Scaffolding a Skeleton

Athina Panotopoulou, Kathrin Welker, Elissa Ross, Evelyne Hubert, Géraldine Morin

Scaffolding a Skeleton

Athina Panotopoulou, Kathrin Welker, Elissa Ross, Evelyne Hubert and Géraldine Morin

Abstract The goal of this paper is to construct a quadrilateral mesh around a one-dimensional skeleton that is as coarse as possible, the "scaffold". A skeleton allows one to quickly describe a shape, in particular a complex shape of high genus. The constructed scaffold is then a potential support for the surface representation: it provides a topology for the mesh, a domain for parametric representation (a quad mesh is ideal for tensor product splines) or, together with the skeleton, a grid support on which to project an implicit surface that is naturally defined by the skeleton through convolution. We provide a constructive algorithm to derive a quad-mesh scaffold with topologically regular cross-sections (which are also quads), and no T-junctions. We show that this construction is optimal in the sense that no coarser quad mesh with topologically regular cross-sections may be constructed. Finally, we apply an existing rotation minimization algorithm along the skeleton branches, which produces a mesh with a natural edge flow along the shape.

Athina Panotopoulou
Dartmouth College, e-mail: athina@cs.dartmouth.edu

Kathrin Welker
Trier University, e-mail: welker@uni-trier.de

Elissa Ross
MESH Consultants Inc., Fields Institute for Research in the Mathematical Sciences e-mail: elissa.ross@meshconsultants.ca

Evelyne Hubert
INRIA Méditerranée, e-mail: evelyne.hubert@inria.fr

Géraldine Morin
IRIT - University of Toulouse, e-mail: morin@n7.fr

1 Introduction

A skeleton, like the skeleton of vertebrates, defines synthetically a shape and makes intelligible its general organization while helping to visualize its possible deformations. Mathematically, a skeleton is a union of lower dimensional manifolds that is homotopy equivalent to the shape. As such it characterizes the topology of the shape. Skeletons, and more particularly medial representations, are computed for analyzing and identifying shapes, with a purpose of automatic recognition of objects, or of a human eye diagnosis. Such models lend themselves to identification and matching, and are therefore used for retrieval [6, 16].

Computer graphics also use skeletons as a starting point for shape design [22]. Skeletal animation (rigging) is an established technique to animate moving characters. Ultimately, shapes will be represented by their boundary surfaces, as either an implicit or parametric surface, or as a fine mesh. Yet the overall design of the shape and its topology is more easily entered as a skeleton. Our approach thus allows to create a 2-manifold mesh of surfaces of high genus starting from a schematic description of the shape (the skeleton) rather than plugging handles on an existing mesh as in [14] for example.

The aim of our project is to construct a quadrilateral mesh around a skeleton that is as coarse as possible, like a scaffold on which to build the surface. Such a mesh provides the first step for defining a joint surface representation for a skeleton based shape; the scaffold quads can be refined by subdivision to fit a regular quad mesh to a surface shape, as in [18]. Our construction is optimal in the sense that it only consists of quads, in contrast to other methods of constructing meshes from skeletons [2, 10, 18]. In addition, our method is minimal in the sense that it is not possible to construct a coarser quad mesh of the same skeleton with uniform sections. By uniform sections, we mean that every section is of the same type of polygon, for example triangle, quad, hexagon.

The paper is organized as follows. In Section 2, we give an overview of the related work. In Section 3 we define the skeleton we shall work with and outline the construction of the scaffold. The two main parts of the algorithm are then described in the next two sections. Finally, in Section 6 we present a selection of examples of skeleton for which we generate a scaffold with the described method.

2 Related Work

3D Skeletons. The skeleton (medial axis and radius function on the axis) can be seen as a dual to a surface representation which captures all shape aspects that an explicit or implicit representation captures [17]. Here, we do not consider a skeleton that characterizes the entire shape, but rather a simple piecewise linear skeleton, similar to skeleteons used for animation [13], or for convolution surfaces [4, 8, 11, 9].

Inverse skeletonization, garbing, and skeletally driven modeling. Recent research works have tackled the problem of generating an explicit mesh based on

a given skeleton. Bærentzen et al. [2] presented an algorithm to create a quad-dominant polygonal mesh. This method, similar to our work, starts by triangulating the spheres on vertices with three or more neighbors (branch nodes). Our approach uses quadrilaterals as building blocks instead of triangles, which results in a more regular mesh. Their method is intended for simpler tree structured meshes, as they do not take into account torsion of the skeleton (i.e. how far the skeleton is from planar) after the rotation step. Furthermore, they also do not bound the number of vertices of a face and may have sections with a varying number of vertices. Usai et. al. [18], use only quadrilaterals for the branching nodes and considers torsion, but their construction leads to meshes with irregular sections and to ambiguous final representations in an effort to solve the algorithm's inherent T-junctions.

Bærentzen et. al. [3] proposed the use of a Polar Annular Mesh as a light-weight representation of any polygonal mesh that is accompanied by a skeletal representation. Because their technique may produce meshes with high valency polar triangle fans, it may be more memory intensive than ours and it lacks the explicit encoding of radius on the skeleton. They do not provide a theoretical justification of the completeness for the meshing of branch points. Therefore our proposed construction could complement their definition of the Skeleton-Mesh co-representation, as our algorithm has a complexity linear in terms of the number of vertices in the graph. Ji et al. [10] mesh the branches before the branching nodes, to produce a quad dominant mesh. Their algorithm can create triangles and irregular vertices on the joints. Also, in cases that they produce meshes of bad edge flow (how well the lines of the edges appear to be aligned with the shape of the skeleton), they need to solve a quadratic optimization problem. In other words, the method used in [10] for selecting the local frames may be heuristic at every node. In contrast, we use a heuristic approach for the frames at the branch nodes, but the frames on the remaining nodes are defined using rotation minimization. This allows for the straightforward realization of examples such as the spiral in Figure 6. The similar problem addressed in [15] is different in that it imposes at the outset a given polygonal section to the branches of the skeleton.

3 Construction outline

In this section we first define a skeleton as an embedding in $\mathbb{R}^3$ of a simple graph and then split our construction of a scaffold into two major steps. These two steps are detailed in the next two sections.

