Toward 3D Selection and Skeleton Construction by Sketching

B. Fabianowski and J. Dingliana

Graphics Vision and Visualisation Group, Trinity College Dublin, Ireland

Abstract

Mesh deformations are generally specified by manipulating an underlying control structure. This may have tediously been built in a preprocessing step or computed ad-hoc from user sketches. We propose an extension to the latter approach, deriving three-dimensional representations of a skeleton and the surrounding region of interest from one or two simple strokes. This provides a substantial improvement over existing techniques, which build only two-dimensional structures on the fly. As a single planar sketch is not sufficient to fully describe a three-dimensional skeleton, its shape must be deduced by interpreting the input strokes in the context of the model. We explain the challenges involved, analyze several algorithms for interpreting the input and conclude by sketching out the direction of our ongoing work on the topic.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Modeling Packages I.3.6 [Methodology and Techniques]: Interaction Techniques

1. Introduction

Mesh deformation is an important component of 3D modeling and animation. It normally begins with a preprocessing stage, where the model is attached to a control structure. Deformations are then specified by manipulating this structure. Kho and Garland [KG05] propose to eliminate the explicit preprocessing. A reference stroke selects part of the model and also serves as its controlling entity. The mesh is deformed by matching the reference to a target stroke. This technique allows for the direct manipulation of unstructured polyhedral meshes. It yields impressive results, but is not without limitations. Most importantly, the control structure is two-dimensional, lying entirely in the view plane. This requires the camera to be locked to a single view per deformation. Complex adjustments must be broken into a more cumbersome sequence of steps. As the mesh is being deformed, self-intersections can occur that are then carried forward into subsequent steps, incrementally corrupting the result.

Our aim is to improve on the previous work by building more sophisticated structures on the fly. The goal is to construct fully three-dimensional representations of the selected region and its skeleton while retaining an intuitive, sketch-based interface. The skeleton can be used to deform a model in all three dimensions, eliminating the need to repeatedly build new control structures. A three-dimensional description of the selected region allows self-intersections to seamlessly be detected and resolved. Building on Kho and Garland's technique, we construct a set of cutting planes orthogonal to the skeleton. The first and last planes delimit the selected region. Cuts through the mesh surface by intermediary planes describe the selection's profile, expressing it as a generalized cylinder. An example is shown in figure 1.

We evaluate two forms of input. Inspired by [KG05], the first has the user draw a single stroke indicating a projection of the desired skeleton onto the mesh surface. The other form consists of two strokes, one specifying the beginning and one the end of the selected region. Our work focuses on the question of how a skeleton and a set of cutting planes orthogonal to it can be reconstructed from this input.

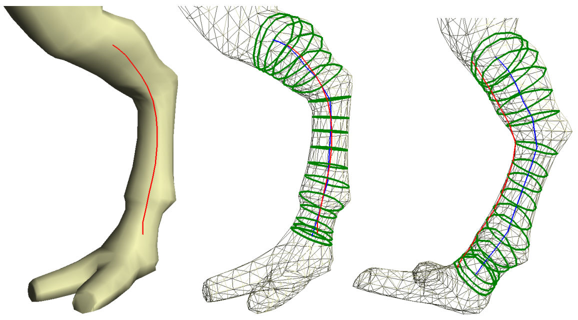

Figure 1: *Selection of a dinosaur's leg (red: stroke on mesh surface; green: generated cuts; blue: resulting skeleton)*

2. Related Work

The first step of mesh deformation generally is building a control structure and attaching the mesh to it. Different types of control structures have been proposed. An FFD lattice [SP86] is very flexible, but also tedious to control due to the large number of parameters it exposes. Wires [SF98] are located on the mesh surface and provide an interaction similar to armatures in sculpting. More common than either of these two is the skeleton. In traditional manual instrumentation [CGC*02], the user builds the skeleton and attaches the mesh to it by hand. While this offers great flexibility, it requires extensive input from the user.

To remove the need for manual input, a skeleton can automatically be extracted from the model. This is routinely done for virtual endoscopy [PWMZ95, LH00, BST05], but the algorithms only operate on volumetric data. A principle that is applicable to both volumetric [HF05] and mesh [MP02] representations is wave propagation. Starting from one or more points, waves travels through the model, producing topological rings on its surface. The rings' centers are then connected to yield the skeleton. Similar rings are generated and connected in Plumber [MPS*04] by sweeping a sphere through the tubular regions of a mesh. The two construction steps can also be reversed [LWTH01], building a skeleton first by incrementally collapsing mesh edges and only then sweeping a plane along it to better assess the shape of the model.

The latter approach allows the mesh to be separated into visually salient components. Skeleton extraction itself can also be based on this principle. In [OIN*06], the mesh is iteratively partitioned using the geodesic distances of its vertices from a random subset of points on the surface. Connecting the centers of the resulting salient parts then yields a skeleton. Smoother boundaries between the mesh components are achieved by probabilistic subdivision [KT03]. A related technique calculates the geodesic distances from a small number of feature points [TVD07] and then returns the dual Reeb graph of this mapping as the skeleton. Approaches derived from the medial axis transform [Blu67] use a Voronoi diagram of the mesh vertices and their connectivity information to build a skeleton [YBS03].

