Editing Operators for Cross-Sectional Data-Sets

Ioannis Kyriazis[1] , Ioannis Fudos[2]

[1]University of Ioannina, Greece, kyriazis@cse.uoi.gr
[2]University of Ioannina, Greece, fudos@cse.uoi.gr

Corresponding author: Ioannis Kyriazis, kyriazis@cse.uoi.gr

Abstract. We introduce a tool suite of editing operators for 3D models of free-form objects, which apply on cross-sectional data-sets with the form of generalized cylinders. The objective is to provide an automated framework that allows a model to be modified according to the user intentions. A new feature is extracted from the 3D model, called the curve of centroids, which is used as a basis for applying modifications on the object. A series of editing operators are defined, which use this curve for performing high level editing on the model. A set of user-defined parameters is used to control the editing process, making the new set of operators easy to use. We present some examples on how these editing operators can be applied on a series of data-sets in various situations, ranging from reverse engineering and digital reconstruction to medical simulations, or for creating works of art.

Keywords: Editing, Operators, 3D Models, Cross Sections, Free Form, Generalized Cylinders
DOI: https://doi.org/10.14733/cadaps.2019.1112-1126

1 INTRODUCTION

Editing 3D objects can apply to many situations and for various purposes, such as Computer Aided Design, Digital Reconstruction, medical simulations, education, gaming, or for creating art works.

There are many methods proposed in the literature, which offer editing of several types. In most cases, these methods require as input a model that has previously been structured in a specific format, either in triangle meshes with specified topology [7, 9], or in higher level representations, such as curves or surfaces [3, 15, 30, 31].

In another domain of research, several methods have been proposed for medical applications [2, 6, 10, 14, 21, 27, 29], which mainly address the issue of reconstructing the surface of a 3D model, but do not provide any tools for editing the model.

This work is an effort to bridge the gap between these two domains, by providing a framework that offers editing of 3D models, which can handle unstructured input, and could potentially be used for applications such as medical simulations. In this paper, we report on the development of a cross-sectional 3D model editor. The aim of this work is to provide an editing framework that uses high level parameters and allows for applying complex parametric deformations with a single transformation.

The models we work with have properties often encountered in medical data-sets (e.g. CT scans of arteries or internal organs). However, our tool-set can be applied to both engineered objects (e.g. mechanical parts) or free-form objects (e.g. human or animal figures). We have developed a set of editing operators for cross-sectional data sets, which takes as input a 3D point cloud that represents a generalized cylinder. Our method automates the process of selecting the points of one or more cross sections with the aim to perform modifications on these points.

The point cloud may be either structured (e.g. from a CT scan), or unstructured (e.g. from a laser scanner). In the first case, where the input is structured, a series of B-Spline contours is computed to represent the boundaries of the cross sections, and the surface of the model is subsequently reconstructed. In the second case, where the input is not structured, we extract the required information by (a) slicing the point cloud into cross sections, (b) projecting the points of each slice to its corresponding cross section, and (c) producing a G^0 feature poly-line that represents the boundary of the cross sections, which is thereafter interpolated by a closed, smooth and continuous G^1 B-Spline curve. This step uses properties of the convex hull and the Voronoi diagram and was described in detail in our previous work [18].

A suite of editing operators has been developed, using high level feature information. These operators can be applied either locally to selected parts of the model, or globally to the entire model. A set of free-form transformations have been presented in our previous work [17], which deform parts of the model using homogeneous transformation matrices. However, their applicability is limited as their complexity makes them unfit for end-users of CAD systems.

In our current work we have built on a new feature called the *curve of centroids*, and introduce a new set of editing operators that makes use of this curve. A set of user-defined parameters is used to control the editing process, making the new set of operators easy to use.

The rest of this paper is structured as follows. Section 2 discusses related work. Section 3 describes the form of objects we handle as input, and how we manage to build editable CAD models from unstructured data-sets. In section 4 we provide details on the types of editing we offer, as well as some limitations we need to take into account. Section 5 offers examples on how our editing tools can be applied to both free-form and medical data-sets, and section 6 compares the results of our tool suite with other commercial tools. In Section 7 we provide conclusions and some future research directions.

2 RELATED WORK

A variety of methods have been proposed that deal with editing 3D objects using several approaches. While editing free-form objects can be done with arbitrary operations, there are some special cases in which the editing process may require specific restrictions to be satisfied. One such case is that of medical simulations, where the editing process of a medical data-set needs to comply with the behavior of the real tissue when a surgery is actually performed.

Most of the free-form editing methods involve editing by hand. In these methods [3, 7, 12, 15, 22, 26, 30, 31], the user is required to draw some strokes or sweeps upon the model. These sweeps or strokes are then used to perform deformations on the model, by dragging the nearby triangles according to the user intention.

Some of them also offer the blending of surfaces, while preserving a smooth connectivity, to transfer a feature from one model to another [3, 7, 9, 12, 26, 31].

The methods discussed in [3, 30] use generalized cylinders and offer editing in the form of deformations on the skeleton of the model.

All these methods require as input a model that has previously been structured in a specific format, either in a triangle mesh with specified topology [7, 9], or in higher level representations, such as curves or surfaces [3, 12, 15, 30, 31].

Our method provides tools for editing the input model and also can handle unstructured data-sets (i.e. point clouds). We also have the means of extracting the required topology information with the use of a tool-set that was previously described in [18]. Our method also uses generalized cylinders, and offers editing in the form of deformations on the skeleton of the model, as in [3, 30], but besides allowing arbitrary deformations and editing by hand, our method offers a tool-set of automated operators, controlled by high level parameters.

3 PROPERTIES OF THE INPUT MODEL

The input for our tool suite may be as a simple as an unstructured point cloud. The only information a point cloud carries is the $[x, y, z]$ coordinates of a set of points that lie on the boundary of the model. Depending on the acquisition method used and the density of the scan, a point cloud may describe the topology of the boundary accurately at all parts of the model (e.g. 3D laser scanners), or describe only feature points of the object that lie on specific parts of the model (e.g. medical CT scans). The case of medical CT scans is considered sensitive, as the direction and the density of the scan are predefined, and cannot be refined after the data-set has been acquired.