A *skeleton* is defined by its topology and its geometry. The topology is given by a simple, unweighted, undirected and connected graph $G = (V,E)$. Here V denotes the set of vertices and E denotes the set of edges. Each vertex v of V is associated with a point p_v in $\mathbb{R}^3$, and a radius $r_v > 0$ which defines a sphere centered at p_v (Figure 1). In other words, a *skeleton* is defined as the triple (G, p, r), where

- $G = (V,E)$ denotes the graph defined above

- $p: V \to \mathbb{R}^3$,
  $v \mapsto p_v$ is the graph embedding, i.e. an assignment of unique positions from $\mathbb{R}^3$ to each vertex of the graph, and
- $r: V \to \mathbb{R}^+$,
  $v \mapsto r_v$ with r_v denoting the radius associated to the vertex $v \in V$.

Consequently, any edge of the graph G is embedded as a line segment of $\mathbb{R}^3$. Edges are assumed to intersect only at endpoints. We furthermore assume that, outside the spheres centered at the vertices, the line segments are far enough from each other for the scaffold not to intersect. In the present work, we also assume that the spheres defined on the embedded graph do not intersect. There does not seem to be a large obstacle to treating this case, as shown in some of the examples (see Figure 8). Yet understanding how the intersections of the spheres could cause failures in the resulting mesh is a topic of future work.

We distinguish between the so-called joints, branch points and extremities: The image in $\mathbb{R}^3$ of a vertex of valency 2 (respectively 1) is called a *joint* (respectively an *extremity*). A *branch point* is the image p_v of a vertex v of valency 3 or more. Together, we call the joints extremities and branch points *skeleton nodes* (or simply *nodes*, where it is clear from context).

A *branch* is a subgraph connecting either two branch points or a branch point to an extremity, with no intermediate branch points.

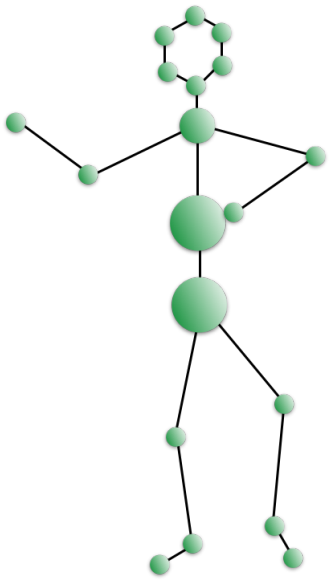


Fig. 1 A skeleton: an embedded graph with a sphere at each vertex.

For a given skeleton (V, E, p, r), a *scaffold* is a closed orientable polyhedral surface enclosing the union of line segments defined by the skeleton, while being ho-

motopically equivalent to it. The scaffold we shall construct is a quad mesh, the faces of which we call *scaffold quads*, (white quads in Figure 2). Each scaffold quad is associated to a single skeleton edge and to a skeleton edge is associated exactly 4 scaffold quads (a *sleeve*). Our construction of a scaffold for a skeleton consists of two main steps:

Partition: For all branch points p_v, partition the sphere S_v centered at p_v and with radius r_v, according to the issuing branches. In this step, the spheres are partitioned into quadrilateral faces, called *spherical quads* (dark green quad in Figure 2). Each spherical quad face is intersected by a single line segment joining the branch point to a neighbor vertex.

Sleeve: With a depth first search (DFS) order, starting from any branch point or extremity, *sleeve* all the branches. In this step, we compute a propagated spherical quad, (light green quad in Figure 2), for every joint (*i.e.* skeleton node of degree-2) following the DFS order. We use a heuristic alignment for the joints or extremities connected to a branch point, and the rotation minimization algorithm for all remaining skeleton nodes. We use these quadrilaterals surrounding the nodes of the skeleton to mesh every edge of the skeleton using four quad faces. Finally, we close the mesh at extremeties.

4 Partitioning the sphere at a branching point

In this section we detail the first step of the construction of a scaffold, the partition of the sphere around a branch point of the skeleton into spherical polygons that separate the adjacent edges of the skeleton. We look for a regular such partition, *i.e.* one where all the spherical polygons have the same number of vertices. We show that the use of quadrangles is the simplest possibility that works independently of the number of adjacent edges to the branch point. Note that we address here only the partition of the sphere around the branch points as the joints and extremities are dealt with when we treat the sleeves.

Let us consider one branch point, and we denote N its valency ($N \geq 3$ by definition). We first compute N points on the sphere, corresponding to the N adjacent edges of the skeleton. We want to partition the sphere into N regular spherical polygons such that each spherical polygon contains exactly one of the N points. Here, regular is meant in a topological sense, i.e., each spherical polygon has an equal number of vertices. Each spherical polygon, also referred to as a face, will be the support of an outgoing sleeve (paart of the scaffold) enclosing the skeleton branch starting with the adjacent edge stemming out this face. The vertices of the spherical polygons will be the only vertices of the scaffold that belong to this sphere. In order to *sleeve* a branch connecting two branch points, without introducing T-junctions, it is necessary that the faces around each branching point all have the same number of vertices.

We seek the simplest scaffold as possible. Hence we first address the question of how the sphere can be partitioned such that the number of vertices of the faces

is minimal. We first show that we cannot always obtain an appropriate partition of the sphere into spherical triangles. We then provide a construction to partition the sphere into spherical quadrangles. This proves that four is the minimal number of vertices that our spherical polygons should have to partition the sphere at branch points.

4.1 Minimum number of vertices of faces on the sphere

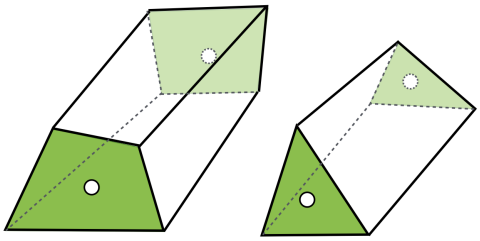


Fig. 2 The cross section of a branch sleeve. In this work we describe a construction method for quad meshes with *quadrilateral* cross sections (left).

