On the Implementation of Centerline Extraction Based on Confidence Vote in Accumulation Map

Bertrand Kerautret[1,2(✉)], Adrien Krähenbühl[4], Isabelle Debled-Rennesson[1,2], and Jacques-Olivier Lachaud[3]

[1] Université de Lorraine, LORIA, UMR 7503, 54506 Vandoeuvre-lès-Nancy, France
{bertrand.kerautret,isabelle.debled-rennesson}@loria.fr
[2] CNRS, LORIA, UMR 7503, 54506 Vandoeuvre-lès-Nancy, France
[3] LAMA (UMR CNRS 5127), Université Savoie Mont Blanc, 73376 Chambéry, France
jacques-olivier.lachaud@univ-savoie.fr
[4] LaBRI (UMR CNRS 5800), 351, Cours de la Libération, 33405 Talence Cedex, France
adrien.krahenbuhl@labri.fr

Abstract. This paper focuses on the implementation details of a recent method which extracts the centerline of 3D shapes using solely partial mesh scans of these shapes. This method [9] extracts the shape centerline by constructing an accumulation map from input points and normal vectors and by filtering it with a confidence vote. This paper presents in details all the algorithms of the method and describes the implementation and development choices. Some experiments test the robustness to the parameter variability and show the current limitations allowing to consider further improvements.

1 Introduction

Extracting the centerline of a shape is a classical problem in geometry processing and in image analysis. It can be seen as a special case of skeleton extraction for shapes with local approximate cylindrical symmetry. This problem has been addressed many times in the literature, one can refer to Tagliasacchi *et al.* [18] and Saha *et al.* [15] for recent surveys. As mentioned in the first survey, the wide deployment of such methods in many relevant applications is still missing [18]. In order to make easier the concrete deployment of academic centerline extraction methods in industrial applications, we present in details a recent method specialized for the extraction of centerlines of approximately tubular shapes with possibly branching [9] and we provide its complete implementation. This method presents several advantages: it can process partial mesh scans, it is robust to perturbations and relatively independent to parameters, it is not difficult to reproduce and implement. For instance, the replication of the

J.-O. Lachaud—This work was partially supported by the ANR grants DigitalSnow ANR-11-BS02-009.

B. Kerautret et al. (Eds.): RRPR 2016, LNCS 10214, pp. 116–130, 2017.
DOI: 10.1007/978-3-319-56414-2_9

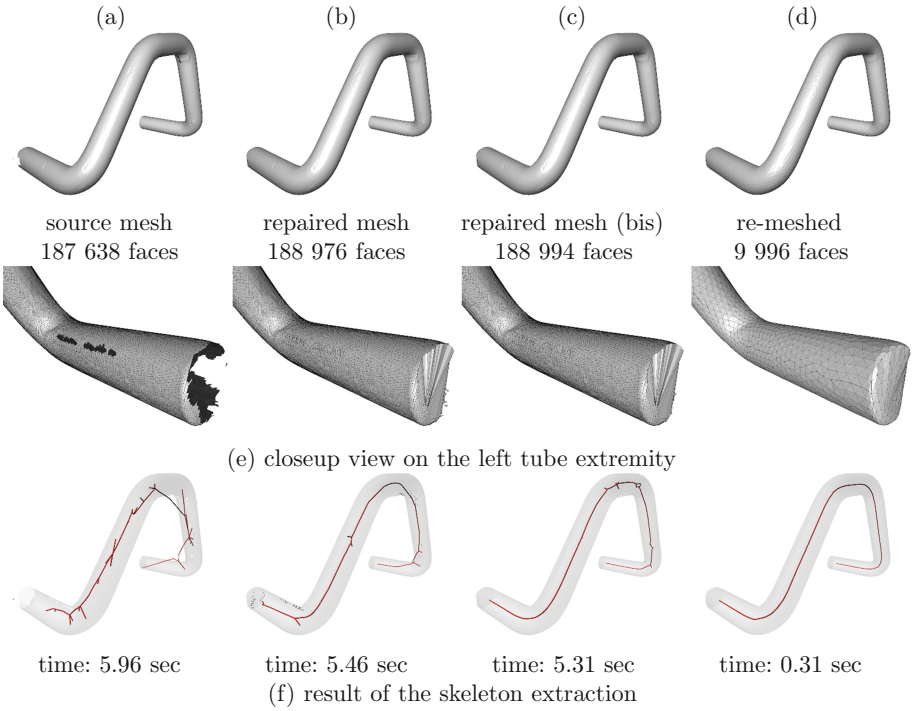


Fig. 1. Measure of the sensibility of the skeleton extraction method of [17] on data with different qualities: (a) raw data extracted directly from the scanner, (b) repaired mesh with hole closure, (c) mesh (b) + face intersection removal, (d) new re-meshing by keeping 5000 points.

method (in the meaning of Goodman's definition [8]) was demonstrated with its integration in an industrial framework by a company manufacturing tubular shapes for aircraft cabins.

