Abstraction & Application **11** (2014) 1 − 18 UADY

Sphere Approximation for Non-tubular Objects

Francisco A. Madera[a], Francisco Moo Mena[b], Enrique Ayala Franco[c]

Facultad de Matemáticas, Universidad Autónoma de Yucatán, México

[a]mramirez@uady.mx, [b]mmena@uady.mx, [c]enrique.ayala@uady.mx

Resumen

Estudiamos el problema de aproximar un objeto tridimensional con esferas. Empleamos un disco que consta de rayos. El disco recorre el objeto para detectar colisiones con la malla poligonal. Los puntos de contacto de los rayos con los polígonos sirven para determinar las esferas que se crearán dentro del objeto. Para eso, debemos descomponer la malla en parches bidimensionales para que sean aproximados con esferas. Posteriormente, las esferas son movidas hacia el centro del disco y se reducen de tamaño para minimizar el error volumétrico. Este método es evaluado con 3 modelos de humano virtual.

Abstract

We study the problem of approximating a $3D$ object with a union of overlapping spheres. We employ a disk filled with rays that comes from the center of the disk and points outwards. The disk traverses the object to detect collisions with the mesh. These contact points serve to determine the spheres to be created in the boundary of the object. This adjustment is performed by decomposing the volumetric mesh in thin surfaces, which are bounded with a sphere each. Afterwards, the spheres are moved towards the center of the disk and are scaled down to minimize the volumetric error. We evaluate our method with three human models, in particular the torso of the objects.

Keywords and phrases : Computer Graphics, Sphere Approximation, Computational Geometry

2010 *Mathematics Subject Classification*: 68U05, 65D18, 68W25

1 Introduction

Important characteristics of 3D models are their geometry and topology. For the mesh representation, the geometry information is computed from the vertex position, and the topology information can be computed from the mesh connectivity. Representing a complex geometric object with primitives is a fundamental task in computational geometry and computer graphics. The choice of the primitives depends on the application: sphere packing, tilings, triangulation, FEM (Finite Element Method), DEM (Discrete Element Method). Filling objects densely with smaller volumes is a highly non-trivial task and still an active field of research, even when restricted to spheres.

We propose to traverse an object mesh with a disk in order to obtain information about the mesh primitives. This work is an extension of [1] from tubular to non-tubular objects. Spheres keep adjacent to the mesh in such a case that animation can be performed and the spheres remained adjusted. Thus, the method can be utilised for both, rigid and deformable objects.

Consider a disk filled with rays, where each ray starts from the center of the disk and points outwards. A tubular surface can be tracked by a disk to record information of the mesh and thus construct spheres to aproximate such tubular surface. For the case of non-tubular surfaces, some regions become wider and narrower so that the aproximation with only one sphere is not suitable in the disk motion step. However, the surface still can be tracked by the disk, regarding two or more spheres in each disk motion step. Let Ω denote the surface of a closed simple object in $3D$. Consider the largest sphere $\bigcirc$ inside Ω. Obviously, $\bigcirc$ touches at least 4 points of Ω, and there are no other points of Ω inside $\bigcirc$. A volumetric object can be traversed by a disk to record information about its mesh. The mesh of a volumetric object can be partitioned in patches. A patch is a $2D$ mesh that can be approximated by a sphere. Afterwards, spheres are created, minimizing the Hausdorff distance between the sphere and the patch. We present a simple algorithm that offers a good efficiency to achieve the accuracy of the mesh aproximation using spheres.

Our first discovery is the construction of the filled disk that records information of the object mesh. Our second discovery is the method to reduce the volumetric error by applying geometric transformations to the spheres. To verify the feasibility of our technique, three human models were tested, in particular the torso of the objects. Our data structures are mainly focused on the disk and spheres. Indeed, all we need is a notion of distances between points.

We choose human meshes for experiments due to character animation are extensively used in games and visual effects. Our main contribution includes:

- A disk traversal method to record mesh information
- A method to approximate non-tubular objects with spheres
- A technique to minimize a sphere given a set of points and a surrounding region

Compared with existing approaches, our method has the following advantages:

- Spheres cover the convex polygons of the same patch
- Minimum size of spheres
- Spheres are able to be adjusted to the corresponding patch
- Small volumetric error

Related work is described in the next section. In Section 3 the problem is defined. In Section 4 we show how to compute the circumference and move it along the mesh. The circumference contains a number of rays that start from the center and point out to the mesh to detect the surrounding region. The mesh is traced by the disk during its motion. The algorithm implementation is detailed in Section 5, spheres are constructed according to the data recorded by the disk. Experiments are described in Section 6. Finally, conclusion and further work are described in Section 7.

2 Related Work

Approximation of objects using basic primitives has been studied for years. In [19] an approximation of the object by a union of balls is given. They collect a set of occluding contours and the result is a cloud of point sampling the axes of the generalised cones composing the object shape. BVH (Bounding Volume Hierarchy) has beeen used to approximate objects in many applications [21, 22]. Bounding spheres for linear blend skinning are employed in [4], where spheres are reffited during animation to detect collisions between humans.

Some algorithms make an approximation by filling or packing the object with basic primitives. The key to working with circle packings lies in recognizing their dual natures: combinatoric on one hand and geometric on the other. Circle packings and their related geometry have a wide variety of applications such as CAD (Computer Aided Design) and optimization problems.

Weller et al. [2] proposed a novel method for filling arbitrary objects very quickly and stably with sets of non-overlapping spheres. Such algorithm was able to efficiently compute a space filling sphere packing for arbitrary objects. Shimada and Gossard [3] developed a circle-packing method called bubble mesh to generate triangular meshes for two and three dimensions. Their packing scheme is based on the simulation of the particles that interact each other with repulsive and attractive forces. In constrast with [4] where the refitting of bounding spheres for spherical blend skinning with sublinear time complexity (with respect to the number of vertices) is applied, our method automatically update the size of the spheres due to some vertices of the mesh are utilised to update the sphere size.

Voronoi approaches to bound a $3D$ object with spheres can be found in [5, 6, 8]. Müller and Chentanez [6] connected particles to form a simulation mesh. These particles are represented by anisotropic shapes such as ellipses which are replaced by a sphere tangent.

Tubular objects are still being studied. In [9], a tracing algorithm for branched curvilinear structures is employed and applied it to plant photomorphogenetic analysis. Mistelbauer et al. [20] worked on vessel visualization by using the centerline and radii similar to [1] for tubular objects, which are traced by spheres to determine the correct shape of the roots. A $4D$ iterative key point searching method is proposed and utilized to detect multi-branch tubular structures with only one initial point [10]. In [11], an extension to deformable shapes is implemented; this enables the packing program to simulate structural changes due to melting and sintering. Circle Packing meshes also constitute a link between architectural freeform design and computational conformal geometry [12].

In the field of medicine, a geometric packing generation algorithm has been presented in [13]. It is based on a tetrahedral mesh to make an isotropic and dense packing of polydisperse spheres in a short computation time. The filling of vessels are performed for simulation by inflating balls is used to treat stenosis, a partial or total blockage of an artery [24]. Pianet et al. [14] compare different ways of using the DEM for consolidating and compressing particle packings in the context of paper-coating applications. Filling with different $3D$ shapes can be found in [23], where Shier and Bourke fill any spatial region with a random fractalization.