An issue affecting all fully automatic techniques is that the skeleton may not fit the user's needs. Whenever the user wishes to deform part of the mesh, a corresponding skeleton edge must be present. If none has been generated, deformation is not possible. The problem may be alleviated somewhat by guiding skeleton extraction via manual input. For wave propagation approaches, this corresponds to choosing the wave starting points by hand [LV99]. Approaches based on partitioning may allow the user to request further subdivision of mesh regions for which the skeleton is not sufficiently detailed [OIN*06]. The medial axis transform suffers from the additional problem that for a three-dimensional model, a 2D medial surface is actually produced. In the hybrid technique of [TT98], both issues are avoided at the expense of substantial manual input. The skeleton is constructed by hand and only its connection to the mesh automatically computed using the correspondence between skeleton and medial axis.

As a byproduct of automatic skeleton extraction, each vertex can be associated with the skeleton edge generated from it. When performing manual instrumentation, this has to be done by hand. For high quality results, every vertex is explicitly linked to an edge. A faster but less precise method is to let each skeleton edge influence the mesh surface within a given radius. Using the latter approach, deformations can easily be transferred from one mesh to another [LCJ93], or a combination of two meshes obtained by morphing [LV94].

Deformation techniques have more recently been developed that do not require the control structure to be built in a preprocessing step. It is instead derived ad-hoc from user sketches. As the structure is usually built only for part of the model, not only must mesh vertices be associated with it, but also the subset chosen that is to be affected. This region of interest is commonly either manually specified in an additional step [CJ06], or automatically computed to include all vertices within a given distance of the stroke [Buj06].

For FFDs, ad-hoc control of two-dimensional deformations is possible by sketching the reference and target shapes of a scalar field [HQ03]. Part of the model's silhouette can also directly be used as its control structure [NSACO05]. Other approaches build a linear structure similar to a skeleton by projecting the user's sketches onto a plane or the mesh surface. In the inspiration for this work [KG05], a reference stroke in the view plane serves as the controlling entity. The region of interest is automatically calculated using cutting planes perpendicular to the stroke at its beginning and end, but can manually be overridden. Strokes on the model's surface are used by [Buj06] and [OSSJ05] as control structures for local deformations affecting only a small region of the surface. A similar approach allows for the sketch-based construction of facial expressions [CJ06]. By building a volumetric region of interest around it, a stroke on the surface can be used as the control structure for larger spatial deformations [ZHS*05]. An interesting alternative for determining the region of interest is to select the salient part of the mesh covered by the user's stroke [ZMY06]. Departing from the sketching metaphor, an ad-hoc skeleton can be constructed directly in space using two 6 DOF input devices [LPRS05].

3. Overview

Our work is based on the ad-hoc principle of deriving a control structure from simple strokes sketched onto an unstructured mesh. Improving on existing techniques, we wish to construct a three-dimensional skeleton inside the model and a set of cutting planes orthogonal to it that spatially describe the region of interest. The user's input may be a single stroke expressing a projection of the skeleton onto the mesh surface. Alternatively, two strokes can be used that delimit the region of interest. Automatic connection of these strokes again yields a projection of the skeleton onto the surface. In both cases, projection depths and directions must be determined to reconstruct the skeleton. These should be based on the model's shape so that the skeleton be consistent with it.

We use the cutting planes to guide the projection. A plane is constructed for every anchor point obtained by densely sampling of the skeleton's representation on the surface. The skeleton is then built by projecting every anchor to the center of the corresponding plane's cut through the mesh. The key question is how the planes should be oriented to cut across the correct mesh region, be consistent with each other, avoid intersections between individual cuts and sudden changes in skeleton direction. As will be seen in the following sections, determining suitable plane orientations is a difficult and often ambiguous task. We treat it as an optimization problem, analyzing different optimization algorithms and criteria.

4. Single stroke

Inspired by [KG05], our initial approach is to have the user specify both the region of interest and its skeleton using a single stroke. This stroke directly provides a representation of the skeleton on the mesh surface. To project it inside the model, cutting planes are constructed at anchor points. Each cutting plane has three degrees of freedom. The first is the angle α it forms with the stroke in the tangential plane of the mesh surface, the second its tilt β relative to this plane. The third angle, rotating the cutting plane around its own normal, is irrelevant, as it has no influence on the resulting cut.

Intuitively, α should be chosen so that the cutting plane be perpendicular to the user's stroke, while β makes it cut across the region of interest. The latter requirement presents a difficultly as the region of interest is not actually known until after the cutting planes have been constructed. Suitable values of β must therefore be derived from user input and the shape of the mesh alone. A natural choice is the angle that aligns the plane with the surface normal at its anchor point in the stroke. Since the normal points away from the model, following it in the opposite direction can be expected to guide the plane toward the mesh center and, consequently, across. Unfortunately, normals are only indicative of the local shape of a surface. Figure 2 illustrates how on the left, surface normals correctly point the cutting planes across the model, while on the right, the normals are based on the shape of the bump and lead to intersecting, inconsistent planes.