The originality of this new method compared to existing works was already presented in the latter reference [9], and is not the main topic of this paper. Nevertheless we examine the sensibility with respect to input data quality of a recent method [17] publicly available in the *CGAL* library [1]. Figure 1 displays the different results obtained by running this method respectively on raw mesh scans (column (a)), on mesh repaired by hole closure (column (b)), on mesh (b) repaired by face intersection (column (c)), and finally after a complete re-meshing and keeping a reduced number of faces (column (d)). Such repairing processes were obtained using the *geogram* library [3,4], which proposes a complete set of re-meshing tools based on different methods [11,12,19]. It appears that the quality of output centerlines depends directly of the mesh quality. In particular, for the source mesh (column (a) of Fig. 1), the resulting skeleton appears really noisy. Its quality is then significantly improved with mesh repaired and re-meshed (columns (b) to (d)). We will see in the following that the proposed method is far less sensitive to the quality of the input data, and even does not use their topology.

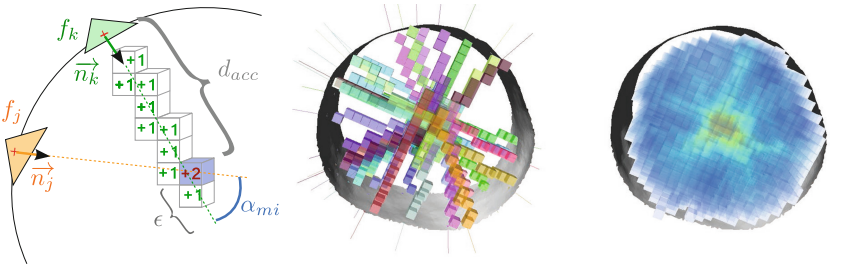
The rest of the paper is organized as follows: in Sect. 2, the main idea and algorithms of method [9] (confidence vote in accumulation map) are introduced, together with a short description of a former method [10] at the origin of this new one. Implementation details and source code are described in Sect. 3. Then the way to reproduce experiments and results are presented in Sect. 4 before concluding with some open problems and perspectives.

2 Centerline Extraction from Confidence Map

We describe first the notion of accumulation map, which was the main idea of a preliminary work for centerline extraction [10]. In a second stage, we describe how the new method introduced in [9] has built upon this work in order to get a much more robust centerline as output. The main idea of this second work was to add a confidence vote to filter the accumulation map.

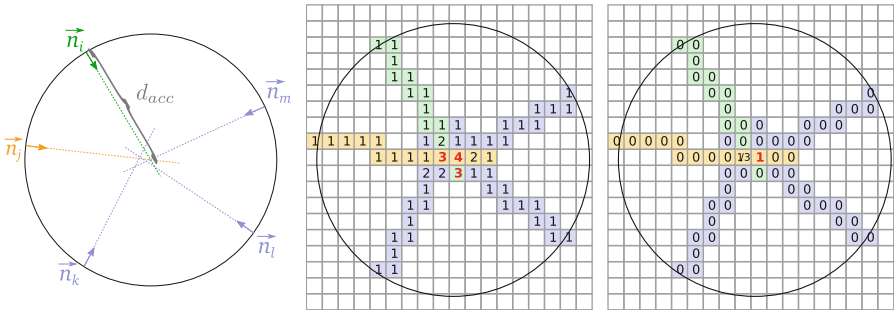
Single accumulation map. The preliminary proposed approach to extract the centerline was built an accumulation map from the set of input points and their associated normal vectors. The principle of accumulation map is illustrated in Fig. 2. Starting from a point, taken for example as the center of a mesh face f_k, and the corresponding normal vector $\overrightarrow{n_k}$ (see Fig. 2(a)), the algorithm adds one to the score of each voxel intersected by the ray of length d_{acc}, starting from the considered point and directed toward vector $\overrightarrow{n_k}$. Voxels located on such rays are illustrated with different colors in Fig. 2(b).

From the resulting accumulation map (see Fig. 2(c)), the centerline points are tracked by moving from the peaks of this map in the direction orthogonal to intersecting rays. This approach is robust enough to handle a simple shape without branching. The algorithm was successfully exploited and reproduced in a concrete application dealing with the detection of wood trunk defects [13]. However such an application is not adapted to deal with a branched centerline.



(a) accumulation idea (b) casting digital rays (c) accumulation map ($imAcc$)

Fig. 2. Main steps of the preliminary approach [10]. The image (a) shows how digital rays are casted from input data to define the accumulation map. The voxels intersected by the same ray are displayed in the same color on image (b). The resulting accumulation map, stored as an image, is illustrated in image (c), where the red (resp. blue) color corresponds to a high (resp. low) accumulation. (Color figure online)



(a) Normal vectors & rays (b) Accumulation map ($imAcc$) (c) Confidence map ($imConf$)

Fig. 3. Construction of the confidence map from the accumulation map. (Color figure online)

Therefore a new notion of confidence in the accumulation value was proposed to extend the centerline detection method to branched tubular objects.

