path e). An illegal cutting path that would detach v_x from the surface f).

time complexity would take place at the end of the algorithm if both regions have the same size of $\frac{n}{2}$, since the searching mechanism visits the biggest number of possible vertices. The memory complexity of the minimal edge front is $O(\sqrt{n})$ since the front's dimension is one and it expands uniformly over the surface. The runtime complexity can be determined by the number of vertices visited while searching the connection path and the number of search passes that are performed until the regions are separated. The search for a cut along the connection path and the switching operation are both done in addition to the search for the connection path, but both have smaller complexity. The number of vertices visited by the search operation is proportional to n and the search has to be accomplished for all incorrect connections. Since the crack between the differently oriented surface areas can be assumed as being one-dimensional and not exposing a fractal structure the number of possible wrong connections is \sqrt{n} . This creates an overall worst case complexity of $O(n^{\frac{3}{2}})$. However, that a twist in a huge surface area remains undetected throughout the entire process is very unlikely. Twists are detected at a relatively constant stage in their development. Thus in the average case the effort of the twist resolving process is not depending on n .

The Stanford Dragon model serves as a good example for a point cloud that should not produce twisted surfaces at all, the Vault model is a suitable example for a simple twisted surface, and finally, the complex Heating Pipes model includes noise, outliers and non-uniform sample densities which requires multiple twist resolving processes. Finally, we provoke the algorithm with an extreme model concerning twists — the point cloud of the Möbius strip. The first time, all tests run without enabling the presented algorithm, and after this, the twist resolution step is added (see table 1 and Fig. 6).

We ran the Heating Pipes model using different random seeds leading to different amounts of resolving processes but always to a soundly oriented final surface. This reliability is achieved through using the mesh topology rather than geometric properties of the vertices. The following results have been produced on an Intel® Core 2 Extreme Quad Core QX9300 (2.53GHz, 1066MHz, 12MB) processor with 8GB 1066 MHz DDR3 Dual Channel RAM. The algorithm is not parallelized.

The presented minimal edge front data structure is a very fast way for gaining edge based distances and other mesh related information while having a very small memory footprint. On average for 1000 different starting vertices, a minimal edge front needs 0.343s to visit all vertices of the Dragon model, while it consists at its maximum of 3069 edges.

The maximum number of twist resolving processes during the Heating Pipes model reconstruction were three. The impact on runtime is obviously beneath the standard deviation of the processes duration (see table 1) since the Heating





	#Pts	#Trigls	without	with	#resolvings
	438K	100K	51sec	51sec	0
	368K	40K	16sec	16sec	1
	918K	100K	73sec	72sec	3
	163K	40K	13sec	76sec	374

Table 1: The table shows results for different models. “#Pts” the sample size of the point cloud, “#Trigls” the number of triangles in the final mesh, time “without” the twist resolving process, time “with” the twist resolving process, “#resolving” the number of completed resolving processes.

Pipes were even faster in our test with the twist resolving process. For the Möbius strip however the twist resolving process was started 374 times until the demanded vertex resolution was reached, which had a significant impact on the overall runtime of the process.

6. Conclusion and Future Work

An iterative refinement process for surface reconstruction has a lot of advantages like the creation of different levels of detail, a strong robustness against noise, the ability to process arbitrary amounts of point data, and being able to create a correct surface gradient even in areas where point normal estimation is hardly to accomplish.

However, up to now, this process was limited to point data which do not cause twists in the produced surface. With the presented method this limitation is lifted and eliminates the last vital limitation to the class of surfaces being able to be reconstructed by iterative refinement approaches.

The minimal edge front delivers a very general solution to the problem of separating two surface areas, which fortunately does not depend on geometry based heuristics. The structure could potentially be used to calculate texture coordinates or geodesics on meshes, or to enable mesh decomposition.

Our method is based on the assumption that surfaces with different orientations can always be encountered at a boundary edge. This assumption can fail if two entirely separate objects are contained in one point-cloud or if the only crossing of two differently oriented surfaces is a self intersection which is unlikely but possible. If a point-cloud contains structures that are represented by only few points it may happen that a twisted surface can not be recognized.

Further, the process has shown weaknesses when solving twists that are caused by thin structures — surfaces with opponent orientations which lie close together — which the approach covered as one surface at initial stages. It leads to greater calculation times of the refinement process. Nevertheless, finally, twists are resolved correctly.

Acknowledgement The authors would like to thank the Stanford University Computer Graphics Laboratory and the Hamburg Hafen City University for making their laser scanned data available to us. We also would like to thank Kai Burjack for his assistance in rendering

References

- [AB10] ANNUTH H., BOHN C.-A.: Surface reconstruction with smart growing cells. *Studies in Computational Intelligence* 321 (2010), 47–66. [2](#), [3](#), [5](#)
- [ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), VIS '01, IEEE Computer Society, pp. 21–28. [1](#), [2](#)
- [ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 415–421. [1](#)
- [BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics* 5, 4 (Oct. 1999), 349–359. [2](#)
- [CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 67–76. [1](#)
- [CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 303–312. [1](#)
- [CLK09] CHANG M.-C., LEYMARIE F. F., KIMIA B. B.: Surface reconstruction from point clouds by transforming the medial scaffold. *Comput. Vis. Image Underst.* 113, 11 (Nov. 2009), 1130–1146. [2](#)
- [DRADLN10] DO RÊGO R. L. M. E., ARAÚJO A. F. R., DE LIMA NETO F. B.: Growing self-reconstruction maps. *Trans. Neur. Netw.* 21, 2 (Feb. 2010), 211–223. [2](#)
- [Fri93] FRITZKE B.: Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks* 7 (1993), 1441–1460. [2](#)
- [GK02] GOPI M., KRISHNAN S.: A fast and efficient projection-based approach for surface reconstruction. In *Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing* (Washington, DC, USA, 2002), SIBGRAPI '02, IEEE Computer Society, pp. 179–186. [2](#)
- [HDD*92] HOPPE H., DE ROSE T., DUCHAMP T., McDONALD J. A., STUETZLE W.: Surface reconstruction from unorganized points. In *SIGGRAPH* (1992), Thomas J. J., (Ed.), ACM, pp. 71–78. [1](#)