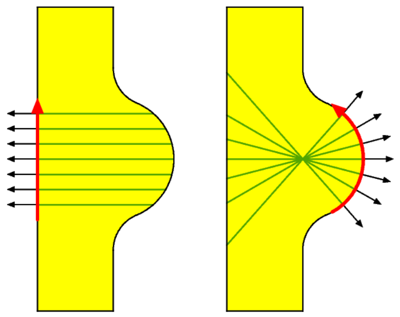
Figure 2: *Side view of cutting plane orientations derived from surface normals (red: stroke; green: cutting planes)*

4.1. Iterative local optimization

To obtain better cutting plane orientations, we introduce an iterative optimization. All planes are initialized as in the previous section and then rotated by incrementally adjusting α and β so that they better cut across the mesh and are more consistent with each other. Since the desired region of interest is unknown, it is difficult to assess how well the arrangement of cutting planes covers its shape as a whole. The optimization criteria used are therefore all local, based on the quality of the cut produced by each plane and its relation to the immediate neighbors. In every step, the angles are adjusted by adding weighted sums of these criteria.

Because cutting planes are constructed for densely spaced anchor points, the skeleton is likely to undergo only gradual changes in direction between them. As a result, consecutive cutting planes should roughly be parallel. The differences in orientation between a cutting plane and its two neighbors are expressed in terms of the two rotation angles, yielding a total of four criteria. These are assigned negative weights so that the iterative optimization minimizes angle differences.

The second optimization target is to keep the planes' cuts through the mesh from intersecting each other. A cut has the form of a planar polygon and can efficiently be computed if the mesh is stored in a half-edge data structure. The degree to which two cuts intersect, however, is difficult to measure. We therefore approximate it by the fraction of a cut's total vertices that lie on the far side of the neighboring cutting plane. This value is an optimization criterion for both α and β, its weight set so that the planes rotate toward parallelism.

Two final criteria are calculated for each plane on its own. They assess how well the plane's orientation agrees not with its neighbors, but with the shape of the mesh. Based on the surface normals along the cut, angles are computed by which the plane should be rotated to cut across the mesh in a more "natural" way. As observed in [LWTH01], simply averaging the normals may indicate a good orientation in some cases, but leads to meaningless results for cylindrical objects. The more robust method used instead is based on the observation that given any surface normal, "natural" cutting plane orientations are those that include this normal. With the cut expressed as a polygon, the average surface normal is calculated for each of its edges. The adjustments to α and β are then recorded that would align the plane with this normal. Optimization criteria are obtained by adding the influences of all edges, weighted by their lengths.

4.2. Results

The approach works for some simple meshes, but fails in more complex cases. We attribute its failings to four main shortcomings. First, inherent to all iterative optimizations, is a strong dependence of the final result on a suitable initial configuration. Our technique of using surface normals to initialize cutting plane orientations leads to good results on, for example, a cylinder, but may already fail for a cone:

Following the normals on the lateral surface of a cone leads not only across, but also away from its apex. For a stroke parallel to the main axis, this leads to the initial planes being directed downward (figure 3, left). Whenever a plane cuts across the lateral surface only, iterative optimization is able to tilt it upward and produce the desired cut across the cone, parallel to its base. If, however, the plane traverses both lateral surface and base, the surface normals direct it in the opposite direction, leading to an inconsistent cutting plane arrangement (figure 3, right). Both rotations are correct in that they locally optimize cutting plane orientations. Nevertheless, the two resulting clusters of planes do not describe an intuitively correct region of interest. A line connecting the cuts' centers is also not likely to be the desired skeleton.

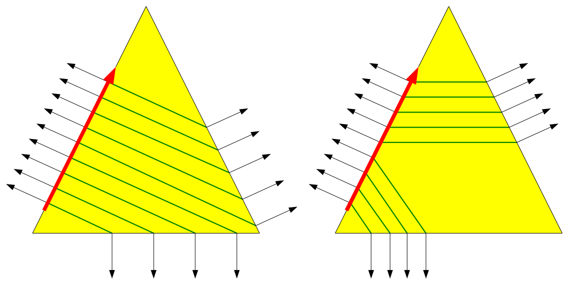
Figure 3: *Side view of cutting plane orientations before and after optimization (red: stroke; green: cutting planes)*

A second issue is that without additional constraints, cutting plane orientations and resulting skeleton may drift arbitrarily far from the user's stroke. The problem is most apparent for strokes orthogonal to the main axis, such as a line drawn along the perimeter of a cylinder. With two of the optimization criteria aiming to rotate the cutting planes toward more "natural" orientations, all planes could end up turning so as to cut across the cylinder, thus becoming parallel to the sketch, coinciding with each other and producing a singular skeleton consisting of a single point in space. To prevent this behavior, the rotation angles must be clamped to a range of values. This immediately leads to the question of how to determine suitable limits. Also, if the cutting planes rotate so far as to hit the minimal or maximal permissible angles, they are prevented from reaching “natural” directions, very likely leading to poor quality region of interest and skeleton.

The third difficulty lies in the large number of constants that need to be tuned. A weight must be set for each criterion, the number of iterations decided and limits for the rotation angles chosen. We have found that parameter values which work for one stroke may produce unacceptable plane arrangements for others, even on the same mesh. No single set of parameters has emerged as generally applicable.

Finally, no systematic handling of ambiguous input is provided. Given just one stroke on the surface, it is impossible to guarantee that even the most elaborate algorithm will be able to capture the user’s intention. If the region of interest and its skeleton do not agree with what the user wanted, it should be possible to request an alternative interpretation or add more input to disambiguate the situation. No such control is available in this approach.