For simplicity of the computations, all point clouds used in this work have a common characteristic, i.e. they are all some type of generalized cylinders. In other words, our models do not have parts that form branches, and also there are no holes present (all objects are of genus 0). For this type of objects, the complexity of constructing a skeleton for our model is minimal.

However, the user requirements usually demand the processing of complex objects, which may have holes in their surface (genus $>$ 0), or have a complex geometry with branches and parts that cannot be described with generalized cylinders. In such cases, the method we propose will not work as it is.

But we do have the option to split such objects into smaller parts to form generalized cylinders and treat each part as a discrete object. For example, we could adopt a method such as the one proposed by [4], which is also discussed in [1], that uses a Reeb Graph to determine the regions where a model could be segmented, to acquire perceptually meaningful components that suit our needs. This, however, may be done as a pre-processing step before we apply our method, and therefore is not discussed here.

To extract an editable CAD model from unstructured point clouds, we may use the method we have previously described in [18], which offers an option to divide the 3D point cloud in cross-sectional slices, and treat each slice as an individual 2D point set. Using properties of the convex hull and the Voronoi diagram, we define a set of representative feature points that describe the boundary of the slice points accurately. These feature points form a closed curve of continuity G^0, which is called a feature poly-line. Subsequently, a G^1 B-Spline is constructed, which interpolates the poly-line and provides a smooth boundary representation. The contours of all cross sections are combined, and a number of points (depending on the required level of detail) are selected on each B-Spline, which are subsequently used for creating a mesh that represents the reconstructed surface with an editable model. This method is partially based on other proposed methods [5, 11, 23, 25], which are often described as lofting, skinning, or surface subdivision techniques.

In case the point cloud that came as input was acquired from a medical CT scan, like the example shown in Fig.1 (left), it will probably be already sliced into cross sections, and the slice points will already form a feature poly-line. In this case we can omit all computations up to this point and define G^1 B-Splines from these point-sets. Or we may ignore the existing topology provided by the input and rearrange the model according to a different slicing direction.

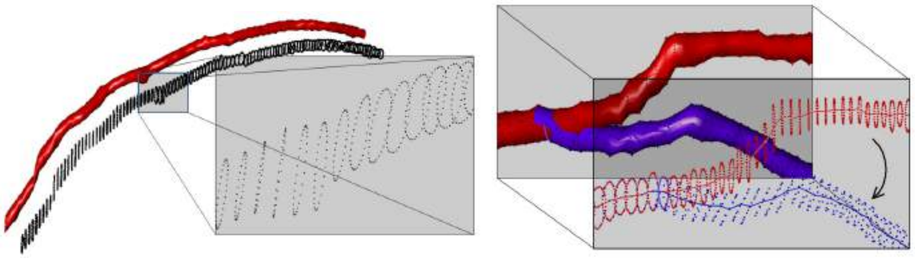


Figure 1: (Left) The model of an artery, and the points that came as input. (Right) The curve that interpolates the centroids of the slices can be edited to perform deformations on the model (Red: initial model - Blue: deformed model).

4 EDITING OPERATORS

Free-form editing usually includes mesh processing, morphing, deformations etc. It may also include the trivial case of modifying individual points, but this type of editing is considered low level editing and does not provide the user with an adequate editing tool-set.

Our method offers high level free-form editing tools, and also a set of operators that perform deformations on the model. To facilitate editing we derive a skeleton of the model by means of a NURBS curve that interpolates all slice centroids. This enables the application for local or global transformations, using this skeleton as a reference. These transformations may be applied as a whole, or change as we move along the skeleton. We call this skeleton the *curve of centroids*.

The editing process is implemented with the use of transformation matrices. All deformations (rotations, translations, scaling) are achieved by multiplying each selected centroid with the corresponding transformation matrix.

4.1 The Curve of Centroids

The curve of centroids is defined as a cubic B-Spline that interpolates the centroids of each cross section. To derive this curve, we compute the centroid of each cross section, i.e. the mean point of the slice points for each slice. We compute all the centroids and then we compute a cubic B-Spline that interpolates these centroids. We use a cubic B-Spline because it is easy to compute and capable of representing adequately most 3D shapes. An example is illustrated in Fig.1 (right).

The curve of centroids is similar to the centerline described in [13, 19, 20, 24]. However, the two representations should not be considered as the same feature, as the point sets used for the computation of each curve are different: The centerline is extracted directly from all points of the CT scan, while our curve of centroids is computed from the final filtered model. In other words, the curve of centroids is indirectly evaluated from the CT scan, which means that it might have some differences from the centerline.

The newly acquired curve of centroids can be used to apply modifications to the model. Instead of applying transformations on the surface of the model, we may easily deform the curve of centroids. The surface will adopt the deformation of the curve, as the slice points of each cross section will constantly maintain their relative positioning around their centroid. This means that if e.g. a centroid is translated to another position, the corresponding slice points will also follow the transformation.

4.2 Alignment Operator

First of all, we define the *alignment operator*, which reorganizes the slices according to the direction of the curve of centroids. The operator is applied to one or more selected cross sections, and its function is to rotate the selected slices around their corresponding centroids, so that the normal of the plane defined by the slice points coincides with the gradient direction of the curve of centroids at each centroid. This operator is performed by applying an affine transformation matrix $R_L(\theta)$ that rotates the slice points around their centroid towards the gradient direction L of the curve of centroids:

$$R_L(\theta) = T(\vec{C_i}) \cdot A(\vec{v})^{-1} \cdot R_z(\theta) \cdot A(\vec{v}) \cdot T(-\vec{C_i}) \text{ where } A(\vec{v}) = R_y(-\theta_2) \cdot R_x(\theta_1).$$

The transformation includes a translation $T(-\vec{C_i})$ of the centroid C_i to the axis origin, followed by an alignment $A(\vec{v})$ of the gradient vector $\vec{v}$ at C_i with the identity vector $\vec{k}$ on the z-axis, then a rotation $R_z(\theta)$ of angle θ around the z-axis to align it to the desired direction L, followed by an alignment $A(\vec{v})^{-1}$ of the identity vector $\vec{k}$ on the z-axis back to the gradient vector $\vec{v}$, and a translation $T(\vec{C_i})$ of the axis origin back to the centroid C_i. Such a transformation is standard for aligning an object to a specified direction, and is typically encountered in graphics textbooks (e.g. examples 3.13, 3.14 in [28]).