Notion of confidence in accumulation. The centerline extraction was significantly improved by adding a confidence value in the vote that represent the accumulation scores [9]. The main idea is to define a confidence map ($imConf$), which is deduced from the accumulation map ($imAcc$) with a maximality principle. More precisely, let v be a voxel and let v_{acc} be the number of rays traversing it, then v_{max} is defined as the number of rays passing trough v and for which v_{acc} is a strict maximum value along the whole ray. Figure 3(b) illustrates this maximality principle where red values are maximal accumulation values for at least one ray. Then, from these v_{max} values, the confidence value v_{conf} is simply defined as the ratio between values v_{max} and v_{acc}:

$$v_{conf} = \frac{v_{max}}{v_{acc}}$$

As illustrated in Fig. 3, the confidence scores appear to be more concentrated near the center of the circular shape than the accumulation map scores. Such quality was already analyzed in a previous work by computing the number of connected components resulting of different confidence/accumulation thresholds (see Fig. 5 in [9]). These analysis were obtained on a single class of shapes. The analysis Fig. 4 complete it on meshes with various quality levels. The better behavior of the confidence map with respect to the sole accumulation map is also well visible: even if we consider partial data, voxels located near the center of the shape are well identified in the confidence map even for very different thresholds, while this is not the case for the accumulation map. The measures can be reproduced from the `compAccFromMesh` program (see Sect. 3) and for instance the experiments presented in Fig. 4(a) are obtained with the command lines in Code 1.1 (and by using the minimum threshold parameter (`-m 25`)).

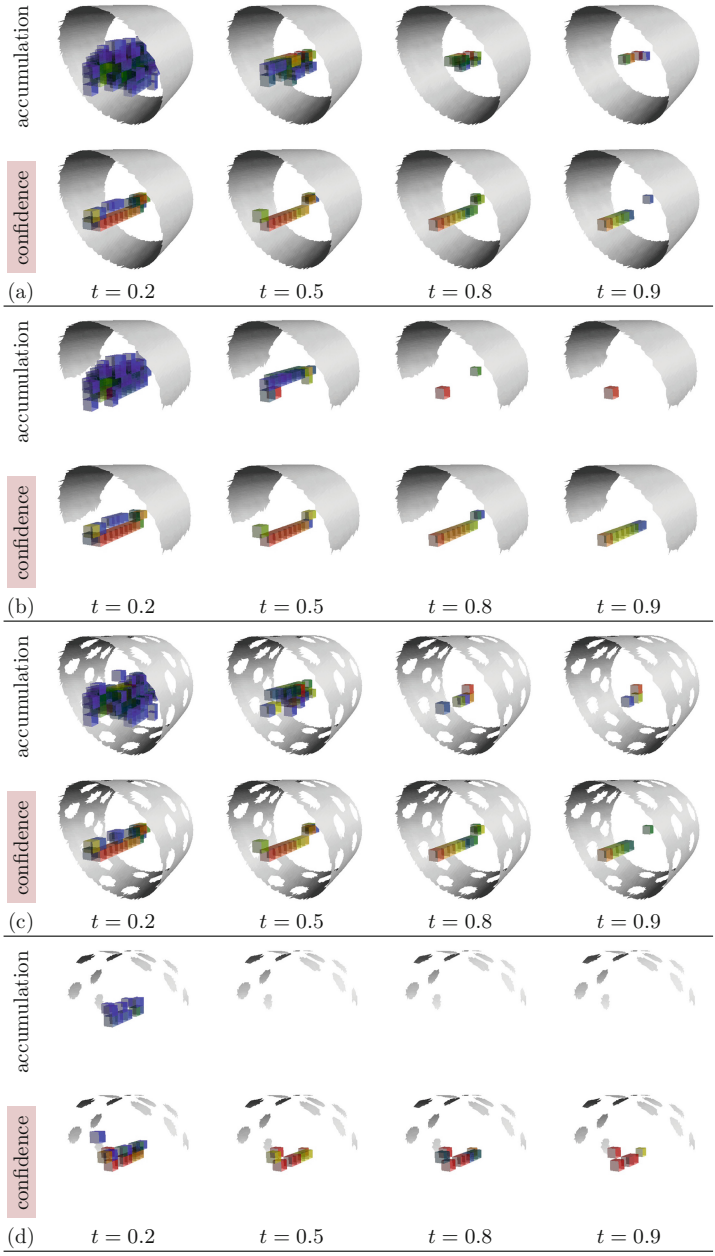


Fig. 4. Experimental comparisons of sensitivity to thresholding between the accumulation map and the confidence map. Thresholds were set between 0 (min accumulation/confidence value) and 1 (max accumulation/confidence value) on all meshes. Note that the parameter t for the accumulation was rescaled according the the maximal value of accumulation. The same maximal radius parameters $R = 7$ was used in all experiments.

Code 1.1. Command lines to compute and display the confidence map, used to generate the experiments presented in the second line in Fig. 4(a). Replace `confidence.longvol` by `accumulation.longvol` to obtain the accumulation map computed at the first line.

```
$ ./bin/compAccFromMesh -i ../SamplesIllustration/sectionATube1.off -r 7
$ longvol2vol -i confidence.longvol -o confidence.vol
$ 3dImageViewer -i confidence.vol -m 25 -M 255 -t 120 \
                --displayMesh ../SamplesIllustration/sectionATube1.off \
                --colorMesh 127 127 127 100
```

All detailed algorithms are given in the appendix: the global process (Algorithm 1) computes the accumulation map with (Algorithm 2) before using it as an input parameter in the algorithm computing the confidence map (Algorithm 3).