4.3. Global adjustment

Our analysis shows local optimization to be insufficient in many cases. We address the issue by introding a global component. As noted earlier, the cutting planes’ global agreement with the region of interest cannot be assessed during optimization because the region is not yet known. It is possible, however, to more globally view the cutting plane arrangement by itself and correct any internal inconsistencies. To limit computational cost, the iterative local optimization algorithm is retained and a global adjustment only periodically performed every fixed number of steps.

We have implemented the global adjustment in the form of sweeps. After a certain number of local steps, plane orientations are propagated, or swept, along the stroke. In the most basic case, the orientation of every plane is set to that of its immediate predecessor. More aggressive sweeps are also possible, propagating the orientation of a plane to several successors. Despite not actively assessing their arrangement, this simple approach directs the planes toward a more globally consistent state. The underlying concept is that local optimization rotates every plane to the nearest local optimum. By periodically reinitializing the planes using the current orientation of a predecessor, each plane rotates toward the locally optimal orientation closest to that of its predecessor. Successive global sweeps then propagate the new orientations further down the stroke.

Computational cost may be further reduced by returning to a completely local optimization and only using global information in the construction of the initial cutting planes. To detect trends and dominant directions, surface normals are clustered along the stroke. Smoothing the normals within each cluster better aligns the initial planes with each other and eliminates singular planes whose directions strongly disagree with those of their neighbors. Such singularities may for example arise when the stroke genrally follows a smooth surface, but passes over a small local bump. Grouping of similar normals allows singularities to be distinguished from sudden changes in normal direction due to actual bends in the model. Contrary to a naïve smoothing of all normals, clustering thus follows the global shape of the model and preserves significant sharp changes.

4.4. Results

Addition of a global component to the iterative optimization process is beneficial for some meshes, while introducting new issues for others. As a positive example, global sweeps help to produce a consistent cutting plane arrangement for a cone. Using only local criteria, some planes may rotate toward the apex and others away from it. With periodic sweeps reinitializing orientations, the planes are adjusted to successively align with their predecessors.

This success, however, also highlights a problem. When propagating plane orientations, the resulting arrangement is dominated by the locally best orientation found for the first plane of the stroke. Very different regions of interest may thus be obtained depending on the stroke direction. In the example of a cone, the planes become parallel to the cone’s base when the stroke is drawn away from its apex. If the stroke is sketched the other way, parallel and consistent planes are also obtained, but cut across both lateral surface and base. In this case, such behavior may be desired, allowing the user to disambiguate the input by choosing a stroke direction. For more complex models, however, sweeps may rotate planes toward a uniform bearing from which local optimization is unable to return them to their correct orientations. The risk increases as the sweeps are made more aggressive, reinitializing a larger number of planes to the same orientation and doing so after a smaller number of steps.

Clustering of surface normals during initialization introduces a different type of problem. It requires precise criteria to be chosen for which normals should be grouped together. This leads to a need for more constants and tuning, aggravating the similar issue encountered in the original local optimization technique. It is again unlikely that a set of parameter values can be found that works for all meshes.

In summary, global adjustments are able to improve some aspects of the local optimization algorithm, but introduce new problems – the risk of an overzealous propagation of cutting plane directions and the need for additional tuning. Other important issues are not addressed at all, among them the missing handling of ambiguous input and the large number of constants already required by the initial algorithm.

4.5. Dynamic programming

A fundamentally different optimization technique can be developed using dynamic programming. It does not rely on iterative improvement, avoiding the issues associated with it. Instead of progressively refining the cutting plane orientation at each anchor point, a number of candidate planes are constructed and the best one chosen among them.

This method requires a compromise to be made between computational expense and the probability that a satisfactory plane arrangement can be found. The more candidates are constructed, the more exhaustively the space of possible cutting plane orientations can be traversed, but the longer and more memory-consuming the calculations become. To achieve an interactive workflow, only one rotation angle is optimized. The value of α is fixed so that each cutting plane be perpendicular to the stroke in the tangential plane of its anchor point. This compromise is based on the observation that if cutting plane and skeleton are orthogonal, so should be their projections into the tangential plane.

To optimize the other angle, β, a set of candidate values must be chosen. This task is faced with difficulties similar to those of finding a suitable initial cutting plane for iterative optimization. Given a mesh and a point on its surface, it is impossible to precisely determine which plane orientations are worth investigating and which are not. We therefore do not attempt to derive candidates angles from the shape of the mesh but rather uniformly sample a likely range of values, from -80° to 80° in two degree increments. The resulting 81 planes may directly be used as candidates. Alternatively, a pruning pass can eliminate those least likely to produce suitable cuts. One pruning criterion are the surface normals along the cut. As explained in section 4.1, these can be combined into a signed metric that indicates in which direction the plane should be rotated to result in a "natural" cut across the mesh. Cutting planes close to the zeros of this metric should be retained as they have the most "natural" orientations. Other criteria are minima of cut circumference or area.