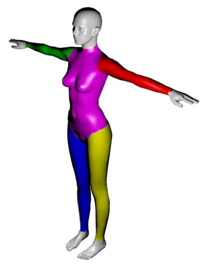
3 Problem Definition

A mesh of an object is defined as $\mathcal{M} = (\Delta, V)$, where $\Delta = \{\Delta_0, \Delta_1, ..., \Delta_{n-1}\}$ is the set of polygons, $V = \{v_0, v_1, ..., v_{r-1}\}$ is the set of vertices, and $\Delta_i =< v_j, v_k, v_h >$ is a polygon triangle. A region of the mesh $\mathcal{R}_i$ consists of a set of polygons $\Delta_i \in \Delta$. A human mesh is divided in five regions as suggested in [16] and illustrated in Figure *1*. The regions are as follows: the left arm $\mathcal{R}_0$, the right arm $\mathcal{R}_1$, the left leg $\mathcal{R}_2$, the right leg $\mathcal{R}_3$, and the torse $\mathcal{R}_4$. Let $\bigodot(\mathcal{R}_i)$ be the set of spheres to approximate region $\mathcal{R}_i$. The first four regions are $\bigodot(\mathcal{R}_0)$, $\bigodot(\mathcal{R}_1)$, $\bigodot(\mathcal{R}_2)$, $\bigodot(\mathcal{R}_3)$. Now we are focusing on the next region $\mathcal{R}_4$, being the goal to find $\bigodot(\mathcal{R}_4)$. Since triangles and bounding spheres are always convex, it is sufficient to enclose only the vertices of a triangle in order to bound the whole triangle.

Accordingly, the problem can be enuntiated as follows: Given a non-tubular $3D$ mesh, find a set of spheres to approximate it. The mesh to be considered is the torso region of an object. The goal is to obtain an accurate approximation and keep it during the object animation.

4 The Algorithm

Rather than using a wrapped or layered approaches [17], we are focusing in the curvature of the mesh points that form a convex primitive. We partition the volumetric object in patches, specifically the surrounding region obtained from the disk translation is divided in several parts. Aside from this, we take the room left


(a) The human is divided in five regions

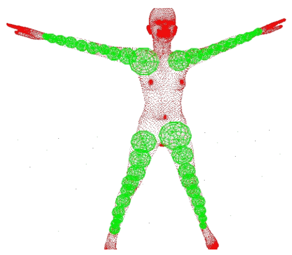

(b) The limbs filled with spheres

Figure 1: The human mesh is partitioned in five regions (a). Tubular regions are approximated with spheres (b) as proposed in [1].

by the other partitions of points to move the sphere and improve the accuracy of the approximation. The sets of points contain the same number of points. Human meshes have their own geometric features, but in general terms a human torso is a convex object similar to an ellipse. This means that a sphere is not a good approximation.

Firstly, the disk is created. The disk is placed on the bottom part of the torso and starts moving up along the torso. During this motion, the ray-triangle intersection test is performed with the rays of the disk and the polygons of the object's mesh. The disk traces the torso and determines the convenient locations to construct spheres to approximate the mesh. Secondly, the information recorded is utilised to create spheres. Contact points define a global set which is partitioned. We connect the points of each partition to form a convex polygon that is used to construct a circumference. A method to reduce the size of the spheres is proposed. A disk filled with rays is employed in [15] to construct cages that serve to decompose the human object, but they do not give details about the disk construction.

4.1 Disk Translation

Recognizing the geometry of an object requires several computations. By constructing a disk $*$, we can use its radius as a ray with starting point the center of the disk $*.c$ and a direction that points outwards. The disk is placed inside the object and at the bottom of $\mathcal{R}_4$ (Figure 2). The ray rotates α degrees in clockwise order in each time step. Rather than rounding the ray around $*.c$ with $\frac{360}{\alpha}$ degrees in each step, we construct the rays around $*.c$.

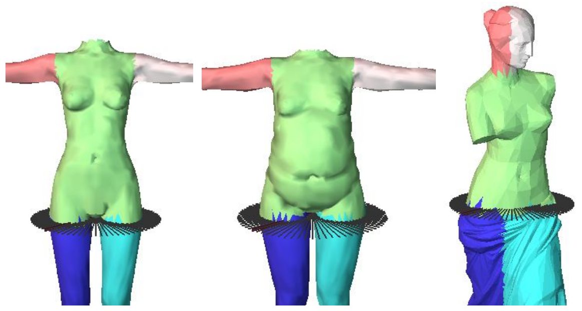
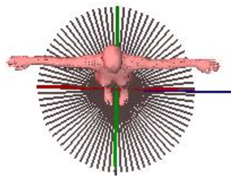

(a) Human1 (b) Human2 (c) Human3

Figure 2: The humans utilised in the experiments: Human1, Human 2, and Human3.

Assuming the human mesh in stand up pose following the $\vec{\gamma}$ vector, then the disk is placed on the $\vec{\gamma_0}\vec{\gamma_1}$ plane, where $\vec{\gamma} = \vec{\gamma_0} \times \vec{\gamma_1}$. Thus, we proceed to construct the rays. The start point of a ray r_i is the center of the disk $*.c$. The first ray r_0 points towards $\vec{\gamma_0}$, then we turn around α degrees in clock wise order to create the second ray r_1, and so on to complete the 360 degrees. We decided to have 80 rays, therefore the angle separation among them is of $\alpha = \frac{360}{80} = 4.5$ degrees. To obtain a well-formed shape disk, the two vectors $\vec{\gamma_0}$ and $\vec{\gamma_1}$ should be orthogonal as shown in Figure 3.


(a) A well formed disk

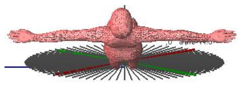

(b) A wrong formed disk

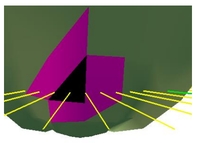
Figure 3: The disk lies on a plane formed with orthogonal vectors (a). A disk created with non-orthogonal vectors forms an incorrect shape (b).

We proceed to translate $*$ in direction of $\vec{\gamma}$, δ units: $*+ = \delta\vec{\gamma}$. The whole disk is translated, the center, the radius and the rays. δ can be determined by the Δ's size and the number of steps to move depends on the size of the mesh. A ray-triangle intersection test is applied in each step to detect the points in the mesh which are called the surrounding region of step h: $\circledast_h$.

We do not need a preprocessing phase as in [1], where the opposite triangles is needed; these are the pair of triangles joined with the normal vector of one of them. Even when the calculus of the opposite triangles serves to save time, in our case this calculus is not suitable since the shape of the object contains some inconsistencies, so that the disk rays computations is a better option to ensure the accuracy.

Lema 4.0.1 *There is at least one ray of the disk colliding with a polygon of a closed* 3*D mesh.*

Proof. Let the disk placed inside the object. As the mesh is a closed surface formed by polygons, and the rays of $*$ point outwards, then they need to leave the surface. Thus, ray r_i needs to collide with polygon Δ_j to leave the mesh. It can happen that two or more rays collide with the same triangle. If the object's mesh is closed, then all the rays collide with a triangle.