Centerline extraction from confidence map. Since the confidence map locates centerline points in a more accurate and stable way, the former centerline extraction algorithm from [10] can be redefined to handle branchings. In particular, as illustrated in side Fig. 5, a simple threshold on the confidence map gives a set of voxels that is almost a connected path. A geodesic-based graph extraction was proposed in [9] to track it and is detailed in Algorithm 4. This algorithm consists in first applying a dilation on the thresholded confidence map before tracking center points according to a the geodesic distance map of $imConf$, computed from an initial point P_{init}. The geodesic map is divided in a set regions corresponding to connected components at a specific geodesic distance from P_{init}, then a representative point of each region is selected and the graph is reconstructed by linking the representative points of connected regions.

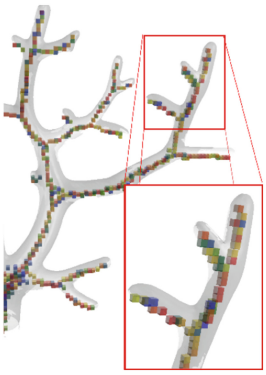


Fig. 5. Points resulting from a thresholding on confidence map.

Following this overview of the proposed method, the next section details the implementation of the different algorithms and describes the potential influence of the different parameters.

3 Implementation Description

The source code of the proposed implementation is available on Github:

https://github.com/kerautret/CDCVAM

The proposed implementation is written in C++ and is based on the *DGtal* library [2]. The algorithms computing accumulation and confidence maps can easily be implemented in another language. The only part of *DGtal* specifically exploited concerns the geodesic graph extraction method, which uses DGtal implementation of the Fast Marching Method [16]. This algorithm is used in Algorithm 4 both to compute the dilation and the geodesic images. Of course other strategies to reconstruct the graph can be defined like for instance the

use of a metric graph algorithm [7]. Moreover, the implementation only deals with meshes as input but the first step consists to extract the point list and the normal vector field. It is really easy to create another wrappers allowing to deal with other entries without changing the core functions.

Sources organization. The main algorithms are implemented in the *src* directory, and organized into classes: the classes `NormalAccumulator` and `GeodesicGraphComputer` implement respectively Algorithms 2 and 4. This directory also contains helper classes and functions with different purposes: adaptation of input data to accumulation map and confidence map algorithms (`AccumulatorHelper`), optimization of the center line position (`CenterLineHelper`) as described in [10]. The main programs used for experiments are located in the *bin* directory (like the program of Code 1.1). Other programs which generate the figures and plots of [9] are also given in this directory. Finally the *tests* directory gives various programs to display and control the main key features of our method: accumulation map, confidence map, geodesic graph extraction.

Implementation choices. For the accumulation map algorithm, we chose to store the scores in a 3D image with bounds corresponding to the mesh bounding box. In the proposed implementation the choice of the grid size is automatically set to 1. The mesh can be scaled in order to have enough precision in the accumulation process, by using for instance the mean distance between input points. The influence of the grid size parameter is experimented in the next section.

The choice to use a 3D image is a first handy solution. For large shapes, this approach can induce a large memory cost. It should then be adapted to another image structure.

Handling other types of data. In the previous work [9] the experiments were exclusively defined on partial mesh scans. However since the proposed method takes as input only a set of points with normal vectors, it can process other types of data like set of points, digital objects and height maps. To process these other types of data one has only to estimate the normal vector at each point and then import these data in the main program `GeodesicGraphComputer`.

It is also useful to provide tools for processing shapes presenting long rectangular faces like in the image on the side. Such meshes are typically built by geometric modeling software when modeling tubular shapes. In this case, the centerline graph extraction algorithm outputs a disconnected set of voxels. We thus adapt this algorithm by exploiting clusters of confident voxels and by using the main local direction estimated from the voting vectors (note that accumulation and confidence map algorithms remain the same). This case can however be adapted by exploiting clusters of confident voxels and by using the main local direction estimated from the voting vectors. The floating figure on the side illustrates the result that we obtain when using Code 1.1 with this particular type of mesh. All the voxels identifying a common tubular section are well detected and can be exploited for the reconstruction.

Optimizations not described in the algorithms. The proposed implementation of Algorithm 4 contains some optimizations which are not described in this paper for sake of clarity. In particular connected components are obtained with an union-find algorithm during the FMM extension. More details are given in the source files *GeodesicGraphComputer(.h/.cpp)*.

4 Reproducing the Results and Influence of Parameters

The reproduction of the results is straightforward from the programs provided in the *bin* directory of the *GitHub* repository. Results can be inspected with the visualisation tools coming from the *DGtal* companion repositories (*DGtalTools* [5] and *DGtalTools-Contrib* [6]). All command line tools provide a full description of their options. For instance, Code 1.2 shows a typical usage for centerline extraction, which builds the result displayed in Fig. 6.

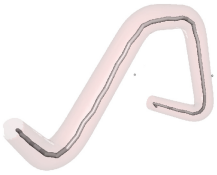


Fig. 6. Result obtained with Code 1.2.