When the final set of candidates has been established, a cost metric is calculated for each pair of candidate planes anchored at neighboring points. It expresses the probability that these two planes are consistent and describe the same region of interest. Planes roughly parallel to each other producing similar cuts should receive a high rating, while planes whose cuts intersect or that point in very different directions should be rated lower. The metric is a weighted sum of several criteria: the angle difference between the planes, relative difference of cut circumference, of cut area and of form factor. The form factor is a shape descriptor that captures the essence of the cut shape in a single number. As defined in [Rus98], it is a value proportional to $area/\sqrt{circumference}$.

The results of these preparatory steps are combined into a directed acyclic graph. For each anchor point, the candidate cutting planes are represented by a group of vertices. Those located in neighboring groups are linked by edges, their lengths given by the cost metric above. The most likely combination of planes then corresponds to the least cost path through the graph beginning at a candidate plane for the first anchor point and ending at one for the last. Since the graph is directed and acyclic, the path can efficiently be found using dynamic programming. Implementation is simplified by the fact that vertices are grouped and each edge points from one group to the next. The total computational cost is $O(k^2 n)$, with k the number of candidate planes for each of the n anchor points extracted from the stroke.

4.6. Results

Dynamic programming often produces better results than the two iterative approaches. Its capabilities are demonstrated by the three-pronged star in figure 4. When a stroke is drawn along the side of one of the prongs, dynamic programming correctly constructs a series of parallel planes cutting across this prong only. If the stroke is extended to span the sides of two prongs, two clusters of planes are returned, covering both. The skeleton then also traverses both prongs, as is likely to have been intended by the user.

An important advantage of dynamic programming is its inherent ability to gracefully deal with ambiguities. Initially, the cutting plane arrangement considered most likely is returned. If this is not what the user intended, alternative interpretations can easily be found by following progressively more expensive paths through the graph. As the cost gradually increases, interpretations are encountered in decreasing order of likelyhood. The user interface for choosing between these alternate solutions could be a simple button for requesting the next one, a slider for browsing between them or a screen showing multiple interpretations, one of which is chosen by clicking on it.

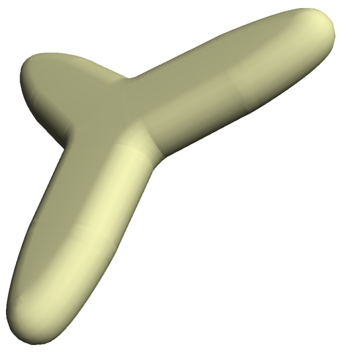

Figure 4: *Three-pronged star*

While simple and elegant in theory, the handling of ambiguous input is more problematic in practice. Most paths through the graph with low cost yield subtle variations of the most likely interpretation. Only after traversing many such similar solutions is one found that provides a genuinely different interpretation of the input. To aid the user in disambiguating a stroke, solutions should be clustered, presenting only one representative for each group of similar cutting plane arrangements. This, unfortunately, introduces the need to develop and tune a suitable post-processing algorithm.

The optimization technique's biggest drawback is that it itself also requires extensive tuning. What arrangement of cutting planes is deemed most likely heavily depends on the weights assigned to the individual criteria that make up the cost metric. As with the iterative approaches, no single set of parameter values has emerged as universally applicable.

A final issue is that the two rotation angles cannot be optimized simultaneously at interactive speeds. The constant value chosen for α is based on a reasonable assumption and often close to optimal. Small adjustments, however, could still greatly improve the cutting plane arrangement in many cases. Since simulatenous optimization is too expensive, an alternative may be to perform multiple passes, alternating between adjusting α and β.

5. Two strokes

A single stroke is the most basic and simple form of input available in a sketch-based system. This does not necessarily mean, however, that it also is the most intuitive for all applications. In [KG05], the user's input directly serves as the control structure. A stroke spanning the entire region of interest is therefore required. Our work presumes a less direct relationship between input and skeleton. Projection from the mesh surface inward produces a skeleton guided both by the user's input and the shape of the model.

To capture the user's intent, it is most crucial that the correct first and last cutting planes be found. These delimit the region of interest, defining the part of the mesh to be covered by the other cutting planes and skeleton. When using a single stroke as input, the user is forced to specify the two cutting planes indirectly: starting and ending point of the stroke serve as anchors, while stroke directions and surface normals provide initial guesses for the orientations. The optimization algorithm must then attempt to construct the desired planes based on this incomplete information. It would seem more intuitive to give the user direct control over the two most important cutting planes, reducing ambiguity and the potential for incorrect interpretations. This is possible by using two strokes as input, one specifying the beginning and the other the end of the region of interest.

The intuitiveness of using a single stroke is questionable not only because it lacks direct control over the first and last cutting planes. A single stroke is interpreted as a projection of the desired skeleton onto the mesh surface. It is debatable whether users can intuitively think in terms of such projections. This is especially true as projection depth and direction are computed only after the stroke has been finalized, making it impossible to visualize the resulting skeleton while the user is sketching. Another issue is that the eventual projection carries over any imperfections in the stroke. Minor jitter due to the user's hand shaking can be eliminated by filtering. More serious defects, such as loops in the stroke, are difficult to detect and impossible to fix, leading to invalid regions of interest and skeletons.

Input in the form of two strokes allows the user to more precisely specify the region of interest, but provides no information about the desired skeleton. To construct the skeleton, we first synthesize its projection onto the mesh surface, simulating the single stroke used before. This is then projected inward with the help of intermediary cutting planes. Their orientations must again be found by optimization, which is now easier due to the additional information available about the region of interest they should cover.