(a) Collision with two rays

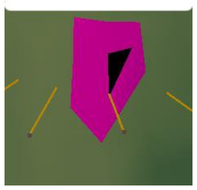

(b) No rays colliding

Figure 4: The black triangle collides with two rays (a), the black triangle does not collide with a ray (b).

A triangle can collide with several rays (Figure 4 (a)) or a triangle can not collide with a ray (Figure 4(b)). The number of collisions of the triangles with a ray is related to the number of rays of the disk. The larger the number of rays the greater the number of collisions. Definitively, many rays require more operations. However, there is not necessary to have all the triangles collided in each time step t of the disk motion $\circledast_h$ since in the further, when spheres $\bigcirc(C,r)$ are created, the closed region of a sphere is computed, taking the closed triangles to such sphere, and consequently all the triangles in the mesh would be considered.

If the object's mesh contains holes, then one or more rays could not be colliding with a triangle. It could happen that the region is an opened surface as shown in Figure 5 (a) or with holes (Figure 5, b). After the disk is translated, its center is re-computed to have it inside the mesh. This is calculated by taking the average of the positions of the surrounding region. We take the contact points of the triangles as the points

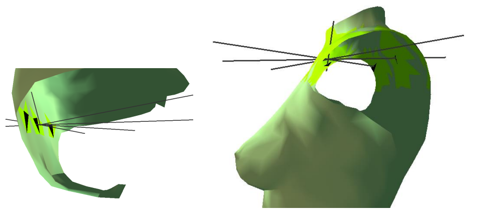

(a) Disk colliding with a surface (b) Disk colliding with a volume

Figure 5: The disk could not find collision with polygons and problems arise in the calculus of the surrounding region. (a) the region is not a volume, (b) the region contains holes.

of $\circledast$. By centering $*$, we guarantee that the disk keeps inside the mesh. Recording the centers of the disk along the surface, we obtain an approximation of a line of the mesh that can serve to construct the skeleton or the PCA (Principal Component Analysis) which could help to construct a skeleton of the object.

4.2 Sphere Construction

In Figure 6 (a) the rays are colliding with some polygons of the torso. We can appreciate the different levels of the disk position and determine their partitions (Figure 6(b)).

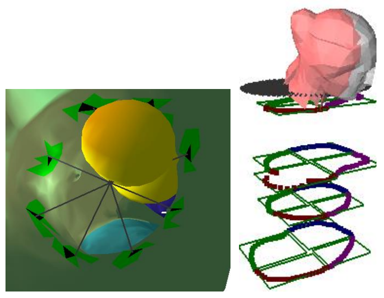
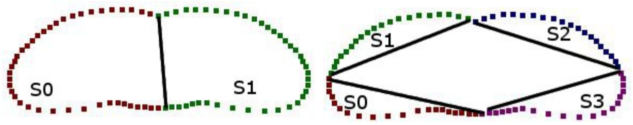

(a) Rays colliding with torso (b) Four partitions in several levels of the disk

Figure 6: The partitions of the torso: (a) rays colliding and (b) partitions in several levels of the disk position.

A volumetric mesh Ω can be decomposed in thin surfaces. The minimum volumetric object can be seen as a surrounding region, a set of connected polygons that form a rounding shape: $\Delta_0\Delta_1\Delta_2...\Delta_{n-2}\Delta_{n-1}$. Taking the contact points of these polygons, we obtain a set of points $S = \{v_0, v_1, ..., v_{n-1}\}$. The connection of these points, and the edge $\overline{v_0 v_{n-1}}$, form a polygon P. A partition of S is $S_0 = \{v_0, v_1, ..., v_{k-1}\}$. The connection of the points in S_0 form a polygon P_0 smaller than P (Figure 7).



(a) Two partitions (b) Four partitions

Figure 7: A surrounding region of the torso with two (a) and four (b) partitions.

Consequently, the problem is transformed: Given a set of points S we can compute a set of circumferences to bound them. The set of points is divided in p partitions, where circumference i bounds partition i. In this work we focus in two and four partitions. Figure 8 illustrates several number of spheres to cover a region of the disk.

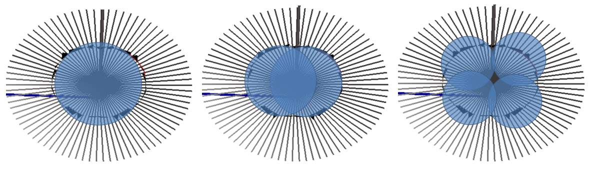


Figure 8: A top view of a torso region. The region is bound by (a) one sphere, (b) two spheres, (c) four spheres.

Convex objects are easier to approximate with a basic primitive such as a sphere, cube, OBB (Oriented Bounding Box), or K-DOP (K Discrete Oriented Polytope) among others. As objects lost their convexity in some patches, the approximation remains more complicated. This difficult can be solved by dividing the region mesh and approximate each partition with basic primitives.

4.3 Spheres and the Levels of the Disk

Observe that the polygons formed by the set of points $S_0, S_1, ..., S_{p-1}$ are convex so that they are suitable to be bounded by a circumference. The polygons formed $P_0, P_1, ..., P_{p-1}$ are bounded by a circle in the plane $\vec{\gamma_0}\vec{\gamma_1}$ and are part of S, the surrounding region of the disk in the current level $*_h$. As we are working with a $3D$ mesh, we should extend the circle to a sphere. A sphere created in this level will contain the points in $\circledast_h$; other closer points that belong to the adjancent levels $\circledast_{h-1}$, $\circledast_{h+1}$ are not considered since the partitions are formed with $S_0, S_1, ..., S_{p-1}$ of level h.

We could create the spheres regarding the points of the adjacent levels to h, $\circledast_{h-1}$, $\circledast_{h+1}$, or other nearer levels to h, $\circledast_{h-2}$, $\circledast_{h+2}$. Definitively, the sphere to be created will be bigger than the first approach, considering only level h, but we would ensure to cover corresponding mesh region. Another strategy is to consider more penetration between spheres of different levels in order to cover the points of adjacent surrounding regions: $h-1$, $h+1$, $h-2$, $h+2$. We proceed to implement the last approach (Figure 9).

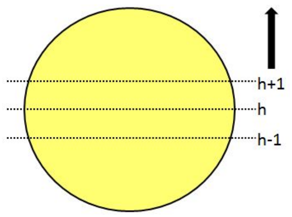
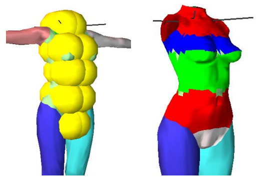


Figure 9: The levels of the disk motion.

Let h be the disk motion step that we call the level of the disk. At the start $(h = 0)$ we have two partitions in $*_0$, so that we construct spheres $\bigcirc_0$, $\bigcirc_1$. Moving the disk to the next level $*_{h+1}$ we compare the further spheres $\bigcirc_2$, $\bigcirc_3$ with $\bigcirc_0$, $\bigcirc_1$. If there is an intersection between the new spheres with the old spheres, then the new spheres are removed. Hence, we move to the next level $*_{h+1}$ to perform the same procedure until the new spheres are not overlapped with the old pair of spheres. In Figure 10, five levels are considered and four spheres for each level are built. Each level is indicated by the surrounding regions of their corresponding spheres.



(a) Sphere approximation (b) The five levels of the human torso

Figure 10: Five levels for the creation of the spheres (left), and their coressponding regions (right).