Let us consider a sphere centred at a branch point of the skeleton under consideration. Assuming that we have N points on the sphere, corresponding to the intersection of the sphere with the N outgoing branches of the skeleton, we wish to separate the sphere into regular faces, such that each region contains exactly one point. As we seek the simplest scaffold possible, our goal is to minimize the number of vertices of the faces, i.e., we should try to have triangles. However, we prove that partitioning the sphere into N triangles is not possible if N is odd. Essential to this proof is the Euler characteristic χ which is classically defined for polyhedral surfaces by

$$\chi = v - e + f, \tag{1}$$

where v, e and f are the numbers of vertices, edges and faces in the polyhedron under consideration. For more information about the Euler characteristic we refer to the literature, e.g., [1, 7, 12]. In the case of a closed orientable surface, it can be calculated from its genus g by $\chi = 2(1-g)$. The genus of a surface is equal to the number of holes of the surface under consideration. For the sphere g it is equal to zero, i.e., we get

$$2 = v - e + f. \tag{2}$$

With the help of this formula, we can prove the following theorem.

Lemma 1. *The sphere S^2 cannot be partitioned into N triangles if N is odd.*

Proof. Consider formula (2) for this setting. We have $f = N$ because we want to have N triangles. Moreover, we assume that each face has the same number k of vertices, then $e = \frac{N \cdot k}{2}$, as each face has also k edges, but each edge is counted twice. Thus, we get

$$v = \frac{N \cdot k}{2} - N + 2 = N\left(\frac{k}{2} - 1\right) + 2. \tag{3}$$

If all faces were triangles, then $k = 3$. From (3) we get $v = \frac{N}{2} + 2$. This is not possible if N is odd. □

The lemma above states that we cannot partition the sphere into an odd number of triangles. However, it may be possible to partition any sphere into N quads, using $N + 2$ vertices. This is because, for quad faces, we have $k = 4$ in the above proof and we get $v = N + 2$ from (3). In the rest of this paper, we will construct such a partition and deduce a quad mesh scaffold of a skeleton of arbitrary topology, with quadrilateral cross-sections (inherited from the quad faces on the sphere, see Figure 2). Because the mesh is based on the partition of spheres into quads, we avoid T-junctions by construction. Next, we give an actual constructive algorithm to partition the sphere.

4.2 Algorithm to partition the sphere into quadrangles

Lemma 1 states that the sphere cannot be partitioned into N triangles if N is odd. However we may construct a partition of the sphere into quads. This subsection is devoted to the formulation and explanation of an algorithm for such a sphere partition into quads (see also the pseudocode recorded in Algorithm 1).

The *input* of the algorithm is the center of a sphere, i.e. a branch point, and its radius, together with a list of the neighbor skeleton vertices of the branch point. The *output* of the algorithm is a set of quadruples of scaffold vertices, each quadruple defining a face in the partition of the sphere.

We now outline how the sphere can be partitioned inductively into quads. For this purpose, let (G, p, r) be a skeleton, and consider a branch point C of (G, p, r). Suppose first that C has three neighbor vertices N_1, N_2, N_3. For each neighbor N_i of C we build the half-line starting at C in direction N_i. The intersection of each half-line with the sphere gives one point P_i on the sphere (Figure 3a).

These three points uniquely determine two antipodal points Q and Q' on the sphere which are equidistant to P_1, P_2 and P_3. These may be computed as the intersection of the three medial planes between P_j and P_k, where $j, k \in \{1, \ldots, 3\}$ with $j \neq k$. We connect Q and Q' with 3 geodesic curves. The geodesic between P_j and P_k being half of the circle intersection of the medial plane between P_j and P_k and the sphere, when P_j and P_k are two consecutive points on the sphere according to an angle projected on a plane orthogonal to $[QQ']$. The sphere is now partitioned like an orange with three arcs of great circles (cf. Figure 3b). Finally, we add one additional vertex on each geodesic at the midpoint of the geodesic passing between

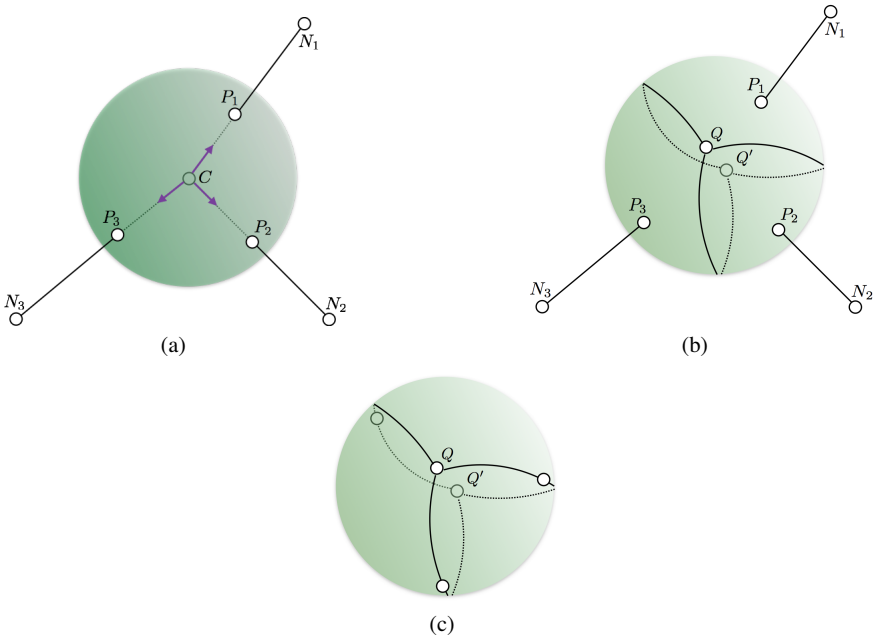


Fig. 3 A sphere with centre C, neighbors $\{N_1, N_2, N_3\}$ and resulting points of intersection on the sphere (a). The segmentation of the sphere into sections (b). The addition of points along the geodesics to create the quadrilaterals (c).

these two points. This creates a partition of the sphere into three (combinatorial) quadrilaterals (Figure 3c).

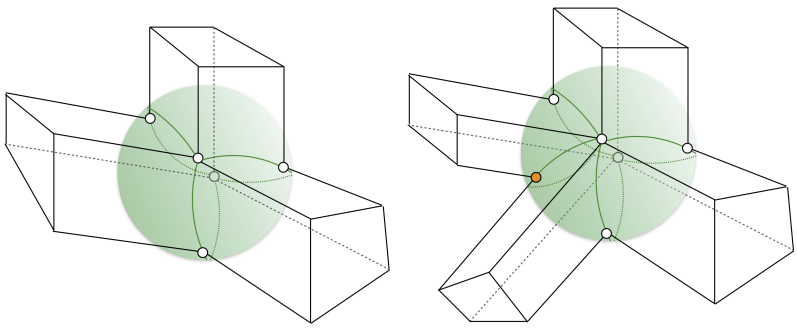


Fig. 4 Left: A sphere with three branches. Right: The same sphere with an additional branch and corresponding new quad face on the sphere.