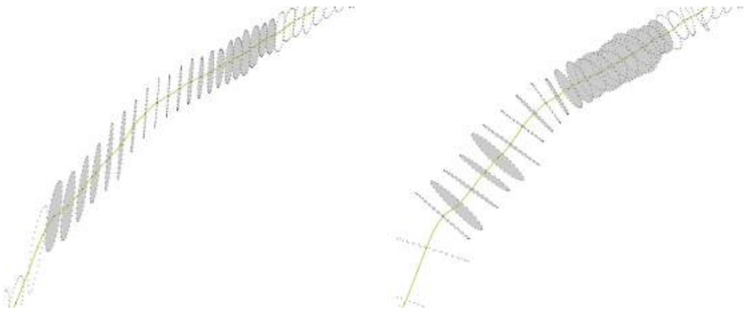


Figure 2: Alignment of the selected cross sections to the curve of centroids.

This operator is essential when editing the model, as it can improve the quality of the modifications or correct possible errors that occur during the editing process. It could even be used in cases where the slicing direction of the input model has not been satisfactory for the entire model. In Fig.2 (left) the initial positioning of the slices does not correspond to the direction of the curve of centroids as required. We can restore their positioning by aligning them to the curve as illustrated in Fig.2 (right).

As we notice on the resulting model, the volume is not preserved with this deformation. This, of course, is a violation to the user requirements in most cases. However, the parameter we are interested in is the connectivity among the slices. For this reason, we did not concern about volume preservation in our demo implementation. Of course, in case someone would like to release an application with a full implementation, obviously the volume preservation of the model would be an essential requirement, which could be roughly achieved with a simple non-uniform scaling, based on the old perpendicular to the plane and the tangent to the curve, probably the cosine of these two vectors, which should approximately keep the volume at least similar. After all, the example of Fig.2 shows an extreme case, which is unlikely to happen in real situations. In practice, we only apply this operator on parts outside e.g. a displacement, to maintain a smooth connectivity among the cross-sections, and the re-alignment does not deform the slices extremely.

The editing operators described in the following section can be applied either in conjunction with the alignment operator, or without it. In case they are applied exclusively, the slices of the modified region will remain parallel after the transformations.

4.3 Displacement Operator

The *displacement operator*, as its name states, performs a displacement on a selected part of the model, and the connectivity between the modified part and the rest of the model is updated properly. First of all, one or more cross sections have to be selected. Then, a series of transformations are applied to the selected centroids, causing the selected region to deform. As the curve of centroids changes in shape, the surface of the model also changes, to preserve its relative position to the curve of centroids. The modification may be any user-specified transformation.

As illustrated in Fig.3 the user has selected some centroids (from C_1 to C_2) and translates them upwards on the z-axis. To maintain a smooth connectivity at the endpoints, a number of centroids outside the selected region is also modified (i.e. C_0 to C_1 and C_2 to C_3), applying a proportional transformation which brings these centroids closer to the modified endpoints. The number of centroids outside the selected region, which are affected by a transformation, is specified by the user as a parameter during the editing process.

The alignment operator has also been applied to the slices with centroids from C_0 to C_1 and from C_2 to C_3. Algorithm 1 describes the details of the displacement operator.

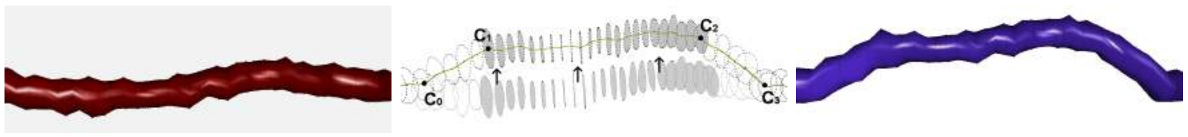


Figure 3: Translation using the displacement operator. (Left) The initial model in red. Center: Details of the operator. (Right) The edited model in blue.

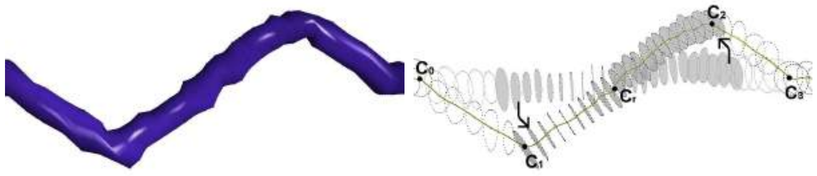
Algorithm 1: The algorithm for the Displacement Operator

```
Input: Centroids $C_1$, $C_2$, Translation vector $v \leftarrow [t_x, t_y, t_z]$
Parameter: $n$ ;                    // num of centroids affected outside selected region
for each centroid $C_i$ from $C_0$ to $C_1$ do
    $n \leftarrow |C_1 - C_0|$ ;
    $h \leftarrow \frac{i}{n}$ ;
    $C'_i \leftarrow C_i \cdot h \cdot v^T$ ;                    // proportional translation
end
for each centroid $C_i$ from $C_1$ to $C_2$ do
    $C'_i \leftarrow C_i \cdot v^T$ ;                             // full translation
end
for each centroid $C_i$ from $C_2$ to $C_3$ do
    $n \leftarrow |C_3 - C_2|$ ;
    $h \leftarrow 1 - \frac{i}{n}$ ;
    $C'_i \leftarrow C_i \cdot h \cdot v^T$ ;            // inverse proportional translation
end
```

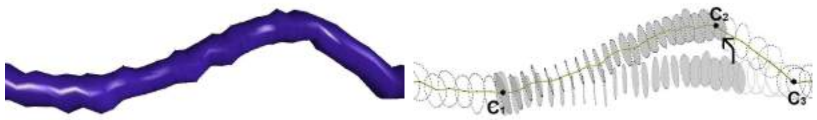
The displacement operator does not only work for translation, but also for rotation and scaling, and any given transformation. All that is needed is to replace the translation vector v by another transformation matrix R or S. For example, a rotation of angle θ around the z-axis with a point of reference C_r would be

$$R = T(\vec{C_r}) \cdot R_z(\theta) \cdot T(-\vec{C_r})$$

In the examples of Fig.4 we have two rotations around the axis defined by the gradient of the curve of centroids at the reference point, i.e. the center of the rotation. In Fig.4(a) the point of reference is $C_r = (C_1 + C_2)/2$, while in Fig.4(b) it is $C_r = C_1$. The point of reference is not affected by the rotation, so in the cases where one of the endpoints is the point of reference, there is no need to modify any cross sections outside the region at the fixed endpoint.