4.4 Sphere Adjustment

Even when two or more spheres can have a better approximation than using only a sphere, spheres can be adjusted to improve their accuracy. The motion of the circumferences takes place in the plane formed by $\vec{\gamma_0}\vec{\gamma_1}$ towards the $*.c$ direction. Doing this translation, the overlapping among the spheres is unavoidable, but this does not affect the accuracy as depicted in Figure 11.

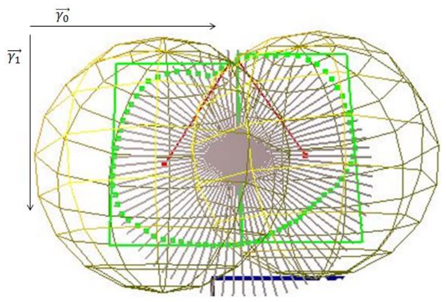


Figure 11: Spheres can be moved along the plane $\vec{\gamma_0}\vec{\gamma_1}$ towards $*.c$.

Assume to have two partitions. Given a set of points $S = \{v_0, v_1, ..., v_n\}$ we can obtain two circumferences to cover them. Firstly, we need to partition the set of points in two parts: $S_0 = \{v_0, v_1, ..., v_{\frac{n}{2}-1}\}$, and $S_1 = \{v_{\frac{n}{2}}, v_{\frac{n}{2}+1}, ..., v_{n-1}\}$. Polygons created are $P_0 = \{\overline{v_0 v_1}, ..., \overline{v_{\frac{n}{2}-2} v_{\frac{n}{2}-1}}, \overline{v_{\frac{n}{2}-1} v_0}\}$ and $P_1 = \{\overline{v_{\frac{n}{2}}, v_{\frac{n}{2}+1}}, ..., \overline{v_{n-2} v_{n-1}}, \overline{v_{n-1} v_{\frac{n}{2}}}\}$ (Figure 12).

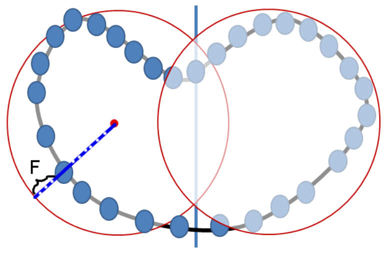


Figure 12: The surrounding region $\circledast_h$ is divided and approximated by two spheres.

Secondly, we have obtained a circumference that covers each partition by using a brute force method. Note that circumferences obtained have their center in the middle of the points, so that the volumetric error F returns a large value. The edges $\overline{v_{\frac{n}{2}-1} v_0}$ of P_0 and $\overline{v_{n-1} v_{\frac{n}{2}}}$ of P_1 are the largest edges due to they do not belong to the boundary of S. Then, we compute circumferences $\bigcirc_0$ and $\bigcirc_1$. F is the difference between the radius of a sphere and $\| *.c - v_i \|_2$ (Figure 12).

Thirdly, we reduce F by moving the spheres created along the plane formed by S. $\bigcirc_0$ can be moved towards $\bigcirc_1$ and viceversa. This movement means that $\bigcirc_0$ and $\bigcirc_1$ are still in the boundary of S and the accuracy to cover the corresponding set of points can be improved. Therefore, we can translate $\bigcirc_0$ towards

$\overrightarrow{\bigcirc_0\bigcirc_1}$, and also we can reduce its size. Let $\vec{N}$ be the vector that comes from the center of $\bigcirc_0$ and goes to the center of $*$, we could translate $\bigcirc_0$ towards $\vec{N}$. It could happen that some points lie outside $\bigcirc_0$, so that the translation will not be allowed. Decomposing $\vec{N} = \vec{N_0} \times \vec{N_1}$, we could try to move $\bigcirc_0$ towards $\vec{N_0}$, or $\vec{N_1}$. As a move of $\bigcirc_0$ is performed, we can try with the scaling transformation to reduce the circumference size. The procedure for $\bigcirc_1$ is analogous (Figure 13).

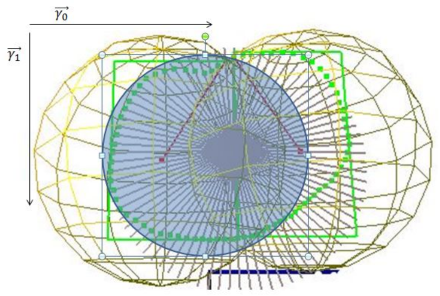
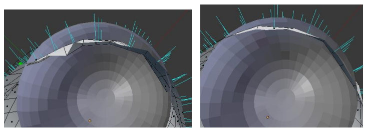


Figure 13: The left sphere can be scaled down to the size of the blue sphere.

Given a set of points divided in two parts bound by a circumference each, we can reduce F by moving the circumferences along the plane $\vec{\gamma_0}\vec{\gamma_1}$. Consider $\bigcirc_0$ and $\bigcirc_1$ that envelop S_0 and S_1. By translating $\bigcirc_0$ towards $\vec{N_0}$ and $\vec{N_1}$ we can vary the function F and achieve its minimization. Figure 14 illustrates the accuracy of the sphere approximation. Normal vectors of the triangle mesh are in blue. Figure 14 (right) shows better accuracy than the left sphere.



(a) Coarse sphere approximation (b) Tight sphere approximation

Figure 14: A sphere bounding a region of the mesh. The right picture shows a more accurate approximation than the left picture.

5 Algorithm Implementation

The stages of the algorithm are shown in the diagram of Figure 15. In the **Disk Construction** stage, the disk's rays are created: $r_i.x = *.c.x + *.r(\vec{N_0}.x \times *.r \times cos(i\pi) + \vec{N_1}.x \times *.r \times sin(i\pi))$. Analogously we can compute $r_i.y$ and $r_i.z$. At the start, rays are not colliding with a polygon, so that $r_i.\Delta = -1$.

The loop cycle involves three stages: the Disk Translation, the Ray Triangle Intersection Test, and the Disk Centered. In the **Disk Translation** stage, the disk is translated towards the direction of the normal vector to the plane of the disk: $\vec{\gamma}$. The disk is formed by a center and its rays with a lenght equals its radius. $r_i+ = \delta\vec{\gamma}, \forall r_i$, and $c+ = \delta\vec{N}$.

In the **Ray Triangle Intersection Test** stage the intersection of each ray with the polygons of the mesh is performed by applying the Ray Triangle Intersection test. $\forall r_i, \forall \Delta_j \in \mathcal{R}_4$ *Ray Triangle Intersection Test*(r_i, Δ_j). If the intersection takes place, then the polygon index is stored in the corresponding ray $r_i.\Delta = j$, the polygon is marked as busy $\Delta_j = 1$. The colliding point in the polygon is added to a variable $C+ = \Delta_j.c$ to be used in the next stage **Disk Centered** to update $*.c$ in order to keep it in the middle of the current surrounding region.