For the skeleton's representation on the mesh surface, we use the shortest path connecting the first and last cutting planes, known as a geodesic. Geodesics can be approximated by Dijkstra's algorithm in $O(n \log n)$ time with n the number of mesh vertices. Because a direct application of Dijkstra's algorithm to the mesh produces very coarse approximations, we use the more sophisticated algorithm of [KS00] instead. As demonstrated by the authors, this technique produces high quality approximations at a fraction of the cost of an exact calculation. The geodesic is a good basis for projection inside the model as it is guaranteed to both traverse the entire region of interest and be free of defects such as loops.

5.1. Sketching cutting planes

The question remaining is how each cutting plane is to be derived from the corresponding stroke. An obvious method is to locate the plane providing the best fit. As the user is sketching, the stroke is projected onto the underlying mesh, resulting in a three-dimensional line lying on the surface. This is sampled at short intervals, producing a point cloud. The cutting plane is then the regression plane that best fits these points. We use principal component analysis for regression. The cutting plane passes through the centroid of the point cloud and its orientation is given by the first two principal components of the cloud. A different regression technique, such as least-squares, could also be used.

5.2. Results

This intuitively appealing approach does not work well in practice. As observed in [LLS*04], a cut across a mesh can take on very different shapes on the front and back surfaces. Because the user is only sketching the front part, the stroke's projection onto the mesh may not be representative of the entire cut. A second problem is that even when the user is certain of the desired cutting plane, precisely drawing a stroke that represents it is very difficult. For example, to sketch a plane cutting through a cylinder, the user must manually follow the surface curvature. The task becomes almost impossible on a cone, where the resulting regression plane is very likely to be tilted up- or downward instead of cutting through the cone parallel to its base. Further difficulty is added if a perspective view is used.

5.3. Sketching rotation axes

The observations made in the previous section suggest a less literal interpretation of the stroke. Instead of two principal components, only the first one is used, corresponding to the most pronounced direction in the stroke. Coupled with one of the points on the mesh surface the stroke passes through, this produces a line that captures the essence of the user's input. The cutting plane should contain this line, making it an axis the rotation about which is the only unknown.

To fully specify the cutting plane, its rotation must be determined using one of the optimization techniques. A plausible initial orientation can be derived from the second principal component. In this interpretation, the first principal component is deemed trustworthy while the second is taken only as an initial suggestion and then optimized. If minor adjustments to first component's direction are also to be allowed, a second rotation axis orthogonal to the first can be used.

5.4. Results

This approach avoids the problems encountered when computing the cutting plane directly from a stroke, but unfortunately introduces new challenges. For one, the additional optimization step brings with it all the weaknesses of whatever optimization technique is chosen. Second, any adjustment of the cutting plane direction bears the risk of producing a plane that does not agree with the user's intentions. While the user has more direct control over the cutting plane than in the single-stroke methods, its final orientation is still influenced by factors other than user input.

6. Conclusions and future work

Ad-hoc approaches have made it possible to deform meshes without the tedious preprocessing during which a control structure is normally built and attached. They have introduced a new restriction, however: The control structures provided are generally two-dimensional. We have proposed in this paper to overcome the limitation by computing three-dimensional region of interest and skeleton on the fly. Building on the strengths of the ad-hoc approach, we do not wish to burden the user with providing additional input to describe the more complex three-dimensional structure. Instead, we perform a more intelligent processing of the input strokes to find an interpretation that agrees with the user's intentions.

Our initial approach of using a single stroke is motivated by the fact that this is the most simple form of input possible. Unfortunately, it also carries the least amount of information, requiring substantial interpretation effort. If the input is misinterpreted, none or the wrong region of the model may be selected. The iterative algorithms are especially sensitive to initial cutting plane orientations. If the cutting planes constructed from the only given at the beginning, the stroke and the surface in its immediate vicinity, do not point roughly in the right directions, iterative optimization is unable to locate the desired region of interest. Dynamic programming is more robust as it considers a wider range of cutting plane orientations, irrespective of the local surface normals. Despite this, significant problems remain.

The computation works best on long, straight cylindrical objects where the skeleton is the main axis and the region of interest the lateral surface around it. This class includes, for example, the limbs of many animals (figures 5, 6). If multiple interpretations are plausible, some of the cutting planes may follow one, some the other. Global adjustments can restore consistency, but do not allow the user to intervene when the wrong interpretation is chosen. Results are undefined if the user's stroke cannot easily be interpreted. Such strokes include those with defects, for example loops or sudden reversals in stroke direction, but also strokes drawn in seemingly nonsensical directions on the mesh surface. As no cutting plane arrangement can be consistent with both the stroke and the shape of the mesh, the result is a region of interest violating at least one of these two factors. Which of them prevails depends only on the optimization weights chosen.

Use of two strokes promises to address these deficiencies. By giving the user direct control over the ends of the region of interest, accidental selection of the wrong mesh region can be avoided. Construction of cutting planes by regression eliminates the sensitivity to loops and other corruptions in the strokes. The only inconsistency possible is the user specifying overlapping ends for the region of interest, which can easily be detected. Cutting plane orientations within the region must still be found by optimization, but that is significantly easier if the region's extents are known.