(a) The selected region is rotated around $C_r = (C_1 + C_2)/2$, on the axis defined by the gradient of the curve of centroids at C_r.



(b) The selected region is rotated around C_1, on the axis defined by the gradient of the curve of centroids at C_1. The cross sections left of C_1 are not affected by the transformation.

Figure 4: Rotations using the displacement operator.

4.4 Elastic Operator

This operator modifies the selected part of the model like an elastic rubber-band that is being stretched. This operation is applied to a selected part of the model, from centroid C_1 to C_2, causing the selected region to become tight and approximate the line segment defined by the endpoints to a certain extent. The selected centroids are translated by a parameter α that represents the amount of force being applied to the selected region, causing it to stretch. This parameter corresponds to the tension on the curve, and is expressed as a percentage that defines how much closer the curve of centroids will get to the segment of endpoints. Tension 0% means the curve remains unaffected, while 100% tension means that the curve of centroids degenerates to the line segment defined by the two endpoints. Fig.5 offers a visual explanation of the parameter α and algorithm 2 shows the formula which brings the centroids to their final positions. Fig.6 shows a region being stretched at 100%.

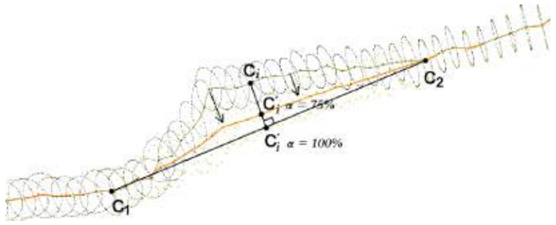


Figure 5: The centroid C_i is getting closer to the line segment $[C_1, C_2]$, as the parameter α increases. For $\alpha = 0$ we have $C_i' = C_i$, and for $\alpha = 100\%$ the centroid C_i is projected on the line segment $[C_1, C_2]$.

```
Algorithm 2: The algorithm for the Elastic Operator
  Input: Centroids $C_1$, $C_2$
  Parameter: $\alpha$ ;                                  // tension as percentage %
  for each centroid $C_i$ from $C_1$ to $C_2$ do
      $C_i' \leftarrow (1-\alpha)\cdot C_i + \alpha\cdot C_1 + \alpha\cdot(C_2 - C_1)\frac{(C_2 - C_1)\cdot(C_i - C_1)}{||C_2 - C_1||}$
  end
```

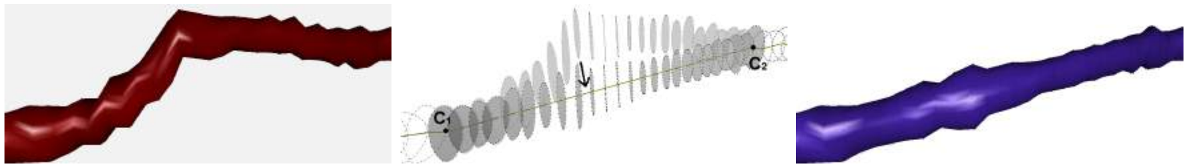


Figure 6: Deformations with the elastic operator. (Left) The initial model. (Center) The transformation. (Right) The edited model.

4.5 Inflation Operator

This operator modifies the selected part of the model like a balloon being inflated. In this case the selected centroids are not affected by the transformation, but the slice points of each cross section are subjected to a scaling transformation around their centroids. Again, to preserve a smooth transition between the cross sections outside the selected region, we enforce a number of additional centroids to follow the transformation proportionally.

Except for the percentage of scaling, we define two more parameters, i.e. the number of cross sections affected outside C_1 and C_2. This time we have two parameters with different values, as the number of cross sections affected by the proportional transformation may be different in the two sides of the selected region. Fig.7 shows an example of the inflation operator and algorithm 3 describes the process of applying the operator to the selected centroids. C_0, C_1, C_2 and C_3 are specified as parameters by the user, along with the scaling factor s.

Algorithm 3: The algorithm for the Inflation Operator

```
Input: Centroids $C_1$, $C_2$, Scaling factor $s$
Parameter: Centroids $C_0$, $C_3$ ;          // centroids affected outside selected region
for each centroid $C_i$ from $C_0$ to $C_1$ do
    $n_1 \leftarrow |C_1 - C_0|$ ;
    $h_1 \leftarrow \frac{i}{n_1}$ ;
    for each point $P_j$ in cross section $i$ do
        $P'_j \leftarrow P_j \cdot h_1 \cdot s$ ;                    // proportional scaling
    end
end
for each centroid $C_i$ from $C_1$ to $C_2$ do
    for each point $P_j$ in cross section $i$ do
        $P'_j \leftarrow P_j \cdot s$ ;                                  // full scaling
    end
end
for each centroid $C_i$ from $C_2$ to $C_3$ do
    $n_2 \leftarrow |C_3 - C_2|$ ;
    $h_2 \leftarrow 1 - \frac{i}{n_2}$ ;
    for each point $P_j$ in cross section $i$ do
        $P'_j \leftarrow P_j \cdot h_2 \cdot s$ ;           // inverse proportional scaling
    end
end
```

where the scaling factor s around the centroid C_r could be expressed as a transformation matrix