Next assume that there are more than three points P_1, P_2, P_3 on the sphere, i.e., we have additional edges starting at the vertex C. In this case, for each additional neighbor vertex N, let P be the intersection of the edge (C,N) with the sphere. P belongs to a quad face of the sphere, say the one of P_j. We place a vertex at the midpoint M of the geodesic between P and P_j and connect M to one of the diagonals of the quad. We choose the diagonal that offers the direction most orthogonal to $[P,P_j]$.

Algorithm 1 Partitioning branch point spheres

Require: (G,p,r) is a skeleton, $C \in \mathbb{R}^3$ is a branch point, r_C is the radius of the sphere at C, $N = \{N_0,\ldots,N_m\}$ is the set of all vertex neighbors of C.

function SPHERE PARTITION(G,C)
  $S \leftarrow$ sphere of radius r_C centred at C
  $N_0, N_1, N_2 \in \mathbb{R}^3 \leftarrow$ the coordinates of the first three vertex neighbours of C in G
  for $i \leftarrow 0$ to 2 **do**
    $P_i \in \mathbb{R}^3 \leftarrow$ intersection of half-line from C to N_i with S
  end for
  $Q, Q' \in \mathbb{R}^3 \leftarrow$ the uniquely defined antipodal points on S that are equidistant to P_1, P_2, P_3
  for $i \leftarrow 0$ to 2 **do**
    $g_i = (Q,Q') \leftarrow$ the geodesic separating P_i and $P_{i+1(\text{mod } 3)}$
    $P_{i,i+1} \in \mathbb{R}^3 \leftarrow$ intersection of g_i with the geodesic connecting P_i and $P_{i+1(\text{mod } 3)}$
  end for
  for $i \leftarrow 0$ to 2 **do**
    $F_i = (Q, P_{i,i-1(\text{mod}3)}, Q', P_{i,i+1(\text{mod}3)}) \leftarrow$ the face of that contains P_i
  end for
  for each additional neighbor $N_i \in N, i \geq 3$ **do**
    $P_i \in \mathbb{R}^3 \leftarrow$ intersection of half-line from C to N_i with S
    $F_j, j < i \leftarrow$ face of the quad-partitioned sphere containing P_i (note F_j contains P_j)
    $P_{i,j} \in \mathbb{R}^3 \leftarrow$ mid-point of the geodesic connecting P_i and P_j
    Connect $P_{i,j}$ with two non-adjacent vertices in the face F_j, and modify the face F_j accordingly (see discussion below).
  end for
  return partition of sphere into quads
end function

A relevant question to examine is how to best adjust the position of the quad vertices on the sphere. In our implementation, after introducing a new point P, we used the following heuristic to reposition each vertex q of a new quad containing P : we look at each quad on the sphere containing q. Each quad corresponds to a point P_i on the sphere, so the new position of q should be equidistant to all sphere points P_i. If q has just two neighbor quads, that is, there are 2 such P_i's we take the midpoint on the geodesic between P_{i_1} and P_{i_2}. If q belongs to three quads, that is, there are 3 P_i's, we take the closest equidistant point (as shown above for Q in Figure 3), if there are more than 3 P_i's we take q equidistant to the P_i's in a least square sense (cf. Figure 4). In the examples we treated, the so adjusted spherical quads remained an appropriate partition of the sphere: each face would contain only one adjacent skeleton edge. It is nonetheless not clear how to prove such a method

works in general. Other possible criteria of adjustment could regard minimizing the twist along the branches, or the convexity of the faces.

5 Constructing *sleeves* around skeleton edges

Algorithm 1 described a partition of each sphere into spherical quads. We now describe the "sleeving" procedure that will connect these partitioned spheres together, and the partitioned spheres to the extremities (see Algorithm 2).

Recall from Section 3 that a skeleton *branch s* is defined by a sequence of connected vertices $\{v_0,\dots,v_l\} \in V$ such that v_0 is a *branch point*, v_l is either a *branch point* or an *extremity*, and v_k, for $0 < k < l$, is a joint. The sequences of nodes that describe the branches is recovered by a depth first traversal starting from any extremity (see Section 3).

5.1 Algorithm: creating a sleeve along the branch

The idea behind creating a sleeve around a branch is to avoid creating a twist in the quadrilateral section, as we go along the branch. For this purpose, rotation minimizing frames have been proposed (see e.g. [5] for a review of recent minimizing frame methods) and require the definition of a frame attached to the branch.

Defining a frame at v_0

The branch s corresponds in a unique way to a spherical quad (p^1,p^2,p^3,p^4) in the partition of the sphere around the branching point v_0. The first step is to associate to (p^1,p^2,p^3,p^4) a local orthonormal frame that approximates this quad in two ways. The frame should follow the direction of the branch and needs to approximate the plane defined by the corresponding quad. First, we define z_0 the unit vector that goes out of the branch point and is parallel to the first edge of the branch. In order to orient the axes x and y on the plane orthogonal to z, we use the following heuristic: We define the auxiliary vector $t = p^1 - p^2$, and compute the x-axis as $x_0 = z_0 \times t$ and the y-axis $y_0 = x_0 \times z_0$ (where $\times$ is the cross product). We define the aligned spherical quad $(q_0^1 = p^3, q_0^2 = p^4, q_0^3 = p^1, q_0^4 = p^2)$ and the orthonormal local frame (x_0,y_0,z_0) associated with the first vertex in the sleeve.

This original frame is then propagated along the branch. There are two possibilities: the branch either terminates with an *extremity*, or with a *branch point*. This approach described above works well for all of our examples: the created frame is well oriented for the starting spherical quad, that is, no twisting happens on the set of starting faces (see Figure 5).