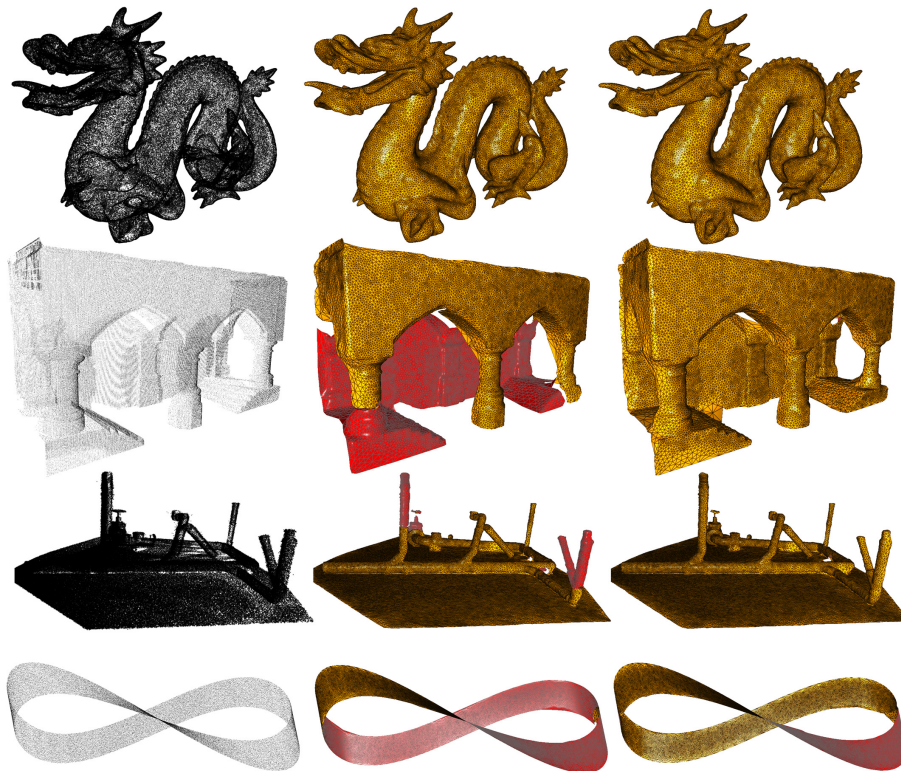


Figure 6: The Dragon, Vault, Heating Pipe and Möbius strip model as a point cloud (left), without our twist correction mechanism (middle) and with it (right).

- [HF08] HOLDSTEIN Y., FISCHER A.: Three-dimensional surface reconstruction using meshing growing neural gas (mgng). *Vis. Comput.* 24 (March 2008), 295–302. 2
- [HK06] HORNING A., KOBELT L.: Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2006), SGP '06, Eurographics Association, pp. 41–50. 1
- [HLZ*09] HUANG H., LI D., ZHANG H., ASCHER U., COHEN-OR D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 176:1–176:7. 2
- [IJS03a] IYRISIMTZIS I., JEONG W.-K., SEIDEL H.-P.: *Neural Meshes: Statistical Learning Methods in Surface Reconstruction*. Tech. Rep. MPI-I-2003-4-007, Max-Planck-Institut für Informatik, Saarbrücken, April 2003. 2
- [IJS03b] IYRISIMTZIS I. P., JEONG W.-K., SEIDEL H.-P.: Using growing cell structures for surface reconstruction. In *SMI '03: Proceedings of the Shape Modeling International 2003* (Washington, DC, USA, 2003), IEEE Computer Society, p. 78. 2, 5
- [KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70. 1
- [LCOLTE07] LIPMAN Y., COHEN-OR D., LEVIN D., TALEZER H.: Parameterization-free projection for geometry reconstruction. In *ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 2
- [Lev03] LEVIN D.: Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization* (2003), 37–49. 2
- [MAVdF05] MEDEROS B., AMENTA N., VELHO L., DE FIGUEIREDO L. H.: Surface reconstruction from noisy point clouds. In *Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2005), SGP '05, Eurographics Association. 1
- [OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 463–470. 1
- [SLS*06] SHARF A., LEWINER T., SHAMIR A., KOBELT L., COHEN-OR D.: Competing fronts for coarse-to-fine surface reconstruction. In *Eurographics 2006 (Computer Graphics Forum)* (Vienna, october 2006), vol. 25, Eurographics, pp. 389–398. 1
- [SSZCO10] SHALOM S., SHAMIR A., ZHANG H., COHEN-OR D.: Cone carving for surface reconstruction. *ACM Trans. Graph.* 29, 6 (Dec. 2010), 150:1–150:10. 1
- [Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM, pp. 351–358. 2