$$s = T(\vec{C_r}) \cdot S \cdot T(-\vec{C_r})$$

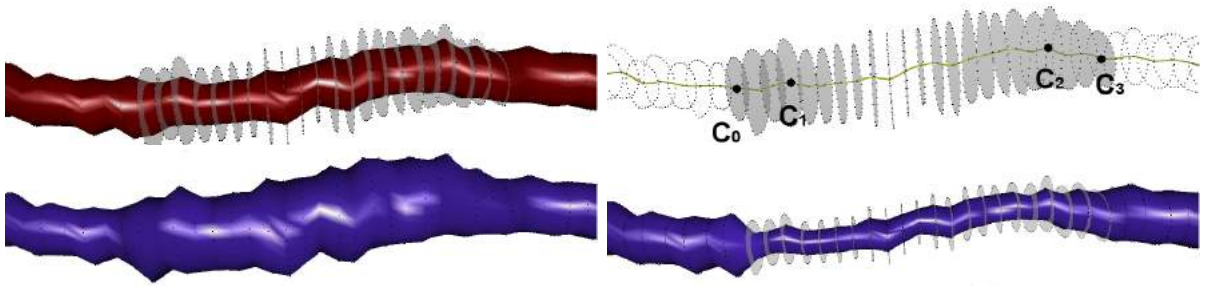


Figure 7: In the inflation operator, the curve of centroids remains unaffected by the transformation, but the points of the selected cross sections are scaled around their centroids. (Top Left) The initial model, with the selected cross sections inflated. (Top Right) The transformation: From C_1 to C_2 the cross sections are completely scaled. From centroid C_0 to C_1 and from C_2 to C_3, scaling is applied proportionally. (Bottom Left) The inflated model. (Bottom Right) For $s < 1$ the operator causes deflation.

4.6 Placing the curve of centroids on a parametric function

Another type of operator, which is based on a parametric function, a feature previously described in [17], applies a transformation that causes the cross sections to deform according to a function or a user specified mathematical expression. This option produces complex deformations with a little effort, as a single transformation may provide different modifications to the cross-sections, depending on their index in the selected region. Fig.8 shows such a deformation, where the model of the artery is placed on a sine function $f(h) = 3sin(10h\pi)$ (h is the index of the cross section inside the list of selected cross sections). The local features and the details of the surface are preserved, but the model is completely reshaped. Algorithm 4 demonstrates how to use the parametric function for the operator.

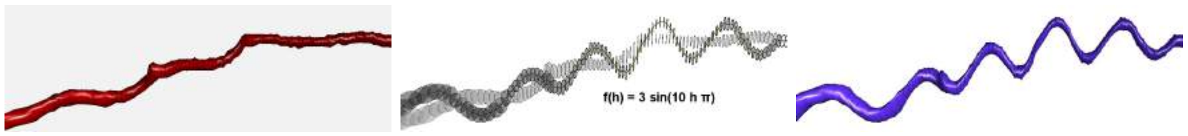


Figure 8: Placing the curve of centroids on a parametric function. Each cross section is positioned on a path defined by a mathematical expression according to its index in the selected region. Here the curve of centroids is placed on a *sine* function. (Left) The initial model. (Center) The transformation. (Right) The edited model.

Algorithm 4: The algorithm for the Parametric Function Operator

```
Input: Centroids $C_1$, $C_2$
Parameter: Function $f(h)$
for each centroid $C_i$ from $C_1$ to $C_2$ do
    $n \leftarrow |C_2 - C_1|$ ;
    $h \leftarrow \frac{i}{n}$ ;                    // h: index of cross section inside selection
    $C'_i \leftarrow C_i \cdot f(h)$ ;                                // parametric function
end
```

The user has also the option to define his own curve of centroids, by choosing a set of feature points by providing their $[x, y, z]$ coordinates, and then interpolate a curve to these points. The existing curve of centroids is then replaced by the user defined curve and the cross sections are positioned accordingly, causing the model to deform in the shape that the user has sketched. The relative positions of the centroids on the new curve are calculated using the arc length [8], which places the first centroid on the beginning of the curve, the last centroid on the end of the curve, and all centroids in between, according to their relative length from the old curve to the new. In the example of Fig.9, the user has defined a set of feature points on R^3, and computed a curve that interpolates these points in the form of a helix. The artery model was then placed on this helix curve of centroids, causing the entire model to stretch in the shape of the helix.

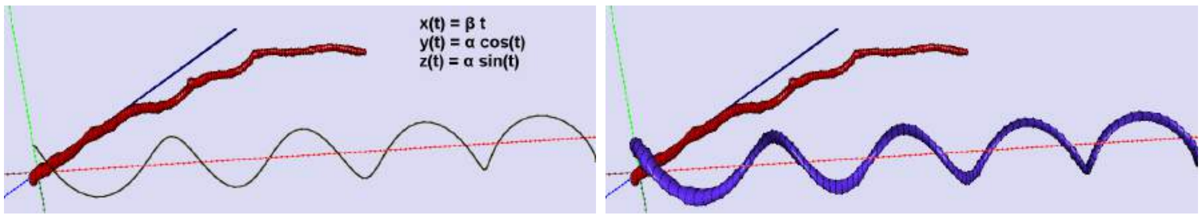


Figure 9: Placing the curve of centroids on a parametric function. (Left) The user defines a set of feature points to form a helix in R^3 and interpolates a NURBS curve on these points. (Right) The curve of centroids of the model is placed on this curve, causing the model to deform in the shape of the helix.

4.7 Cross-Sectional Free-Form Editing

A more generalized and unrestricted variation of the editing operator is the one that applies deformations to the model without any consideration about the curve of centroids. So far, when a transformation was applied, the reference points for the transformation would always be the centroids of the selected cross sections, or one point calculated from the selected centroids when many cross sections are selected. But the user is also allowed to choose an arbitrary point of reference to apply a transformation, to obtain different results. One simple idea would be to apply a transformation around the axis origin, or around a specified point on a particular axis. The surface points of the selected cross sections would then be modified in a different shape, providing another resulting model. This operator can be used to refine local details or correct any flaws on the surface of the resulting model. An operator like that has already been described in [17], where the models where not from medical datasets, but free form objects like those in Fig.11. The variation we discuss here includes the option to use any point as center for the transformation, and not only the axis origin, or the centroid of a cross section.

4.8 Limitations

The editing operators we described in this paper may be used in free-form editing of medical data-sets, such as arteries, to produce models of any possible shape and properties. The curve of centroids is computed from the surface points of the initial model, and when it is modified, it re-defines the shape of the model. The operators discussed above make use of this feature, and when applied to a model, provided some additional restrictions, may potentially produce models that simulate the behavior of real arteries in a surgical operation.