Code 1.2. Command lines to extract and display the centerline of a given shape.

```
$ ./bin/centerLineGeodesicGraph -i ../Samples/tube3.off -R 6  -g 6
$ graphViewer -v resultVertex.sdp -e resultEdges.sdp \
              -m ../Samples/tube3.off --meshColor 250 100 100 25 -b 2
```

In order to integrate the proposed method in other frameworks you can use the code snippet given in Code 1.3. It shows how to compute the centerline of an arbitrary mesh stored as an OFF file. The computation itself is done in two main stages: (i) computation of the accumulation and confidence maps, (ii) threshold on the confidence map followed by graph reconstruction. Note that by importing normal vectors, you can adapt these code sample to process voxels or point clouds. For instance the tool *compAccFromSDP* extracts the centerline of a point cloud and uses the PCL library [14] to estimate normal vectors.

Code 1.3. Compute the centerline of a mesh given as the OFF file "example.off".

```
// Preliminary: read input off file:
DGtal::Mesh<P3d> aMesh; aMesh <<"example.off";

// Step i): compute the accumulation and confidence (with dacc=7)
NormalAccumulator acc(7);
acc.initFromMesh(aMesh);
acc.computeAccumulation();
acc.computeConfidence();
Image3Dd imConf = acc.getConfidenceImage();

// Step ii): apply the centerline extraction from confidence map.
GeodesicGraphComputer::TSet aConfidenceSet(imConf.domain());
// ii. a)  Thresholding the confidence map:
for (auto const &p: imConf.domain())
    if(imConf(p)>= 0.5)
        aConfidenceSet.insert(p);
P3d p0 = acc.getMaxAccumulationPoint();
// ii. b) Computing the graph:
GeodesicGraphComputer gg(4, aConfidenceSet, 3,  acc.getDomain(), p0);
gg.computeGraphFromGeodesic();
```

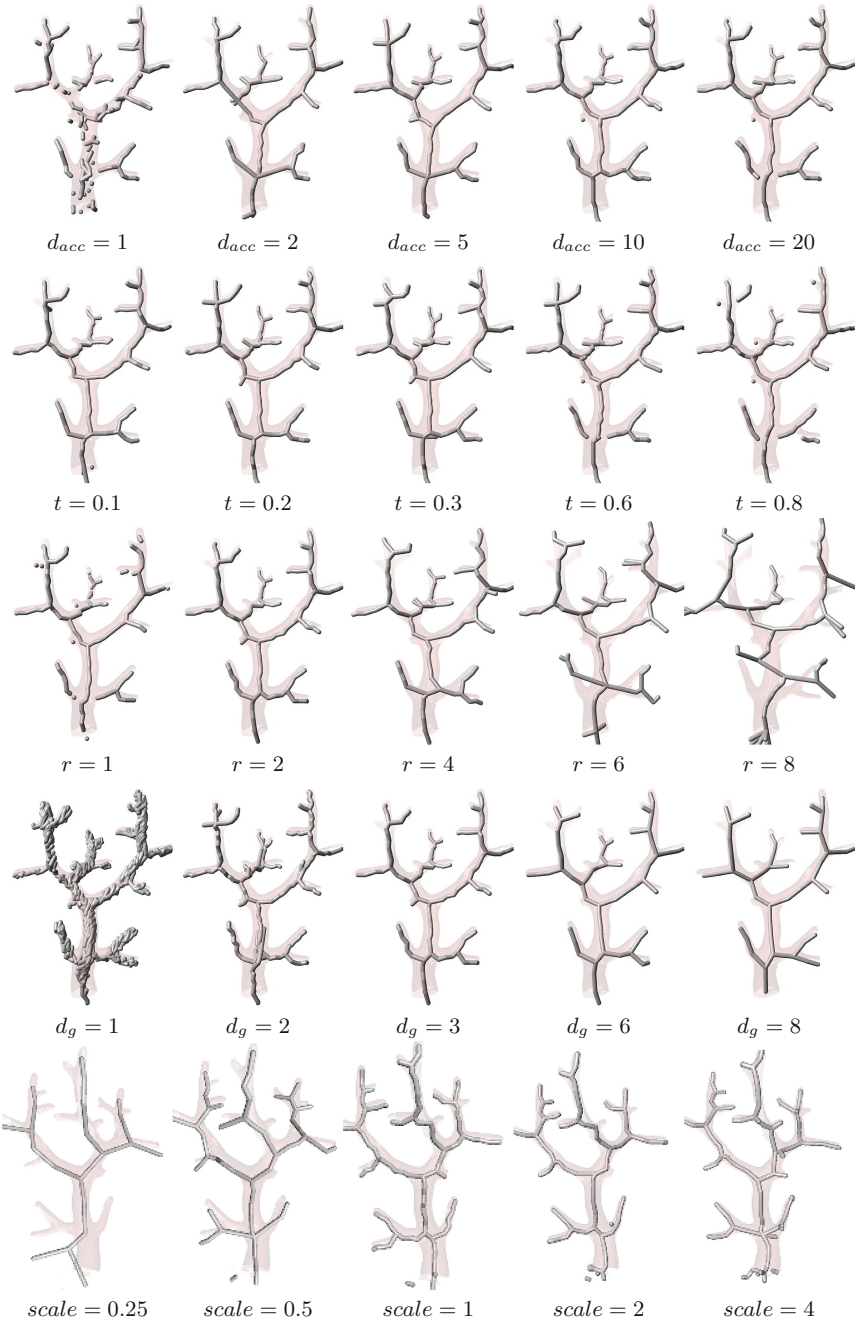


Fig. 7. Influence of parameters when extracting the centerline of *playmobil* tree. For all experiments (expect for the scale) the other parameters were set to their default values ($d_g = 3$, $r = 2$, $t = 0.5$, $d_{acc} = 6$). For the scale parameter experiment, default parameters were scaled accordingly.