The contact points obtained by the Ray Triangle Intersection Test are stored in a data structure. Assume we have p partitions, then we keep the 80 contact points in the p data structures $S_0, S_1, ..., S_{p-1}$ for the current

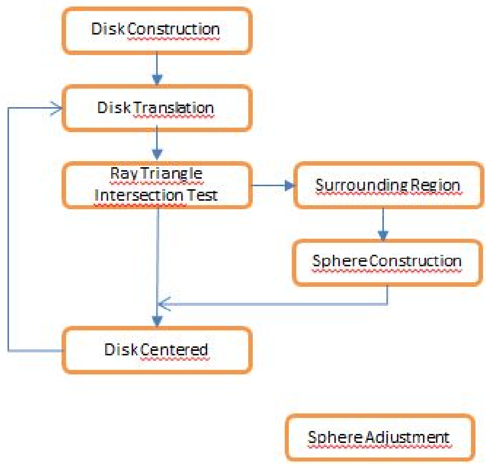


Figure 15: The stages of the algorithm.

level h of the disk. This is calculated in the **Surrounding Region** stage. The set of points are the input to the routine to construct spheres in the **Sphere Construction** stage. We implemented two methods to create a sphere. The first method $M1$ uses the brute force technique, the center of the circumference is the average of the set of points. The second method $M2$ is the minimum enclosing ball proposed by Welz [25]. The idea is to adjust spheres from $M1$ and compare the result with $M2$.

In the **Sphere Adjustment** stage, spheres created are moving towards the center of the correspondent patch to increase the accuracy as described in Section 4.4. This stage is not connected to the other stages due to it is not essential, but definitely it is convenient.

6 Experimental Results

Our experiments were run on a PC with a 2.40GHz Xeon processor, 12 GB main memory with Windows 7 operating system. The algorithm was coded in C++ with OpenGL API. We employed 3 human meshes (Figure 2). The first two humans were constructed using MakeHuman [18] so that they have the same number of polygons $(26, 292)$, while the third object $(11, 240$ polygons) was downloaded from the Stanford University repository.

The accuracy of the spheres created can be tested by the distance measure F among the points of the surrounding region and the spheres that covers such region. The parameters considered are as follows: number of rays in the disk (N_r), number of Spheres per level (N_s), displacement of the disk δ, and penetration depth PD. We take $N_r = 80$ and $\delta = 0.25$. Penetration depth refers to the overlapping spheres, that is $PD = 0$ if non-overlap exists, and 0.50 if the half of the spheres are overlapped. We mention the contact point of r_i with the mesh to refer of the accuracy of a method.

Our experiments were conducted to measure the accuracy of the spheres created. We employed $N_s = 1, 2, 4$, and $PD = 30\%$. For each N_s we perform the calculus of the spheres by the first $(M1)$ and the second $(M2)$ methods. Then we adjusted $M1$ to enhance the accuracy ($M1$-A) using the **Sphere Adjustment** routine.

Thus, we start employing $N_s = 1$, the number of contact points is 80, and the comparison is made between $M1$ and $M2$ since the spheres adjustment are not allowed with $N_s = 1$. Human3 contains two spheres in the torso, the bottom sphere (sphere 0) has a better approximation due to its rounding shape. From picture 26 we observe that $M1$ has a better approximation than $M2$ in r_0 - r_{18}, then $M2$ is better from r_{19} - r_{49}, and finally $M1$ is better in r_{50} - r_{79}. In sphere 1 (the top sphere), $M1$ and $M2$ coincide from r_0 until r_{18}, after that, $M1$ is better. Graphs from 26 to 36 indicate the accuracy of the radii, being the horizontal axis the radius value and the vertical axis the distance measure F. Note that spheres obtained with $M2$ provides smaller spheres that $M1$ (Figure 16).

In Human2, there is an additional sphere, and the approximation is better than in Human3 since the mesh of the Human2 has a rounding shape. The top sphere (sphere 2) covers a region with a hole, so that there are no points collided with some rays of the disk. In the bottom sphere (sphere 0), $M2$ has a better approximation in r_{17}-r_{30}; in the other colliding points, the approximation is the same $M1 = M2$ (Figure 27). $M2$ is better in sphere 1, except in r_0-r_6 and r_{72}-r_{79}. In Sphere 2, $M1$ has a better approximation in r_{17}-r_{43}; in the other points, $M2$ is better. $M2$ is better in the middle of the interval of sphere 1, while $M1$ is better in the middle of the interval of sphere 2. Spheres of $M2$ are smaller than spheres of $M1$ (Figure 17).

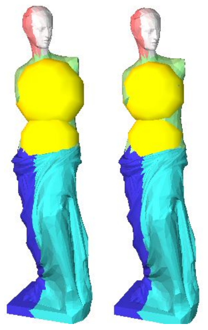

(a) $M1$ (b) $M2$

Figure 16: Approximation of spheres 0 and 1 in Human3, using 1 sphere per level.

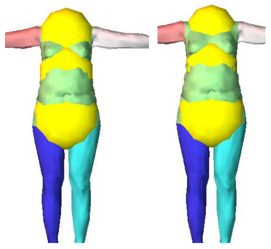

(a) $M1$ (b) $M2$

Figure 17: Approximation of spheres 0, 1, and 2 in Human2, using 1 sphere per level.

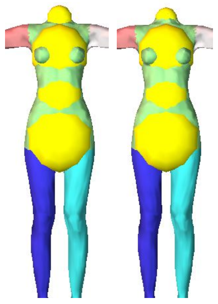

(a) $M1$ (b) $M2$

Figure 18: Approximation of spheres 0, 1, and 2 in Human1, using 1 sphere per level.

In Human1 we have an additional sphere in the top of the torso (sphere 3). We have a good approximation in this mesh for both methods $M1$, and $M2$. Like Human2, the top sphere does not contain the 80 colliding points, it has 66 colliding points. In sphere 0, $M1 = M2$, except in r_{21}-r_{60}, where $M2$ is better, however the approximation is almost the same for both methods. Spheres 1 and 2 have the same approximation. In sphere 3, $M2$ is better in $r_0 - r_{26}$ and in $r_{43} - r_{66}$, while $M1$ is better in $r_{27} - r_{42}$. In this last sphere the difference was significant (Figure 28). This occurs because the colliding points in the surrounding region has a circular shape, therefore $M2$ has a better approximation (Figure 18).

Until now we have just employed methods $M1$ and $M2$ to create spheres in the objects to pursuit the best approximation. However, we have not used the adjustment sphere process since this is only valid for more than two spheres per level. We are going to analyse the approximation of spheres with 2 and 4 partitions.

Human3 contains 3 levels, 6 spheres. Each partition has 40 points. The adjustment process takes the spheres created by $M1$ as the input and then, it applies some geometric transformations to obtain $M1$-A. As $M2$ returns a minimum sphere, there are not space to apply geometric transformations and thus we will have $M2 = M2$-A after the adjustment process. Observe that Human3 has an arm adjacent to the torso, this affects the torso shape and threfore the sphere approximation. Sphere 2 involves this region. Comparison is performed between $M1$-A and $M2$. In sphere 0, $M2$ is better at the start $(r_0 - r_1 9)$ then $M1$-A approximates better in $r_{20} - r_{36}$, and finally $M2$ is better in the last 4 colliding points.

In spheres 1 and 2, $M2 = M1$-A until r_{10}; after that, $M2$ is better (Figure 29). In sphere 2, $M1 = M1$-A and they are better than $M2$ in the first 9 rays. Then $M2$ is better for a minimum difference. In sphere 3, $M1$-A $(r_{10} - r_{34})$ is better than $M2$. In sphere 4, $M2$ is better, except in rays $r_{26} - r_{33}$, where $M1$-$A = M2$. In sphere 5, $M2$ is better until r_{16}, then $M1$ is better. Spheres of $M2$ are smaller than sphere of $M1$-A (Figure 29). Definively, two spheres approximates better than one sphere as depicted in Figure 19. This is due to the torso has the shape of an ellipse which can be well approximated with two spheres.