The restrictions when applying these operators regard the resulting models rather than the actual modifications. In case of modeling human tissues, the physical properties of these tissues have to be preserved during the editing process. This means that a model cannot be modified in such a way that would deform a tissue beyond its structural capabilities. In other words, even if an arbitrary transformation can be applied on the model, the resulting model has to satisfy the user intentions, and generally have a useful meaning. In this work, however, we do not make any suggestions to comply with such restrictions. Additional research is required for this issue, which includes the collaboration with medical experts.

Furthermore, not all the operators presented here would be valid for use in medical simulations. The operator with the parametric function, for example, could preferably be used to create works of art. Even in such cases, the user should keep in mind not to perform extreme modifications that could, for example, cause the cross-sections to overlap. Such issues, however, are outside the scope of this work and therefore are not discussed here.

5 EXAMPLES

To test the editing operators we described in this paper we have implemented our own geometric modeling software, under the Microsoft Visual C++ programming environment, using the Qt UI framework. The rendering process is done with OpenGL, and some calculations concerning computational geometry are done with the Qhull library.

The editing process is performed in real time, meaning that the user has the model at their disposal and performs editing by applying transformations on the selected cross-sections. The extraction of the model and the surface reconstruction, which have been discussed in [18], are done with a minimal human interaction, and generally require $O(n \log^2 n)$ time (where n is the number of points in the cloud).

The editing operators we described in section 4 can potentially be adopted by medical applications, such as simulations of surgical operations. An interesting example of a medical application is the bypass surgery, where a segment from an artery or vein from elsewhere in the patient's body is grafted to the coronary arteries to bypass atherosclerotic narrowings and improve the blood supply to the coronary circulation supplying the heart muscle. When a doctor performs such a surgery, they need to calculate the path of the artery in advance, before the segment is placed. With our editing operators, a possible path can be calculated by modifying the curve of centroids according to the doctors instructions. An example is illustrated in Fig.10 (left).

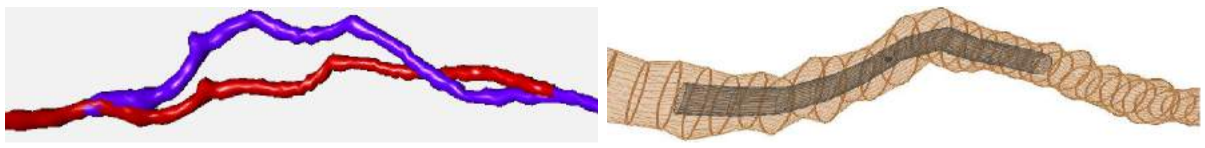


Figure 10: Examples of modification of human tissues. (Left) The path of an artery can be redefined by editing the curve of centroids. This is a key feature for medical applications such as the bypass surgery. (Right) The surgical operation of angioplasty often includes a stent insertion. When a stent is inserted, the tissue of the artery is both stretched, as described with the elastic operator, and also inflated, as described with the inflation operator.

Another example of medical application is the angioplasty, a technique of mechanically widening narrowed or obstructed arteries, the latter typically being a result of atherosclerosis. A balloon catheter is passed into the narrowed locations to crush the fatty deposits, opening up the blood vessel for improved flow, and is then withdrawn. A stent may be inserted at the time of ballooning to ensure the vessel remains open. An example is shown in Fig.10 (right). If a stent is inserted, it would follow the same path inside the artery, as the curve of centroids, as it is a flexible object, and will bend as the walls of the artery apply pressure on it. But as the walls of the artery apply pressure on the stent, so will the stent apply pressure on the artery walls. Furthermore, as the stent is deployed, it pushes the artery walls outwards, widening the opening of the artery.

The first of these two actions will modify the curve of centroids at the region where the stent is inserted, causing it to stretch to some extent (as we described with the elastic operator), as long as the flexibility of the tissue allows for stretching. The second action has no effect on the curve of centroids, but it causes the artery walls to inflate (as we described with the inflation operator). The artery will behave as the model in Fig.5 and Fig.7.

The operators we described here could be used in medical data-sets. However, this doesn't mean that they are addressed exclusively to such applications. They could be applied to other domains as well. For example, we could apply the displacement operator or the inflation operator on the drill bit model of Fig.11(a), to deform the model in various ways. Some of these models may not have useful meaning, but they may be valid as artworks.

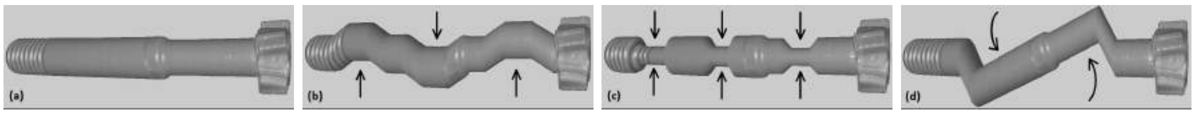


Figure 11: (a) The model of a drill bit, (b) Using the displacement operator to translate parts of the model, (c) Using the inflation operator to deflate parts of the model, and (d) Using the displacement operator to rotate a part of the model around its center.

A video with a visual demonstration of the operators described in this paper is also available in [16].

6 VALIDATION - COMPARISON TO COMMERCIAL TOOLS

To validate the quality of the results of our editing tool-set, we tried to apply similar modifications on our model, using 3ds Max. Fig.12 illustrates such an example, in which we tried to apply a $sin(\theta)$ function, which is actually a translation of each cross section and creates the effect of a large wave.

Of course, one could also perform such operations in 3ds Max or other modeling packages. But the issue in such packages is that the operations are actually achieved by dragging the selected control points with the mouse. This may seem a user friendly solution, but it lacks in accuracy of the results. There is an option for setting specific transformations on selected control points, but this option requires the manipulation of individual points.

The operations we propose make use of high level parameters that control the transformation of the model, which have a meaning to the user. When editing by hand in other modeling packages, such parameters are not available, or have to be manually defined.

Additionally, 3ds Max and other modeling packages do not offer a direct tool for applying a proportional transformation, such as the one we have developed with the parametric function. Therefore, our editing tool offers a powerful feature, which allows for applying parametric transformations with a single deformation.