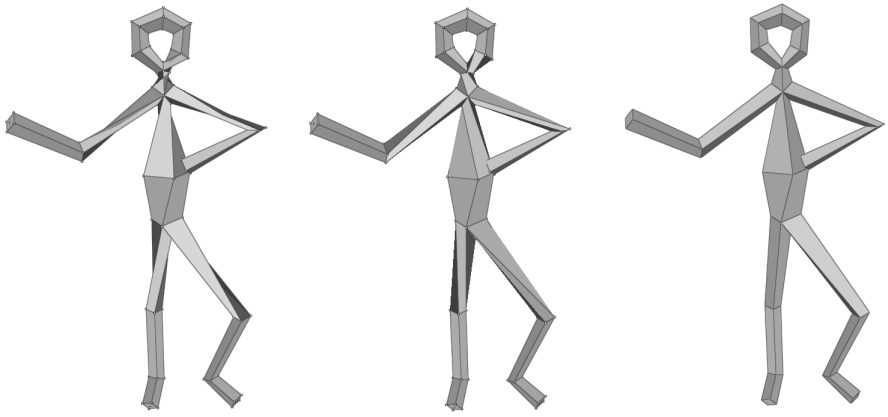


Fig. 5 Three meshes based on the same skeleton, the first two exhibit twisting, the last one is the output of our algorithm that minimizes the twist.

Branches ending at *extremities*

For branches ending at extremities, as in for example the leg of the stick woman (Figure 1), we require a method to distribute the necessary twist along a branch of a skeleton. Such methods appear for instance in [20, 21]. We use the algorithm described in [21, page 388] to propagate the 0-th frame over the branch. The algorithm contains a technique, the so-called Double Reflection Method, which tries to minimize the rotation of the propagated frames along consecutive sampled points of a curve in 3D. In our case the sequence of points is the sequence of vertices of the branch, all joints outside of the end points. For every vertex v_i, $1 \leq i \leq l$, we define the associated tangent vectors z_i as a unit vector following the direction $((v_i - v_{i-1}) + (v_{i+1} - v_i))$, that is, the average direction between the inward and outward edges of the joint, which is the z-axis of the local frame. Following [20], the local orthonormal frame (x_i, y_i, z_i), associated with the node v_i is computed by applying two consecutive reflections of the frame $(x_{i-1}, y_{i-1}, z_{i-1})$ of the previous vertex v_{i-1} of the branch. Specifically, $x_i = R_2 R_1 x_{i-1}$, where $R_j = I - 2(n_j n_j^T)/(n_j^T n_j)$, $j = 1, 2$ are the reflections relative to the planes with normals $n_1 = v_i - v_{i-1}$ and $n_2 = (v_i + z_i) - (v_i + R_1 z_{i-1})$ respectively. The vector y_i is then the cross product of the two axis z_i, x_i, i.e., $y_i = z_i \times x_i$.

Next, we propagate the spherical quad from the originating branch node of the branch along the axis z_i and on the plane defined by the other two axes x_i, y_i. Specifically, given that we have a well oriented frame to the quad of the node v_{i-1}, we define the next (propagated spherical) quad $(q_i^1, q_i^2, q_i^3, q_i^4)$ of the node v_i of radius r_i and with local frame (x_i, y_i, z_i) as:

$$q_i^1 = v_i + r_i x_i + r_i y_i$$

$$q_i^2 = v_i + r_i x_i - r_i y_i$$
$$q_i^3 = v_i - r_i x_i - r_i y_i$$
$$q_i^4 = v_i - r_i x_i + r_i y_i.$$

Branches ending at another *branch point*

In the setting where the last node of the branch is not an extremity but another *branch point* v_l, we need to match the last generated propagated spherical quad to the associated spherical quad. Since v_l is a branch point, a spherical quad corresponding to the edge (v_{l-1}, v_l) is already defined from the sphere partition at v_l (as detailed in Section 4); we denote it $(p^i)_{i=1\ldots4}$. If needed, we orient this quad (coming from the partition at v_l) consistently by minimizing the euclidean distance between every vertex of the propagated quad $(q_l^1, q_l^2, q_l^3, q_l^4)$ and the assigned spherical quad (p^1, p^2, p^3, p^4) (four possibilities are considered, offseting mod 4 the four indices of the quad vertices in reverse order). We update the indices of the $(p_i)_{i=1\ldots4}$ to match the indices of the $(q_l^i)_{i=4\ldots1}$.

Generating the scaffold mesh

For each edge, in each branch, we create four new faces that are associated with the edge (v_{i-1}, v_i). The four scaffold quads are:

$$(q_i^1, q_i^2, q_{i-1}^2, q_{i-1}^1), (q_i^2, q_i^3, q_{i-1}^3, q_{i-1}^2), (q_i^3, q_i^4, q_{i-1}^4, q_{i-1}^3), (q_i^4, q_i^1, q_{i-1}^1, q_{i-1}^4).$$

If the branch ends with an extremity we generate an additional scaffold quad to close the mesh: $(p_l^4, p_l^3, p_l^2, p_l^1)$.

6 Implementation and results

The implementation was written in Matlab and C++, with examples generated in Rhino3D/Grasshopper3D. One consideration in viewing these results is that the scaffold is intended as a base mesh, and is useful for its topology and orientation but is not a final mesh representing the geometry of intended shape sketched by the skeleton. The visual appearance of the result is thus not the emphasis of the current project. The emphasis is on the coarseness of the generated mesh that is free of any T-junctions: the scaffold is obtained by connecting the spherical quads at each end of a branch (i.e. between two branch points or a branch point and an extremity). We illustrate in this section the effectiveness of our algorithm on several examples.

Figure 6 shows a spiral starting at a degree 3 node, and illustrates the rotation minimizing section along the long branch.