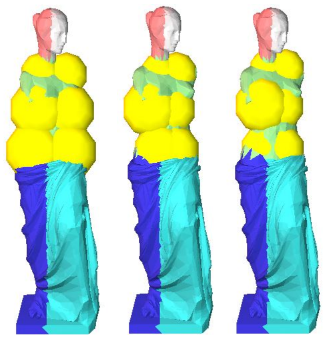

(a) $M1$ (b) $M2$ (c) $M1$-A

Figure 19: Approximation of spheres 0, 1, 2, 3, 4 and 5 in Human3, using 2 spheres per level.

Human2. In sphere 0, $M2$ is better until r_{30}, in sphere 1 $M2$ is better, even in $r_{11} - r_{16}$ the methods are equal (Figure 30). In sphere 2, $M1$-A is better until r_{20}, thereafter $M2$ is better. In sphere 3, $M2$ is better until r_{15}, then $M1$-A is better. The graphs of spheres 4 and 5 have the same behaviour and there was no variation between $M1$ and $M1$-A. In sphere 4, $M2$ is better until r_{17}, then they are equal until r_{21} and again $M2$ is better. In sphere 5, $M2$ is better until r_5 then $M1$-A is better until r_{13} and finally $M2$ is better. Like Human3, in the last two spheres, there are only 30 colliding points, as the existence of a hole in the top region of the mesh (Figure 20).

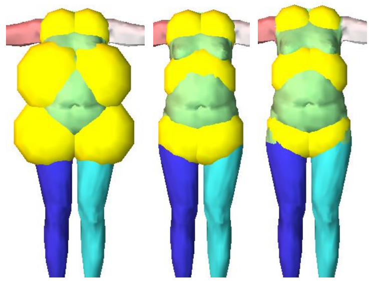

(a) $M1$ (b) $M2$ (c) $M1$-A

Figure 20: Approximation of spheres 0, 1, 2, 3, 4 and 5 in Human2, using 2 spheres per level.

Human1. There are two additional spheres, that is, one level more in comparison with Human3 and Human2. In sphere 0, $M2$ is better in the first half of the set of the colliding points (Figure 31). On the contrary, in sphere 1, $M1$-A is beter in the first half of the set of the colliding points. In spheres 2 and 3 the behaviour is the same as in spheres 0 and 1, respectively. This means that the torso shape remains equal. In sphere 4, $M2$ is better from r_8 and in sphere 5, $M2$ is better in r_0-r_{25}. In spheres 6 and 7, $M2 = M1$-A in the middle of the colliding points, and $M1$-A is better before r_{16} for sphere 6 and after r_{16} for sphere 7 (Figure 21).

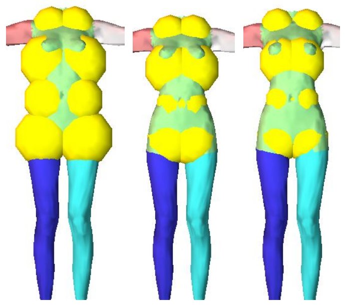

(a) $M1$ (b) $M2$ (c) $M1$-A

Figure 21: Approximation of spheres 0, 1, 2, 3, 4 and 5 in Human1, using 2 spheres per level.

From now on, we will consider 4 spheres per level. This increases the number of spheres, so that we proceed to analyse the case of Human1. In the first level (spheres 0 to 3) we observe better approximation of $M1$-A (Figures 32-36). In the second level (spheres 4 to 7) $M2$ is better, and $M1$ is almost equal $M1$-A. In level 2 $M2$ approximates better than $M1$-A, being $M1 = M1$-A in spheres 8 and 10. In level 3, $M2$ remains better, being $M1 = M1$-A in spheres 12 and 13. In the last level, $M1$-A starts gaining better aproximation than $M2$, except in the last sphere where $M2$ remains better from r_6 to the end (Figure 22). There are 5 levels, 20 spheres. The number of colliding points per partition is 20 (Figure 22).

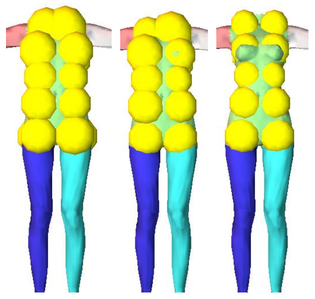

(a) $M1$ (b) $M2$ (c) $M1$-A

Figure 22: Approximation of the 20 spheres of Human1, using 4 spheres per level.

Note that some regions of the mesh are not covered as shown in Figure 23, where red points indicate the spaces left. To solve this drawback, we increase the penetration depth between spheres in the levels of the disk translation. In Figure 24, the overlap between spheres is varied: $30, 40, 50$ % using $M1$ in Human1.

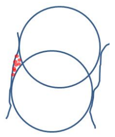
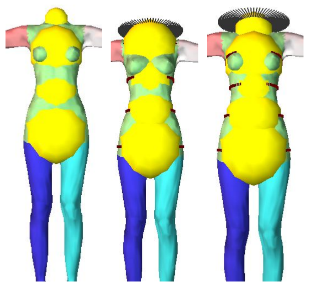


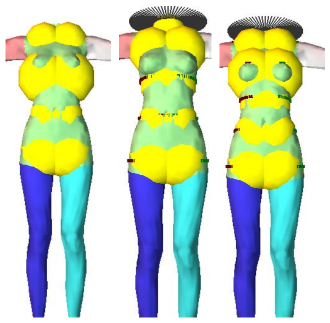
Figure 23: The space left by overlapping sphere is shown in red.

The value of the penetration depth depends on the shape of the mesh and in the size of the polygons.



(a) $M1, 30\%$ (b) $M1, 40\%$ (c) $M1, 50\%$

Figure 24: Overlapping spheres of 30%, 40%, 50% in Human1, employing $M1$ for 1 sphere per level.



(a) $M1$-$A, 30\%$ (b) $M1$-$A, 40\%$ (c) $M1$-$A, 50\%$

Figure 25: Overlapping spheres of 30%, 40%, 50% in Human1, employing $M1$-A for 2 spheres per level.

7 Conclusion

We present an algorithm for computing the approximation of a human torso using spheres. The main contribution of this paper is the methodology to improve the accuracy of the sphere approximation. We provide a set of experiments to proof the efficiency of the method and compare it with the smallest enclosing disk method proposed by Welzl. $M1$-A has a better approximation than $M2$ in regions with non circular shape, otherwise $M2$ has a better approximation. This work can be employed as the broad phase of the collision detection paper to cull away objets.

For further work we could make more experiments with different objects and other partitions, including boxes, OBB (Oriented Bounding Volume), or K-DOP (K-Discrete Oriented Polytope). Also, the error function F could be extended to other levels of the disk: $h + 1$, $h - 1$, $h + 2$, $h - 2$. The functions we employed in the algorithm involves several processes that can be parallelized. Some functions are the disk motion, the Ray Triangle intersectoin test, the adjustment process. In fact, the most expensive routine is the calculus of the surrounding region since it requires to determine the collision between the rays of the disk and the polygons of the mesh, giving a $O(N_r, n)$ time.