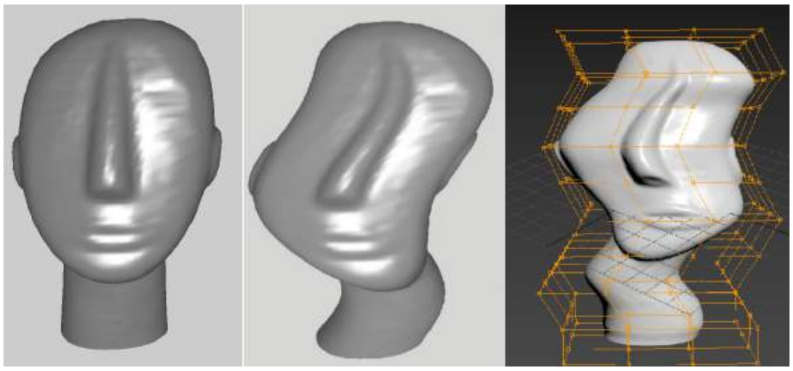


Figure 12: (Left) The model of a Cycladean idol. (Center) With our tool-set, editing is performed using high level parameters according to a parametric function. (Right) With 3ds Max, editing is done by manually deforming individual control points on the bounding box.

Tab.1 shows the comparison of our tool with the 3ds Max, a powerful tool which offers several editing tools, targeted in a variety of applications. But most of the tools require editing by hand, meaning that the modifications are achieved by selecting a part of the model, or a set of control points (eg. on its bounding box), and by dragging the selection with the mouse. Although there is an option to set specific values on the parameters, this type of editing is considered as editing by hand, and requires additional effort and expert skills.

Moreover, 3ds Max does not directly offer complex editing operators, such as those described in this paper, and certainly does not allow for editing according to a parametric function, such as we described in section 4.6.

Tool / *Feature*	*3ds Max*	*Our Tool*
Requires a specific type of input	No An unstructured point cloud is enough	No An unstructured point cloud is enough
Selecting points Grouping in regions	Yes Manual selection with the mouse	Yes Cross-sectional slices
Use feature points	Yes Control points of bounding box	Yes B-Splines interpolating cross-sectional feature poly-lines
Allows for free-form editing	Yes Manually dragging with the mouse	Yes Using a transformation matrix
Offers complex editing operators	No Editing is done by hand	Yes Using the curve of centroids
Editing with parametric functions	No	Yes Complex deformations with a single operation

Table 1: Comparison of our editing process with 3ds Max.

7 CONCLUSIONS

We have introduced a tool-set of editing operators for modifying free-form models of cross-sectional data-sets addressed to various applications, such as medical simulations. The modifications performed on the model resemble the deformation of an arterial tissue in a bypass or angioplastic surgery.

The editing operators we described, and especially the operator with the parametric function, allow the user to perform complex deformations with a single transformation.

Our current work allows the users to perform any modifications they desire arbitrarily, without any further robustness checks. In this context, a future extension is to allow imposing geometric constraints that have to be respected when applying modifications on a model.

ORCID

Ioannis Kyriazis, http://orcid.org/0000-0003-3781-4254
Ioannis Fudos, http://orcid.org/0000-0002-4137-0986

REFERENCES

[1] Agathos, A.; Pratikakis, I.; Perantonis, S.; Sapidis, N.; Azariadis, P.: 3d mesh segmentation methodologies for cad applications. Computer-Aided Design and Applications, 4(6), 827–841, 2007. https://doi.org/10.1080/16864360.2007.10738515.

[2] Albu, A.B.; Beugeling, T.; Laurendeau, D.: A morphology-based approach for interslice interpolation of anatomical slices from volumetric images. IEEE Transactions on Biomedical Engineering, 55(8), 2022–2038, 2008. https://doi.org/10.1109/TBME.2008.921158.

[3] Andrews, J.; Jin, H.; Sequin, C.: Interactive inverse 3d modeling. Computer-Aided Design and Applications, 9(6), 881–900, 2012. https://doi.org/10.3722/cadaps.2012.881-900.

[4] Antini, G.; Berretti, S.; del Bimbo, A.; Pala, P.: 3d mesh partitioning for retrieval by parts applications. In 2005 IEEE International Conference on Multimedia and Expo, 1210–1213, 2005. ISSN 1945-7871. https://doi.org/10.1109/ICME.2005.1521645.

[5] 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), 349–359, 1999. ISSN 1077-2626. https://doi.org/10.1109/2945.817351.

[6] Bors, A.G.; Kechagias, L.; Pitas, I.: Binary morphological shape-based interpolation applied to 3-d tooth reconstruction. IEEE Transactions on Medical Imaging, 21(2), 100–108, 2002. https://doi.org/10.1109/42.993129.

[7] Botsch, M.; Kobbelt, L.: An intuitive framework for real-time freeform modeling. ACM Transactions on Graphics, 23(3), 630–634, 2004. ISSN 0730-0301. https://doi.org/10.1145/1015706.1015772.

[8] Chen, X.; Yong, J.; Zheng, G.; Sun, J.: Automatic g^1 arc spline interpolation for closed point set. Computer-Aided Design, 36(12), 1205–1218, 2004. https://doi.org/10.1016/j.cad.2003.12.001.

[9] Chen, Y.; Cheng, Z.Q.; Li, J.; Martin, R.R.; Wang, Y.Z.: Relief extraction and editing. Computer Aided Design, 43(12), 1674–1682, 2011. ISSN 0010-4485. https://doi.org/10.1016/j.cad.2011.07.011.

[10] Chi, J.; Zhang, C.: Optimization of medical ct data with high precision. Computer-Aided Design and Applications, 10(1), 17–31, 2012. https://doi.org/10.3722/cadaps.2013.17-31.

[11] Edelsbrunner, H.; Mücke, E.P.: Three-dimensional alpha shapes. ACM Trans. Graph., 13(1), 43–72, 1994. ISSN 0730-0301. https://doi.org/10.1145/174462.156635.

[12] Gingold, Y.; Zorin, D.: Shading-based surface editing. ACM Transactions on Graphics, 27(3), 95:1–95:9, 2008. ISSN 0730-0301. https://doi.org/10.1145/1360612.1360694.