Algorithm 2 Sleeving the branches

Require: A skeleton (G,p,r), an ordering of the vertices given by the depth first search, and a partition of each sphere corresponding to a branch point into spherical quads.

function CREATESLEEVES(G)
  for each branch in G given by the sequence $v_0,\ldots,v_k$ of the skeleton vertices **do**
    $p_1 \ldots p_4 \leftarrow$ the face in the partition of the sphere at v_0 associated with the neighbor v_1
    Create a local orthonormal frame (x_0,y_0,z_0) for the v_0
    $q_0^1 \leftarrow p_3$
    $q_0^2 \leftarrow p_4$
    $q_0^3 \leftarrow p_1$
    $q_0^4 \leftarrow p_2$
    Align the frame to the spherical quad for the vertex v_0, $\{p_j^1 \ldots p_j^4\}$ if needed
    for each node $v_i, i > 0 \in s_j$ and corresponding radius r_i **do**
      Create the local frame (x_i,y_i,z_i) at v_i, using the 'Double Reflection Method' [20].
      $q_i^1 \leftarrow v_i + r_i * x_i + r_i * y_i$
      $q_i^2 \leftarrow v_i + r_i * x_i - r_i * y_i$
      $q_i^3 \leftarrow v_i - r_i * x_i - r_i * y_i$
      $q_i^4 \leftarrow v_i - r_i * x_i + r_i * y_i$
      if $i = k$ and v_k is a branch node **then**
        Align the propagated spherical quad associated with node v_k to the frame.
      end if
      Add to the Mesh the four scaffold quad faces $f_1,\ldots,f_4$:
      $f_1 \leftarrow (q_i^1, q_i^2, q_{i-1}^2, q_{i-1}^1)$
      $f_2 \leftarrow (q_i^2, q_i^3, q_{i-1}^3, q_{i-1}^2)$
      $f_3 \leftarrow (q_i^3, q_i^4, q_{i-1}^4, q_{i-1}^3)$
      $f_4 \leftarrow (q_i^4, q_i^1, q_{i-1}^1, q_{i-1}^4)$
      if $i = k$ and v_k is an extremity **then**
        Add to the Mesh face $f_5 = (q_i^4, q_i^3, q_i^2, q_i^1)$
      end if
    end for
  end for
  return Coarse quad mesh surrounding the skeleton
end function

The second example consists of a simple tree skeleton, in two cases. First, we consider non-intersecting spheres with constant radius at each branch point of the skeleton. The skeleton and resulting scaffold are shown in Figure 7. We then assign to the branch point and joints a larger radius leading to intersecting spheres (Figure 8(b)). The output is shown in Figure 8, and we note that the mesh generated is not self-intersecting.

Figure 9 is the quad scaffold of a skeleton with branch points of degree 4 and 3 (and varying radii), and a graph with a cycle. The 'Stick Woman' illustrates the simplicity of the generated mesh, as every edge of the graph generates only 4 quads on the scaffold; an additional quad at each extremity is created to make a closed manifold mesh.

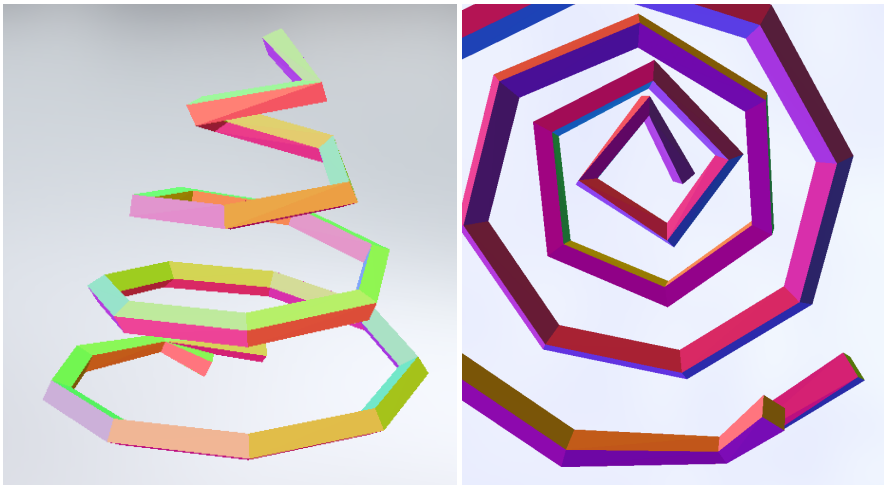


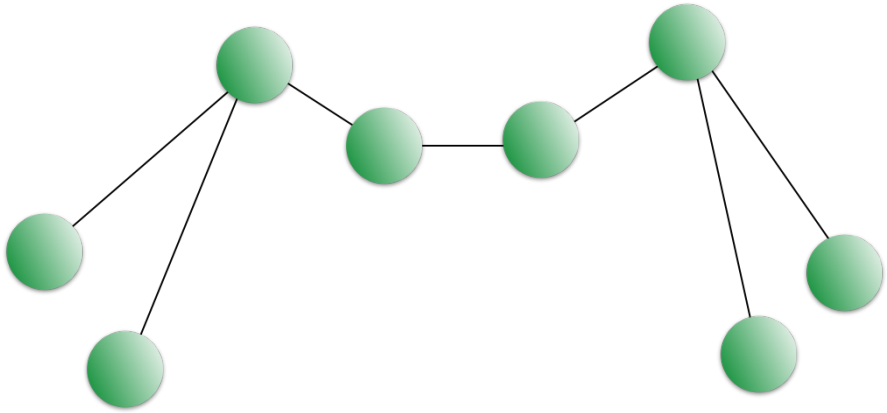
Fig. 6 A simple skeleton with constant radii and 2- and 3-valent vertices. We can see in the long spiral the consistent rotation of the frame along the turning sleeve.

7 Conclusion, limitations and future work

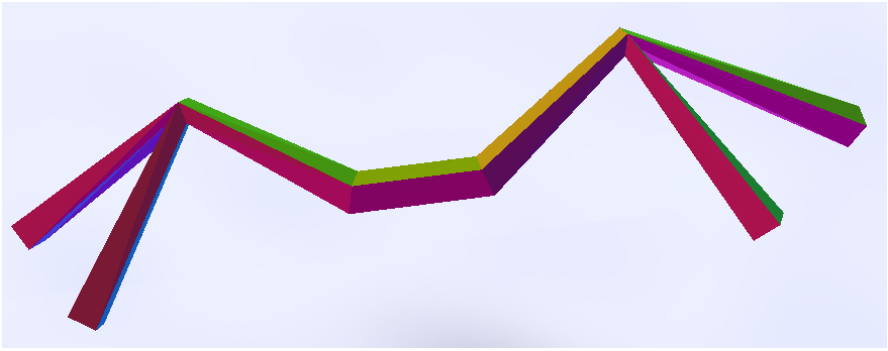
In conclusion, we have provided an algorithm for generating a coarse mesh around a skeleton. The constructed scaffold is a first step to skinning the skeleton with an explicit representation and offers several advantages. In particular, we guarantee an exclusively quad mesh. This was not possible by the method of [2]. In addition, by construction we completely avoid T-junctions in the mesh, which was one of the goals of [18]. In that paper, T-junctions occur in a preliminary stage of the mesh creation, and are then removed by a rather involved process; moreover, some difficult cases remain. It would therefore be an advantage to use our proposed scaffold as an initial step for both work.