One challenge remaining is finding the optimal interpretation method for the two strokes. As shown in section 5.1, cutting directly along a stroke may not lead to correct results. The less literal interpretation of the user's stroke as a rotation axis in section 5.3 allows more corrections to be made to the cutting plane orientations, but at the risk that the result may deviate from the user's intentions. We want to more precisely investigate these two and other, alternative methods. For example, the mesh could automatically begin to rotate as the pen approaches its silhouette, allowing the user to sketch a cut around the entire model in a single stroke. After implementing these methods, we want to run a user study, soliciting feedback and choosing the one most suitable.

The second issue we need to address is choosing an optimization algorithm for the intermediary plane orientations. Dynamic programming has shown the most promise and is the most likely candidate. However, the problem of its reliance on constants and tuning remains to be solved. We currently use a weighted sum of four criteria. A precise analysis of each criterion's influence should help to decide which of these can be dropped, simplifying the algorithm, reducing the number of constants and making it more likely that a widely applicable set of parameter values can be found. If no single set of parameters proves satisfactory, we will provide the user with a choice of multiple interpretations. This can be tightly integrated with dynamic programming's inherent ability to produce multiple interpretations for a single stroke, allowing the user to obtain the desired cutting plane arrangement even in difficult, ambiguous cases.

After these issues have been addressed, we want to extend our work to provide actual mesh deformation. Similar to [KG05], two strokes select part of a mesh and provide it with a reference skeleton. A single stroke is then sufficient to serve as the target shape toward which the skeleton is deformed. Contrary to and improving on Kho and Garland, the

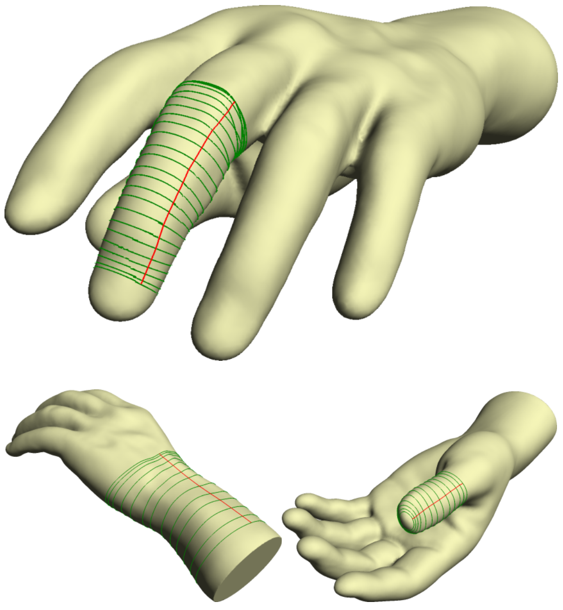
Figure 5: *Results of local iterative optimization for a human hand model (red: stroke; green: generated cuts)*

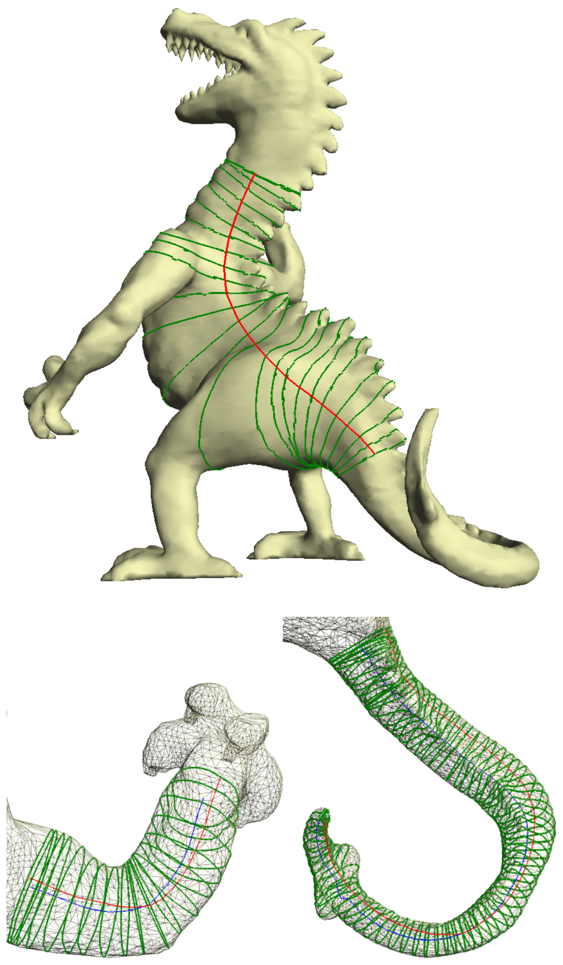
Figure 6: *Results of local iterative optimization for a dinosaur (red: stroke; green: generated cuts; blue: skeleton)*

view can then be rotated and a new target sketched without constructing a new skeleton and region of interest. This will allow the user to specify complex deformations without risking that errors and self-intersections propagate. Using the region of interest's spatial representation provided by the cutting planes, self-intersections can additionally be detected and optionally automatically resolved by moving the offending parts of the mesh away from each other.

7. Acknowledgments

This work is financed by the "Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan". The example models were found on the Internet with their original sources unfortunately unknown.