[13] Jandt, U.; Schafer, D.; Grass, M.; Rasche, V.: Automatic generation of 3d coronary artery centerlines using rotational x-ray angiography. Medical Image Analysis, 13(6), 846 – 858, 2009. ISSN 1361-8415. https://doi.org/10.1016/j.media.2009.07.010.

[14] Ju, T.; Warren, J.; Carson, J.; Bello, M.; Kakadiaris, I.; Chiu, W.; Thaller, C.; Eichele, G.: 3d volume reconstruction of a mouse brain from histological sections using warp filtering. Journal of Neuroscience Methods, 156(12), 84 – 100, 2006. https://doi.org/10.1016/j.jneumeth.2006.02.020.

[15] Kara, L.B.; D'Eramo, C.M.; Shimada, K.: Pen-based styling design of 3d geometry using concept sketches and template models. In Proceedings of the 2006 ACM symposium on Solid and physical modeling, SPM '06, 149–160. ACM, New York, NY, USA, 2006. https://doi.org/10.1145/1128888.1128909.

[16] Kyriazis, I.; Fudos, I.: Youtube video: Editing operators for cross-sectional data-sets. https://youtu.be/JeW-bhOk-GA.

[17] Kyriazis, I.; Fudos, I.: Building editable free-form models from unstructured point clouds. Computer-Aided Design and Applications, 10(6), 877–888, 2013. https://doi.org/10.3722/cadaps.2013.877-888.

[18] Kyriazis, I.; Fudos, I.; Palios, L.: Extracting cad features from point cloud cross-sections. Proceedings of the The 17-th International Conference on Computer Graphics, Visualization and Computer Vision, EUROGRAPHICS, Plzen, Czech Republic, 137–144, 2009. https://hdl.handle.net/11025/10899.

[19] Lesage, D.; Angelini, E.D.; Bloch, I.; Funka-Lea, G.: A review of 3d vessel lumen segmentation techniques: Models, features and extraction schemes. Medical Image Analysis, 13(6), 819 – 845, 2009. https://doi.org/10.1016/j.media.2009.07.011.

[20] Li, G.; Liu, L.; Zheng, H.; Mitra, N.J.: Analysis, reconstruction and manipulation using arterial snakes. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH ASIA), 29(5), Article No. 152, 1–10, 2010. https://doi.org/10.1145/1882262.1866178.

[21] Llamas, I.; Powell, A.; Rossignac, J.; Shaw, C.: Bender: a virtual ribbon for deforming 3d shapes in biomedical and styling applications. ACM Symposium on Solid Modeling and Applications, 89–99, 2005. https://doi.org/10.1145/1060244.1060255.

[22] Luffel, M.; Sati, M.; Rossignac, J.; Yoganathan, A.P.; Haggerty, C.M.; Restrepo, M.; Slesnick, T.C.; Kanter, K.R.; del Nido, P.; Fogel, M.A.: Surgem: A solid modeling tool for planning and optimizing pediatric heart surgeries. Computer-Aided Design, 70, 3 – 12, 2016. ISSN 0010-4485. https://doi.org/10.1016/j.cad.2015.06.018. SPM 2015.

[23] Nasri, A.H.; van Overveld, C.; Wyvill, B.: A recursive subdivision algorithm for piecewise circular spline. Computer Graphics Forum, 20(1), 35–45, 2001. https://doi.org/10.1111/1467-8659.00473.

[24] Schaap, M.; Metz, C.T.; van Walsum, T.; van der Giessen, A.G.; Weustink, A.C.; Mollet, N.R.; Bauer, C.; Bogunovic, H.; Castro, C.; Deng, X.; Dikici, E.; O'Donnell, T.; Frenay, M.; Friman, O.; Hoyos, M.H.; Kitslaar, P.H.; Krissian, K.; Kuhnel, C.; Luengo-Oroz, M.A.; Orkisz, M.; Smedby, O.; Styner, M.; Szymczak, A.; Tek, H.; Wang, C.; Warfield, S.K.; Zambal, S.; Zhang, Y.; Krestin, G.P.; Niessen, W.J.: Standardized evaluation methodology and reference database for evaluating coronary artery centerline extraction algorithms. Medical Image Analysis, 13(5), 701 – 714, 2009. https://doi.org/10.1016/j.media.2009.06.003.

[25] Sederberg, T.W.; Greenwood, E.: A physically based approach to 2-d shape blending. SIGGRAPH Comput. Graph., 26(2), 25–34, 1992. ISSN 0097-8930. https://doi.org/10.1145/142920.134001.

[26] Sorkine, O.; Cohen-Or, D.; Lipman, Y.; Alexa, M.; Rössl, C.; Seidel, H.P.: Laplacian surface editing. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, SGP '04, 175–184. ACM, New York, NY, USA, 2004. ISBN 3-905673-13-4. https://doi.org/10.1145/1057432.1057456.

[27] Sun, W.; Starly, B.; Nam, J.; Darling, A.: Bio-cad modeling and its applications in computer-aided tissue engineering. Computer Aided Design, 37(11), 1097–1114, 2005. ISSN 0010-4485. https://doi.org/10.1016/j.cad.2005.02.002.

[28] Theoharis, T.; Papaioannou, G.; Platis, N.; Patrikalakis, N.M.: Graphics and Visualization: Principles & Algorithms. A K Peters/CRC Press, 2008. https://dl.acm.org/citation.cfm?id=1554684.

[29] Weng, N.; Yang, Y.H.; Pierson, R.: Three-dimensional surface reconstruction using optical flow for medical imaging. IEEE Transactions on Medical Imaging, 16(5), 630 –641, 1997. https://doi.org/10.1109/42.640754.

[30] Yoon, S.H.; Kim, M.S.: Sweep-based freeform deformations. Eurographics Computer Graphics Forum, 25(3), 487–496, 2006. https://doi.org/10.1111/j.1467-8659.2006.00968.x.

[31] Zimmermann, J.; Nealen, A.; Alexa, M.: Silsketch: automated sketch-based editing of surface meshes. In Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling, SBIM '07, 23–30. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-915-9. https://doi.org/10.1145/1384429.1384438.