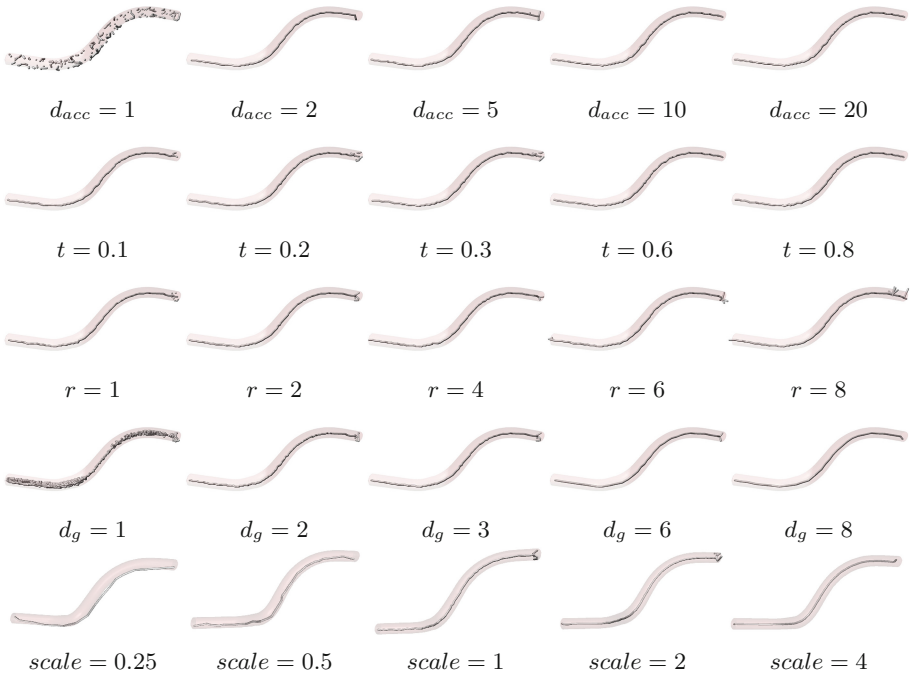


Fig. 8. Influence of parameters when extracting the centerline of a tube shape. For all experiments (expect for the scale) the other parameters were set to their default values ($d_g = 3$, $r = 2$, $t = 0.5$, $d_{acc} = 6$). For the scale parameter experiment, default parameters were scaled accordingly.

Influence of Parameters. The proposed method is tuned by several parameters which can be set according to properties of the shape under study. The first stage of the algorithm has two parameters: the maximal distance of accumulation (d_{acc}) and the threshold on the confidence score (t). As illustrated in Figs. 7 and 8, the method is not very sensitive to these two parameters and they can be chosen arbitrarily in a wide range of values. Significant changes are only visible in the output if we choose a too small distance of accumulation ($d_{acc} = 1$) or a too high threshold on confidence map ($t = 0.8$). The latter disconnects some branches of the *playmobil* tree. A rule of the thumb is to choose for parameter d_{acc} a value greater than the maximal possible radius, and to set $t = 0.5$.

In the same way the parameters of the second stage of the algorithm, which are the dilatation radius (d_r) and the geodesic step distance (d_g) have not a significant influence (if we omit extremal values). Finally, we have measured the effect of a scale change in the input mesh (equivalent to change the grid resolution during the accumulation process, i.e. to reduce the input point density). As expected, increasing the scale makes the computation time longer (with the

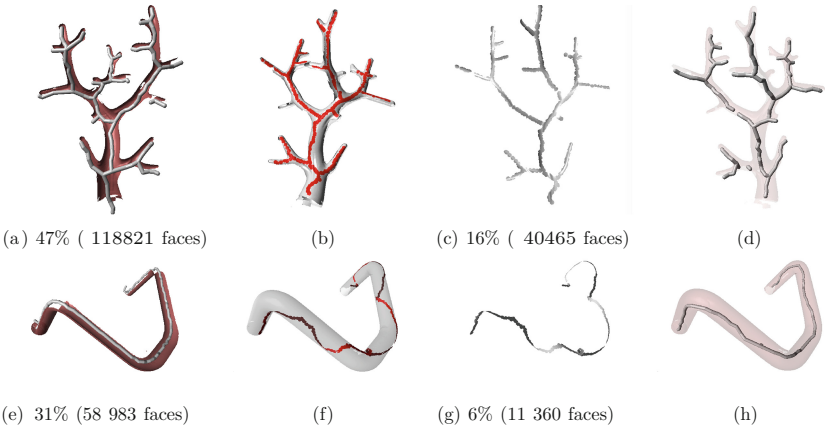


Fig. 9. Experiments on reduced scans by applying the centerline extraction on mesh with a very limited selection of faces. Images (a,e) show the result obtained with reduced scan parts (in red). Images (b,f) display the selected faces (isolated in images (c,g)), and the extracted centerlines are shown in images (d,h). (Color figure online)

increase of size of the accumulation digital space) but the resulting centerline is smoother and more accurate, and presents more details.