One of the motivations for this project comes from the desire to provide an explicit surface representation of a *convolution surface*. Recent work has used techniques from symbolic computation to provide simplified descriptions and efficient computation of a convolution surface based on a skeleton [9]. However, although these methods give very satisfying smooth surfaces, they lack an explicit (e.g. mesh-based) representation of the resulting surfaces, and are therefore computationally expensive to render. Typically a method such as the marching cubes algorithm is required to render the surface. The scaffold mesh constructed in the present work could be used to map a set of representative points from the convolution surface, which would lead to generating a mesh, that is, an explicit representation of the surface.

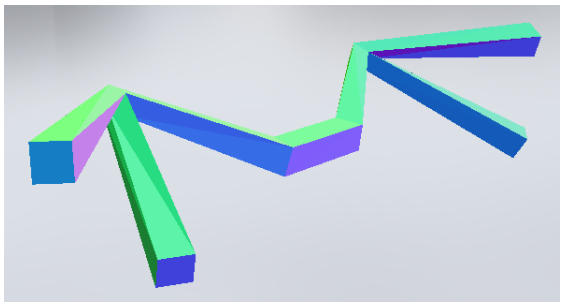
Several aspects of the present work remain open for investigation. While the sphere partition algorithm of Subsection 4 guarantees a partition of the sphere into



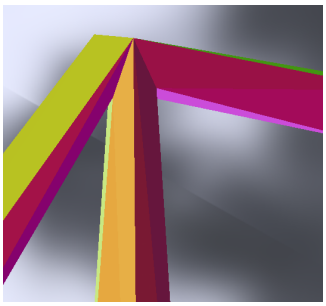
(a) A skeleton with equal radii at each vertex



(b) The uniform quad scaffold mesh around the skeleton

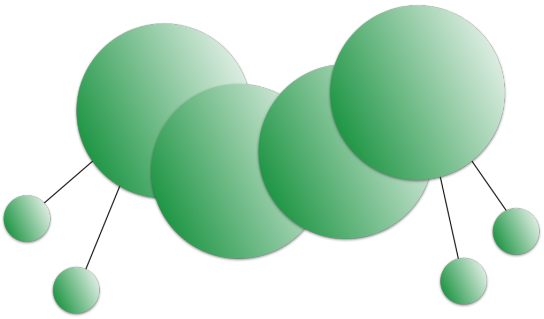


(c) The scaffold mesh from a different viewpoint

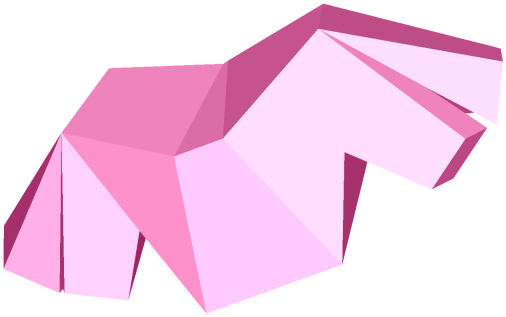


(d) Around a *branch point*

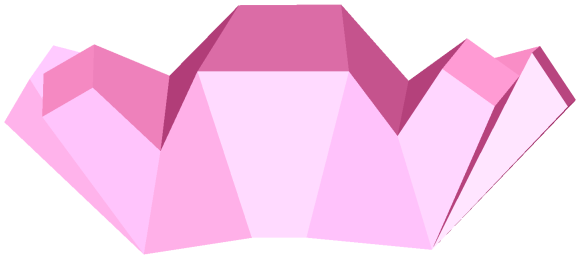
Fig. 7 A Scaffold simple skeleton with unit radii and 2- and 3-valent vertices. Note the regularity and stitching of the mesh faces around branch points (d).



(a) A skeleton with intersecting spheres



(b) Scaffold mesh



(c) Scaffold mesh

Fig. 8 The simple skeleton of Figure 7 with varying radii. The large radii lead to intersecting spheres at the interior vertices, but the faces remain quads.