References

[Blu67] BLUM H.: A transformation for extracting new descriptions of shape. In *MPSVF 1967* (1967), pp. 362–380.

[BST05] BOUIX S., SIDDIQI K., TANNENBAUM A.: Flux driven automatic centerline extraction. *Medical Image Analysis 9*, 3 (2005), 209–221.

[Buj06] BUJANS R.: *A Thesis on Sketch-Based Techniques for Mesh Deformation and Editing*. Master's thesis, Washington University in St. Louis, St. Louis, MO, USA, 2006.

[CGC*02] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Interactive skeleton-driven dynamic deformations. In *SIGGRAPH 2002* (2002), pp. 586–593.

[CJ06] CHANG E., JENKINS O.: Sketching articulation and pose for facial animation. In *SCA 2006* (2006), pp. 271–280.

[HF05] HASSOUNA M., FARAG A.: Robust centerline extraction framework using level sets. In *CVPR 2005* (2005), pp. 458–465.

[HQ03] HUA J., QIN H.: Free-form deformations via sketching and manipulating scalar fields. In *SPM 2003* (2003), pp. 328–333.

[KG05] KHO Y., GARLAND M.: Sketching mesh deformations. In *I3D 2005* (2005), pp. 147–154.

[KS00] KANAI T., SUZUKI H.: Approximate shortest path on a polyhedral surface based on selective refinement of the discrete graph and its applications. In *GMP 2000* (2000), pp. 241–250.

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. In *SIGGRAPH 2003* (2003), pp. 954–961.

[LCJ93] LAZARUS F., COQUILLART S., JANCÈNE P.: *Interactive Axial Deformations*. Tech. rep., INRIA, 1993.

[LH00] LAW T., HENG P.: Automatic centerline extraction for 3D virtual bronchoscopy. In *MICCAI 2000* (2000), pp. 786–795.

[LLS*04] LEE Y., LEE S., SHAMIR A., COHEN-OR D., SEIDEL H.: Intelligent mesh scissoring using 3D snakes. pp. 279–287.

[LPRS05] LLAMAS I., POWELL A., ROSSIGNAC J., SHAW C.: Bender: A virtual ribbon for deforming 3D shapes in biomedical and styling applications. In *SPM 2005* (2005), pp. 89–99.

[LV94] LAZARUS F., VERROUST A.: *Feature-based shape transformation for polyhedral objects*. Tech. rep., INRIA, 1994.

[LV99] LAZARUS F., VERROUST A.: Level set diaglrams of polyhedral objects. In *SPM 1999* (1999), pp. 130–140.

[LWTH01] LI X., WOON T., TAN T., HUANG Z.: Decomposing polygon meshes for interactive applications. In *SI3D 2001* (2001), pp. 35–42.

[MP02] MORTARA M., PATANÈ G.: Shape-covering for skeleton extraction. *Shape Modeling 8*, 2 (2002), 139–158.

[MPS*04] MORTARA M., PATANE G., SPAGNUOLO M., FALCIDIENO B., ROSSIGNAC J.: Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *SM 2004* (2004), pp. 339–344.

[NSACO05] NEALEN A., SORKINE O., ALEXA M., COHEN-OR D.: A sketch-based interface for detail-preserving mesh editing. In *SIGGRAPH 2005* (2005), pp. 1142–1147.

[OIN*06] ODA T., ITOH Y., NAKAI W., NOMURA K., KITAMURA Y., KISHINO F.: Interactive skeleton extraction using geodesic distance. In *ICAT 2006* (2006), pp. 275–281.

[OSSJ05] OLSEN L., SAMAVATI F., SOUSA M., JORGE J.: Sketch-based mesh augmentation. In *SBIM 2005* (2005), pp. 43–52.

[PWMZ95] PISUPATI C., WOLFF L., MITZNER W., ZERHOUNI E.: A central axis algorithm for 3D bronchial tree structures. In *ISCV 1995* (1995), pp. 259–264.

[Rus98] RUSS J.: *The Image Processing Handbook*, 3rd ed. CRC Press, 1998.

[SF98] SINGH K., FIUME E.: Wires: A geometric deformation technique. In *SIGGRAPH 1998* (1998), pp. 405–414.

[SP86] SEDERBERG T., PARRY S.: Free-form deformation of solid geometric models. In *SIGGRAPH 1986* (1986), pp. 151–160.

[TT98] TEICHMANN M., TELLER S.: Assisted articulation of closed polygonal models. In *CAS 1998* (1998), pp. 87–102.

[TVD07] TIERNY J., VANDEBORRE J., DAOUDI M.: 3d mesh skeleton extraction using topological and geometrical analyses. In *SGP 2007* (2007), pp. 153–162.

[YBS03] YOSHIZAWA S., BELYAEV A., SEIDEL H.: Free-form skeleton-driven mesh deformations. In *SPM 2003* (2003), pp. 247–253.

[ZHS*05] ZHOU K., HUANG J., SNYDER J., LIU X., BAO H., GUO B., SHUM H.: Large mesh deformation using the volumetric graph laplacian. In *SIGGRAPH 2005* (2005), pp. 496–503.

[ZMY06] ZHAO M., MA L., YONG Z.: Mesh cutout. *IEICE Transactions on Information and Systems E89-D*, 7 (2006), 2207–2213.