Robustness to missing parts in mesh scans. To conclude this section, the sensitivity to partial data in meshes was tested. Even if experiments of partial scans of real objects were given in the previous publication, we measure here extreme cases for the method with a major removal of information: first on a partial scan obtained by filtering the normal direction (see Fig. 9(a to d)), and second on a thin portion of the original mesh to measure the limitations of the method (see Fig. 9(e to h)). In both cases, the confidence map is remarkably stable, and the algorithm is able to approach the centerline located inside the original shape, with a slight loss in accuracy in the second experiment. Note that the optimization process was not included here but it could potentially significantly increase the quality of the result.

5 Conclusion and Discussions

We have presented in full details a method for centerline extraction from meshes. Algorithms, code organization, tools and dependencies were described, and the complete source code is available online. Reproducibility was demonstrated with complementary experiments, and a particular attention was devoted to the sensibity of the method to parameter tuning. The method was presented here only on triangulated mesh data type, but the method can easily be tailored to other

data types (like point clouds or digitized objects) from the provided source codes. The extension of the confidence map method to process volumetric grayscale images is a future challenging task and would have many applications in the medical imaging domain, with for instance the extraction of 3d vascular vessels or bronchial trees.

A Appendix

In this section we present in details the different algorithms that were mentioned in the method description. These algorithms were implemented almost as is in our C++ code. They are already proposed in another manner in the paper [9].

Algorithm 1. Global process that computes the normal vector accumulation image *imAcc* and the corresponding confidence image *imConf* from an input set of points and the corresponding normal filed.

procedure GLOBALPROCESS
Input
List<Point3D> sp ▷ List of surface points
List<Vector3D> nv ▷ Normal vectors of sp
Int d_{acc} ▷ Accumulation distance
Double r ▷ Morphological dilatation radius
Double d_{geo} ▷ Geodesic step distance
Double t_{conf} ▷ Threshold on imConf, in $[0, 1]$
Output
Graph *graph* ▷ Resulting centerline
Begin
Image3D<Int> $imAcc$ = COMPUTEACC(sp, nv, d) ▷ Accumulation image
Image3D<Double> $imConf$ = COMPUTECONF($sp, nv, imAcc, d_{acc}$) ▷ Confidence image
Image3D<Double> $imConf_T$ = THRESHOLD($imConf, t_{conf}$)
$graph$ = COMPUTEGRAPH($imAcc, imConf_T, r, d_{geo}$)
return *graph*
End

Algorithm 2. Compute the accumulation image $imAcc$ from a normal vector field nv with an accumulation distance d.

```
procedure ComputeAcc
Input
    List<Point3D> sp                          ▷ Surface points
    List<Vector3D> nv                         ▷ Normal vectors
    Int d_acc                                 ▷ Accumulation distance
Output
    Image3D<Int> imAcc                        ▷ Accumulation image
Begin
    for i : 0 → nv.size()−1 do
        Vector 3D norm = nv[i]
        Point3D orig = sp[i]
        Point3D pos = orig
        while orig.distanceTo(pos) < d_acc do
            imAcc(pos)++
            pos.translate(norm)
    return imAcc
End
```

Algorithm 3. Compute the confidence map $imConf$ for each accumulation value of $imAcc$ from each normal vector of nv contributing to this accumulation value.

```
 1: procedure ComputeConf
 2: Input
 3:     List<Point3D> sp                       ▷ Surface points
 4:     List<Vector3D> nv                      ▷ Normal vectors
 5:     Image3D<Int> imAcc                     ▷ Accumulation map
 6:     Int d_acc                              ▷ Accumulation distance
 7: Output
 8:     Image3D<Double> imConf                 ▷ Confidence map
 9: Begin
10:     for i : 0 → nv.size() - 1 do
11:         Vector3D norm = nv[i]
12:         Point3D orig, pos, maxPos = sp[i]
13:         Int maxAcc = 0
14:         while orig.distanceTo(pos) < d_acc do
15:             if imAcc(pos) > maxAcc then
16:                 maxAcc = imAcc(pos)
17:                 maxPos = pos
18:             pos.translate(norm)
19:         imConf(maxPos)++
20:     for all pos ∈ imConf.domain() do
21:         imConf(pos) = imConf(pos) ÷ imAcc(pos)
22:     return imConf
23: End
```

Algorithm 4. Compute the geodesic graph from an accumulation map $imAcc$ and a thresholded confidence map $imConf_T$.