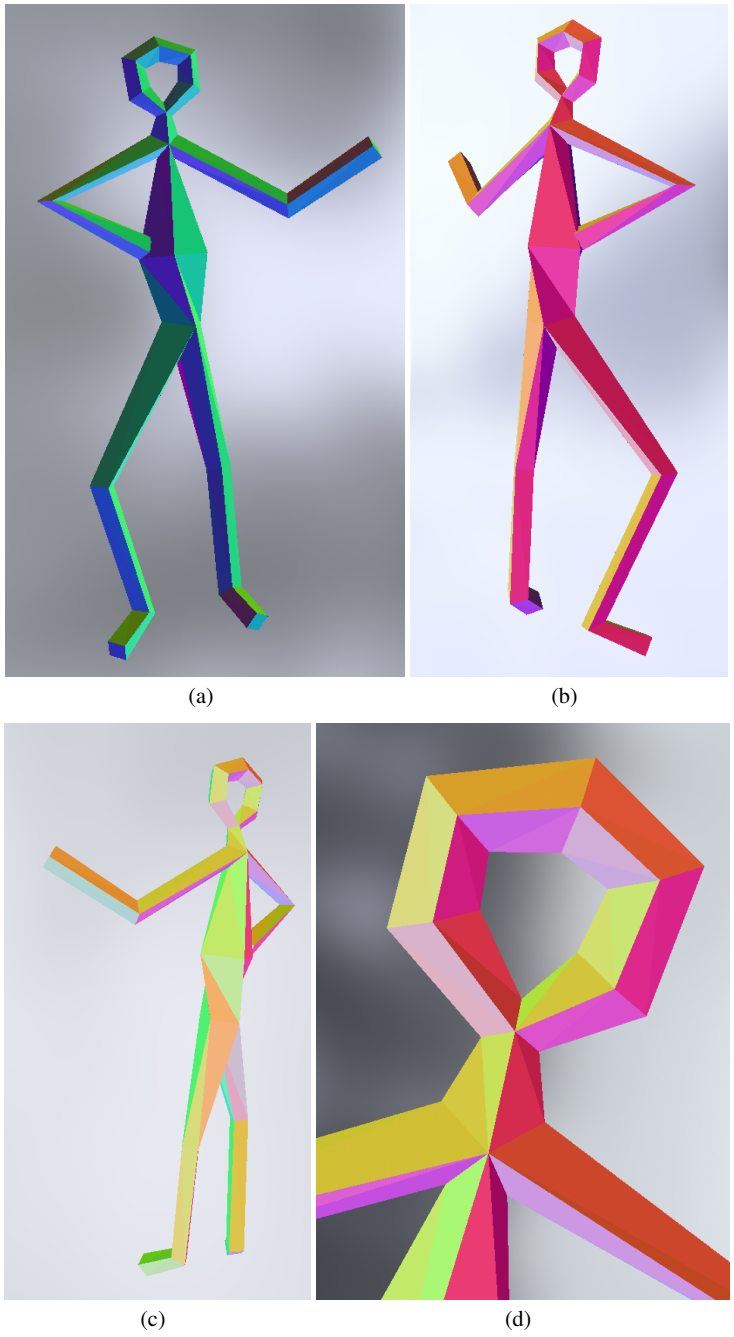


Fig. 9 The scaffold mesh created from the skeleton in Figure 1; all faces of the mesh are quads, as well as the section of any sleeve.

spherical quads, that partition is not necessarily unique. In particular, the partition depends on the order in which the branches are considered. Further investigation is required to determine what methods may be available to either standardize this partition (by sliding the vertices appropriately on the surface of the sphere) or to potentially allow some element of user-interaction to guide the partition. This may be desirable, for example, in a computer graphics application, where a skeleton may represent an object such as a hand or a creature, and one particular partition of the sphere may enhance its appearance. If the skeleton changes, but not its topology, our algorithm always produces a mesh with the same number of vertices and faces, something that is useful for simulation related applications. On the other hand, when the topology of the skeleton changes we do not have any intuition about how the mesh changes. It is an interesting direction to investigate for animation related applications.

Another consideration for further work is the rigorous treatment of the case of intersecting spheres at the nodes of the skeleton. Finally, the technique that provides an initial frame for the spherical quads needs further research to guarantee that the resulting frame-quad combination is always well aligned. For that reason, a relaxation of the positions of the nodes could be used as a post-processing to improve the result.

Acknowledgements This work has been partly supported by the French Research National Agency (ANR) program CIMI, ANR-11-LABX-0040-CIMI and the German Research Foundation (DFG) within the priority program SPP 1962 under contract number Schu804/15-1.

References

1. M.R. Adhikari, *Basic algebraic topology and its applications*. Springer, 2016.
2. J.A. Bærentzen, M.K. Misztal, and K. Welnicka. Converting skeletal structures to quad dominant meshes. *Computers and Graphics*, 36(5):555-561, 2012.
3. J.A. Bærentzen, R. Abdrashitov, and K. Singh. Interactive shape modeling using a skeleton-mesh co-representation. *ACM Transactions on Graphics (TOG)*, 33(4), Article 132, 2014.
4. M.-P. Cani, S. Hornus. Subdivision curve primitives: a new solution for interactive implicit modeling. In: International Conference on Shape Modeling and Applications. IEEE Computer Society Press, pp. 82–88, 2001.
5. R.T. Farouki. Rational rotation-minimizing frames Recent advances and open problems. Applied Mathematics and Computation, 272, pp80-91, 2016.
6. W.B. Goh. Strategies for shape matching using skeletons. *Computer vision and image understanding*, 110(3):326-345, 2008.
7. M.P. Hitchman. *Geometry with an introduction to cosmic topology*, Jones and Bartlett Publishers, 2009.
8. S. Hornus, A. Angelidis, M.-P. Cani. Implicit modelling using subdivision curves. Visual Comput. 19 (2-3), 94–104, 2003.
9. E. Hubert and M.P Cani. Convolution surfaces based on polygonal curve skeletons. *Journal of Symbolic Computation*, 47(6):680-699, 2012.
10. Z. Ji, L. Liu, and Y. Wang. B-Mesh: A Modeling System for Base Meshes of 3D Articulated Shapes. *Computer Graphics Forum*, Volume 29, Number 7, Blackwell Publishing Ltd, pp. 2169-2177, 2010.

11. Jin, X., Tai, C.-L., Feng, J., Peng, Q.. Convolution surfaces for line skeletons with polynomial weight distributions. ACM Journal of Graphics Tools 6 (3), 17–28, 2001.
12. I. Lakatos, *Proofs and refutations*, Cambridge University Press, 1976.
13. M. Raptis, D. Kirovski, and H. Hoppe. Real-time classification of dance gestures from skeleton animation. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation, pp. 147-156. 2011.
14. V. Srinivasan, E. Akleman and J. Chen, Interactive Construction of Multi-Segment Curved Handles, Proceedings of Pacific Graphics, Beijing, China, 2002.
15. V. Srinivasan, E. Mandal and E. Akleman, Solidifying Frames, Bridges: Mathematical Connections in Art, Music, and Science 2004, Banff, Canada, 2005.
16. H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. *Shape Modeling International*, IEEE, 2003.
17. A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. 3D Skeletons: A State -of-the-Art Report. In: J. Madeira, G. Patow, and T. Romão, editors, *Eurographics 2016*, 35(2):2016.
18. F. Usai, M. Livesu, E. Puppo, M. Tarini, and R. Scateni. Extraction of the Quad Layout of a Triangle Mesh Guided by Its Curve Skeleton. *ACM Transactions on Graphics (TOG)*, 35(1), Article 6, 2015.
19. H.-B. Yan, S.-M. Hu, and R. Martin. Skeleton-based shape deformation using simplex transformations. In: H.-P. Seidel, T. Nishita, and Q. Peng, editors, *Advances in Computer Graphics*, LNCS 4035, Springer, pp. 66-77, 2006.
20. W. Wang, B. Jttler, D. Zheng, and Y. Liu. 2008. Computation of rotation minimizing frames. ACM Trans. Graph. 27, 1, Article 2 (March 2008)
21. W. Wang, B. Joe. 1997. Robust Computation of the rotation minimizing frame for sweep surface modeling. Computer Aided Design, 29:5 pp 379–391.
22. Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: a sketching interface for 3D freeform design. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)