Finally, we expect to include this algorithm to join the spheres of the torso to the spheres of the upper and lower limbs, therefore apply a human animation and detect self collisions in a fast manner.

References

[1] Francisco A. Madera, Stephen D. Laycock, Carlos G. Herrera. *Ray-Triangle Collision detection to approximate Objects with Spheres*. Proceedings of the IASTED International Conference on Computer Graphics and Imaging, CGIM 2013. (2013), pp. 70–76.

[2] Rene Weller, Gabriel Zachmann. *ProtoSphere: A GPU-assisted Prototype Guided Sphere Packing Algorithm for Arbitrary Objects.* ACM SIGGRAPH ASIA 2010 Sketches, (2010), pp. 81–82.

[3] Kenji Shimada, David C. Gossard. *Bubble Mesh: Automated Triangular Meshing of Non-manifold Geometry by Sphere Packing.* Proceedings of the Third ACM Symposium on Solid Modeling and Applications SMA '95. (1995) pp. 409–419.

[4] Ladislav Kavan, Carol O'Sullivan, Jiří Žára. *Efficient Collision Detection for Spherical Blend Skinning.* Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia. (2006) pp. 147–156.

[5] Joachim Giesen, Balint Miklos, Mark Pauly. *The Medial Axis of the Union of Inner Voronoi Balls in the Plane.* (2012) pp. 515–523.

[6] Matthias Müller, Nuttapong Chentanez. *Virtual Reality Interactions and Physical Simulations (VRIPhys).* (2011), pp. 93–91.

[7] . O. Aichholzer, F. Aurenhammer, T. Hackl, B. Kornberger, M. Peternell, H. Pottmann. *Approximating Boundary-Triangulated Objects with Balls.* (2007), pp. 13–133.

[8] Svetlana Stolpner, Paul Kry, Kaleem Siddiqi. *Medial Spheres for Shape Approximation.* IEEE Trans. Pattern Anal. Mach. Intell. (2012), vol. 34, No. 6, pp. 1234 – 1240.

[9] Liya Wang, amir Assadi, Edgar Spalding. *Tracing Branched Curvilinear Structures with a Novel Adaptive Local PCA Algorithm.* Book IPVC, CSREA Press, (2008), pp. 557–563.

[10] Hua li, Anthony J. Yezzi, Laurent Cohen. *3D Multi-branch Tubular Surface and Centerline Extraction with 4D Iterative Key Points.* Lecture Notes in Computer Science, MICCAI (2009), pp. 1042–1050.

[11] X Jia and R.A Williams. *A packing algorithm for particles of arbitrary shapes.* Powder Technology Journal. Vol. 120, No. 3, pp. 175–186 (2001).

[12] Alexander Schiftner, Mathias Höbinger, Johannes Wallner, Helmut Pottmann. *Packing Circles and Spheres on Surfaces.* ACM Trans. Graph. vol. 28, No. 5, (2009), pp. 1–8.

[13] Jean-François Jerier, Didier Imbault, Frederic-Victor Donze, Pierre Doremus. *A geometric algorithm based on tetrahedral meshes to generate a dense polydisperse sphere packing.* Granular Matter Journal, vol. 11, No. 1, (2009), pp. 43–52.

[14] G. Pianet, F. Bertrand, D. Vidal, B. Mallet. *Discrete element method-based models for the consolidation of particle packings in paper-coating applications.* Asia-Pacific Journal of Chemical Engineering Journal, vol. 6, No. 1, (2011), pp. 44–54.

[15] Jituo Li, Guodong Lu, Juntao Ye. *Automatic Skinning and Animation of Skeletal Models.* Vis. Comput. Journal, vol. 27, No. 6, pp. 585–594.

[16] Pengjie Wang, Rynson W.H. Lau, Zhigeng Pan, Jiang Wang, Haiyu Song. *An Eigen-based motion retrieval method for real-time animation.* Computers & Graphics journal, vol. 38, No. 0, pp. 255–267.

[17] D. James, D. Pai. *BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models.* ACM Transactions on Graphics (SIGGRAPH 2004), vol. 23, No. 3.

[18] *Make Human: An Open Source tool to create 3D characters.* http://www.makehuman.org.

[19] Marco Livesu, Riccardo Scatent. *Extracting curve-skeletons from digital shapes using occluding contours.* Vis. Comput. journal, vol. 29, No. 9, (2013), pp. 907–916.

[20] G. Mistelbauer, A. Morar, A. Varchola, R. Schernthaner, I. Baclija, A. Köchl, A. Kanitsar, S. Bruckner, E. Gröller, E. *Vessel Visualization Using Curvicircular Feature Aggregation.* Proceedings of the 15th Eurographics Conference on Visualization, (2013), pp. 231–240.

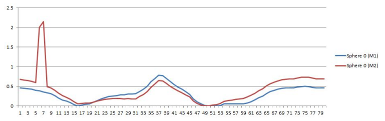
[21] Fuchang Liu, Y.J. Kim. *Exact and Adaptive Signed Distance FieldsComputation for Rigid and DeformableModels on GPUs*. IEEE Transactions on Visualization and Computer Graphics, vol. 20, No. 5c, (2014), pp. 714–725.

[22] Xinyu Zhang and Y.J. Kim. *Scalable Collision Detection Using p-Partition Fronts on Many-Core Processors*. IEEE Transactions on Visualization and Computer Graphics, vol. 20, No. 3, (2014), pp. 447–456.

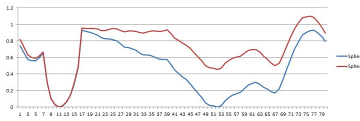
[23] John Shier, Paul Bourke. *An Algorithm for Random Fractal Filling of Space*. Computer Graphics Forum, vol. 32, No. 8, (2013), pp. 89–97.

[24] Vincent Luboz, Jim Kyaw-Tun, Sayan Sen, Roger Kneebone, Robert Dickinson, Richard Kitney, Fernando Bello. *Real-time stent and balloon simulation for stenosis treatment*. The Visual Computer, vol. 30, No. 3, (2014), pp. 341–349.

[25] Emo Welzl. *Smallest Enclosing Disks (balls and Ellipsoids*. Results and New Trends in Computer Science, Springer-Verlag, (1991), pp. 359–370.

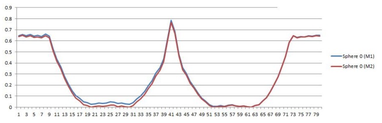


(a) Human3, sphere 0

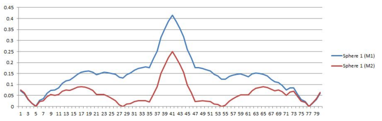


(b) Human3, sphere 1

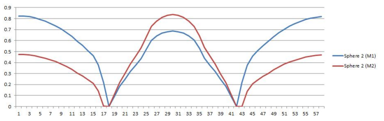
Figure 26: Approximation of spheres 0 and 1 of Human3 with 1 sphere per level, using $M1$ and $M2$.



(a) Human2, sphere 0

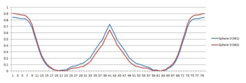


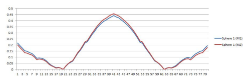
(b) Human2, sphere 1

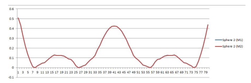


(c) Human2, sphere 2

Figure 27: Approximation of Human2 with 1 sphere per level, using $M1$ and $M2$.