procedure ComputeGraph
Input
  Image3D<Int> $imAcc$ ▷ Accumulation map
  Image3D<Double> $imConf_T$ ▷ Thresholded confidence map
  Double r ▷ Dilatation radius
  Double d_{geo} ▷ Geodesic step distance
Output
  Graph $graph$ ▷ Resulting graph
Begin
  Image3D<Bool> $imConf_B$ = binarize($imConf_T$,0)
  Image3D<Bool> $imConf_D$ = dilate($imConf_B$,r)
  Point3D P_{init} = maximumCoordinates($imAcc$)
  Image3D<Double> $imGeo$ = geodesicDistanceTransform($imConf_D$,P_{init})
  Map<Int,Vector<Point3D>> $regions$ ▷ Geodesic regions
  for all $pos \in imGeo$.domain() **do**
    $regions[floor(imGeo(pos)/d_{geo})]$.push(pos)
  Map<Int,List<Vector<Point3D>>> $regionCCs$ ▷ Connected components
  for all $key, region \in regions$ **do**
    List<Vector<Point3D>> CCs = splitIntoCCs($region$)
    $regionCCs[key]$.push(CCs)
  for all $key, CCList \in regionCCs$ **do** ▷ Graph building
    for all $CC \in CCList$ **do**
      Point3D $bary$ = barycenter(CC)
      $graph$.addVertex($bary$)
      for all $childCC \in regionCC[key+1]$ **do** ▷ key+1 repr. the next region
        if areConnected(CC,$childCC$) **then**
          $graph$.addEdge($bary$, barycenter($childCC$))
  return $graph$
End

References

1. CGal: release 4.8. http://www.cgal.org
2. DGtal: digital Geometry tools and algorithms library. http://dgtal.org
3. Geogram: release 1.0.0. http://alice.loria.fr/software/geogram/doc/html/index.html
4. Geogram vorpaview: online demonstration. http://webloria.loria.fr/levy/GEOGRAM/vorpaview.html
5. Dgtaltools: companion repository of DGtal library (2016). https://github.com/DGtal-team/DGtalTools
6. Dgtaltools-contrib: companion repository of DGtal library (2016). https://github.com/DGtal-team/DGtalTools-contrib
7. Aanjaneya, M., Chazal, F., Chen, D., Glisse, M., Guibas, L., Morozov, D.: Metric graph reconstruction fron noisy data. Int. J. Comput. Geom. Appl. **22**(04), 305–325 (2012). http://www.worldscientific.com/doi/abs/10.1142/S0218195912600072

8. Goodman, S.N., Fanelli, D., Ioannidis, J.P.: What does research reproducibility mean? Sci. Transl. Med. **8**(341), 341ps12 (2016). http://stm.sciencemag.org/content/8/341/341ps12
9. Kerautret, B., Krahenbül, A., Debled Rennesson, I., Lachaud, J.O.: Centerline detection on partial mesh scans by confidence vote in accumulation map. In: The proceedings of ICPR 2016 (2016, to appear)
10. Kerautret, B., Krähenbühl, A., Debled-Rennesson, I., Lachaud, J.-O.: 3D geometric analysis of tubular objects based on surface normal accumulation. In: Murino, V., Puppo, E. (eds.) ICIAP 2015. LNCS, vol. 9279, pp. 319–331. Springer, Cham (2015). doi:10.1007/978-3-319-23231-7_29
11. Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D.M., Lu, L., Yang, C.: On centroidal voronoi tessellation - energy smoothness and fast computation. ACM Trans. Graph. **28**(4), 32 (2009). Article No. 101, Presented at SIGGRAPH 2010
12. Lévy, B., Bonneel, N.: Variational anisotropic surface meshing with voronoi parallel linear enumeration. In: Jiao, X., Weill, J.C. (eds.) Proceedings of the 21st International Meshing Roundtable, pp. 349–366. Springer, Heidelberg (2012)
13. Nguyen, V.T., Kerautret, B., Debled Rennesson, I., Colin, F., Piboule, A., Constant, T.: Segmentation of defects on log surface from terrestrial Lidar data. In: soumis à ICPR 2016 (2016)
14. Rusu, R.B., Cousins, S.: 3D is here: point cloud library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011
15. Saha, P.K., Borgefors, G., di Baja, G.S.: A survey on skeletonization algorithms and their applications. Pattern Recogn. Lett. **76**, 3–12 (2016)
16. Sethian, J.A.: Fast marching methods. SIAM Rev. **41**(2), 199–235 (1999). http://epubs.siam.org/doi/abs/10.1137/S0036144598347059
17. Tagliasacchi, A., Alhashim, I., Olson, M., Zhang, H.: Mean curvature skeletons. Comp. Graph. Forum **31**(5), 1735–1744 (2012). http://dx.doi.org/10.1111/j.1467-8659.2012.03178.x
18. Tagliasacchi, A., Delamé, T., Spagnuolo, M., Amenta, N., Telea, A.: 3D skeletons: a state-of-the-art report. Comput. Graph. Forum **35**(2), 573–597 (2016)
19. Yan, D., Lévy, B., Liu, Y., Sun, F., Wang, W.: Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In: ACM/EG Symposium on Geometry Processing/Computer Graphics Forum (2009)