(a) Human1, sphere 0


(b) Human1, sphere 1


(c) Human1, sphere 2

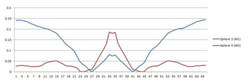
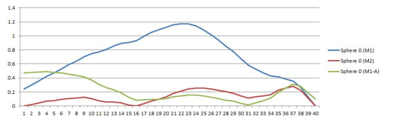
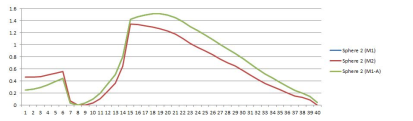

(d) Human1, sphere 3

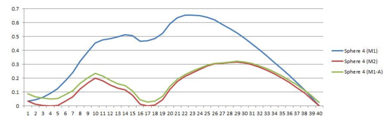
Figure 28: Approximation of Human1 with 1 sphere per level, using $M1$ and $M2$.



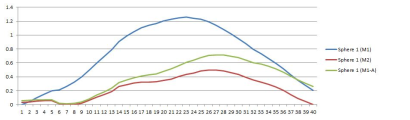
(a) Human3, sphere 0



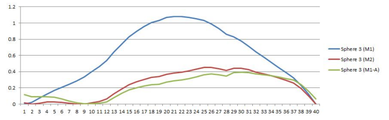
(b) Human3, sphere 1



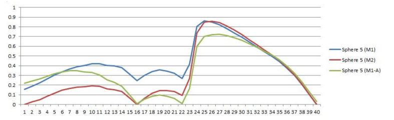
(c) Human3, sphere 2



(d) Human3, sphere 3

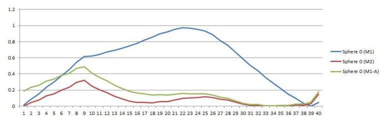


(e) Human3, sphere 4

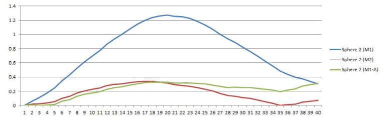


(f) Human3, sphere 6

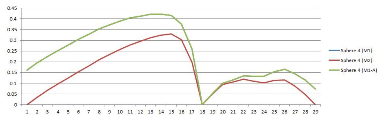
Figure 29: Approximation of Human3 with 2 spheres per level, using $M1$, $M2$ and $M1$-A.



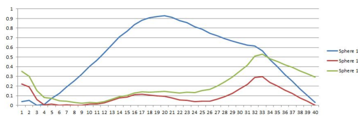
(a) Human2, sphere 0



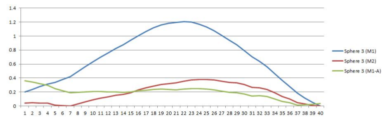
(b) Human2, sphere 1



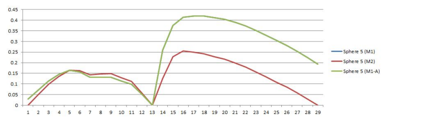
(c) Human2, sphere 2



(d) Human2, sphere 3

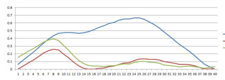


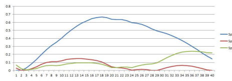
(e) Human2, sphere 4

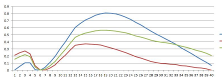


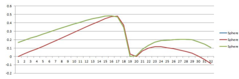
(f) Human2, sphere 5

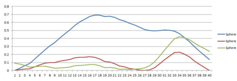
Figure 30: Approximation of Human2 with 2 spheres per level, using $M1$, $M2$ and $M1$-A.

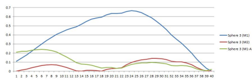

(a) Human1, sphere 0

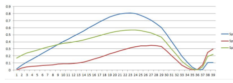

(b) Human1, sphere 1


(c) Human1, sphere 2


(d) Human1, sphere 3


(e) Human1, sphere 4


(f) Human1, sphere 5


(g) Human1, sphere 6

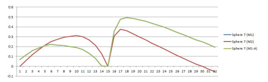
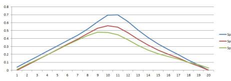
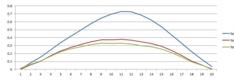
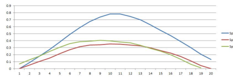

(h) Human1, sphere 7

Figure 31: Approximation of Human1 with 2 spheres per level, using $M1$, $M2$ and $M1$-A.


(a) Human1, sphere 0, level 0


(b) Human1, sphere 1, level 0


(c) Human1, sphere 2, level 0

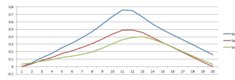
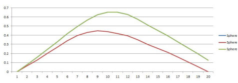
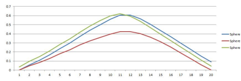
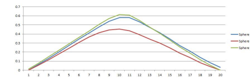

(d) Human1, sphere 3, level 0

Figure 32: Approximation of Human1 with 4 spheres per level, using $M1$, $M2$ and $M1$-A, level 0.


(a) Human1, sphere 0, level 1


(b) Human1, sphere 1, level 1


(c) Human1, sphere 2, level 1

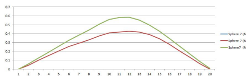
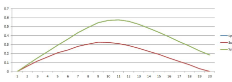
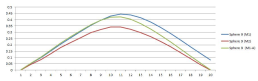
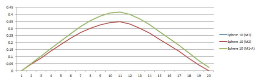

(d) Human1, sphere 3, level 1

Figure 33: Approximation of Human1 with 4 spheres per level, using $M1$, $M2$ and $M1$-A, level 1.


(a) Human1, sphere 0, level 2


(b) Human1, sphere 1, level 2


(c) Human1, sphere 2, level 2

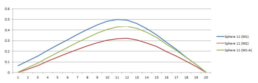
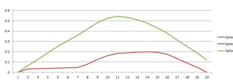
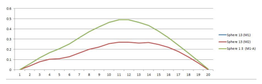
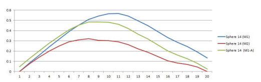

(d) Human1, sphere 3, level 2

Figure 34: Approximation of Human1 with 4 spheres per level, using $M1$, $M2$ and $M1$-A, level 2.


(a) Human1, sphere 0, level 3


(b) Human1, sphere 1, level 3


(c) Human1, sphere 2, level 3

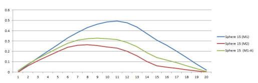
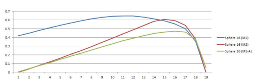
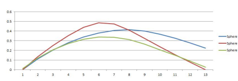
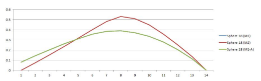

(d) Human1, sphere 2, level 3

Figure 35: Approximation of Human1 with 4 spheres per level, using $M1$, $M2$ and $M1$-A, level 3.


(a) Human1, sphere 0, level 4


(b) Human1, sphere 1, level 4


(c) Human1, sphere 2, level 4

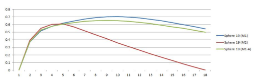

(d) Human1, sphere 3, level 4

Figure 36: Approximation of Human1 with 4 spheres per level, using $M1$, $M2$ and $M1$-A, level 4.