Scene Representation Technologies for 3DTV—A Survey

A. Aydın Alatan, *Member, IEEE*, Yücel Yemez, *Member, IEEE*, Uğur Güdükbay, *Senior Member, IEEE*, Xenophon Zabulis, Karsten Müller, *Senior Member, IEEE*, Çiğdem Eroğlu Erdem, *Member, IEEE*, Christian Weigel, and Aljoscha Smolic

(Invited Paper)

***Abstract*—3-D scene representation is utilized during scene extraction, modeling, transmission and display stages of a 3DTV framework. To this end, different representation technologies are proposed to fulfill the requirements of 3DTV paradigm. Dense point-based methods are appropriate for free-view 3DTV applications, since they can generate novel views easily. As surface representations, polygonal meshes are quite popular due to their generality and current hardware support. Unfortunately, there is no inherent smoothness in their description and the resulting renderings may contain unrealistic artifacts. NURBS surfaces have embedded smoothness and efficient tools for editing and animation, but they are more suitable for synthetic content. Smooth subdivision surfaces, which offer a good compromise between polygonal meshes and NURBS surfaces, require sophisticated geometry modeling tools and are usually difficult to obtain. One recent trend in surface representation is point-based modeling which can meet most of the requirements of 3DTV, however the relevant state-of-the-art is not yet mature enough. On the other hand, volumetric representations encapsulate neighborhood information that is useful for the reconstruction of surfaces with their parallel implementations for multiview stereo algorithms. Apart from the representation of 3-D structure by different primitives, texturing of scenes is also essential for a realistic scene rendering. Image-based rendering techniques directly render novel views of a scene from the acquired images, since they do not require any explicit geometry or texture representation. 3-D human face and body modeling facilitate the realistic animation and rendering of human figures that is quite crucial for 3DTV that might demand real-time animation of human bodies. Physically based modeling and animation techniques produce impressive results, thus have potential for use in a 3DTV framework for modeling and animating dynamic scenes. As a concluding remark, it can be argued that 3-D scene and texture representation techniques are mature enough to serve and fulfill the requirements of 3-D extraction, transmission and display sides in a 3DTV scenario.**

***Index Terms*—Animation, dense depth map, modeling, MPEG-4, nonuniform rational B-spline (NURBS), octree, point-based modeling, polygonal mesh, pseudo-3D, rendering, scene representation, subdivision surfaces, texture, volumetric representation, VRML, X3D, 3DTV.**

Manuscript received March 10, 2007; revised June 1, 2007. This work was supported in part by the European Commission Sixth Framework Program under Grant 511568 (3DTV Network of Excellence Project). This paper was recommended by Guest Editor L. Onural.

A. A. Alatan is with Department of Electrical Engineering, M.E.T.U., 06531 Ankara, Turkey (e-mail: alatan@eee.metu.edu.tr).

Y. Yemez is with Department of Computer Engineering, Koç University, 34450 Istanbul, Turkey (e-mail: yyemez@ku.edu.tr).

U. Güdükbay is with Department of Computer Engineering, Bilkent University, Bilkent 06800, Turkey (e-mail: gudukbay@cs.bilkent.edu.tr).

X. Zabulis is with ITI-CERTH, Thessaloniki 57001, Greece (e-mail: xenophon@iti.gr).

K. Müller is with Fraunhofer Institute for Telecommunications-Heinrich-Hertz-Institut, 10587 Berlin, Germany (e-mail: kmueller@hhi.de).

Ç. E. Erdem is with Momentum A.Ş., TÜBİTAK-MAM-TEKSEB, 41470 Kocaeli, Turkey (e-mail: cigdem.erdem@momentum-dmt.com).

C. Weigel is with Institute of Media Technology, Technical University of Ilmenau, 98684 Ilmenau, Germany (e-mail: Christian.weigel@tu-ilmenau.de).

A. Smolic is with Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut, 10587 Berlin, Germany (e-mail: smolic@hhi.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2007.909974

I. Introduction

A 3DTV is an end-to-end system for broadcasting 3-D scene information to consumer displays that are capable of providing 3-D perception to viewers. The content input to a 3DTV system may be synthetic (computer-generated) or captured from real scenes, and can be provided in various ways and forms depending on the type of the scene to be transmitted, the desired level of realism, the type of the specific application and/or the available bandwidth of the transmission channel. In this regard, 3-D scene representation is the bridging technology between content generation, transmission and display stages of a 3DTV system. The requirements of each of these stages for scene representation are often very different one from another; even conflicting in some cases (e.g., rate versus quality) and the employed methods to meet these requirements are quite diverse. Hence, an effective 3DTV system will eventually need to support a large variety of representation techniques existing in the literature, from the most simplistic image-based techniques to the sophisticated geometry modeling based approaches adopted from computer graphics.

In this paper, we provide a comprehensive survey of existing techniques and approaches that could be used in a fully functional 3DTV system for scene description, considering the specific requirements that a scene representation methodology needs to fulfill. These requirements include *generality, accuracy, perceptual quality, level of detail scalability, progressivity, compression, editing, animation* and *compatibility*.

Generality refers to the ability of a representation to deal with arbitrary topology and geometry. This requirement is essential for 3DTV, since scanned real objects can indeed be of complex shapes. *Accuracy and perceptual quality* are two other key properties, especially for applications where realism is the main concern. Ideally, a representation would have controllable

smoothness, i.e., the capability to both be smooth and represent fine detail at the same time. *Level of detail (LoD) scalability* addresses the ability to produce quality-reduced or simplified versions of a 3-D model once its complete description is available. LoD representations enable less powerful rendering engines to render the object at a reduced quality; they are also useful in the editing and animation of detailed 3-D models, as they provide coarse manipulation semantics. *Progressivity* refers to the problem of progressive transmission and visualization of highly detailed 3-D models. Note that LoD scalability does not necessarily imply progressivity, which requires an incremental LoD representation. Progressive modeling schemes enable a decoder or a viewer to construct a 3-D model from a partial bit stream. *Compression* addresses space-efficient storage of 3-D models; this also implies efficient transmission during 3DTV broadcast. Although compression is conceptually more related to statistical decorrelation of data, the intrinsic structural compactness of a representation is also an important issue. *Editing* addresses issues, such as deformability, ease of manipulation and the capability to model time-varying geometry; these issues are all important for virtual reality applications, such as interactive 3DTV applications and computer animation. Finally, *compatibility* refers to the availability of hardware support to render a specific representation.

The organization of the paper is as follows. In Sections II–IV, we address three ways of representing the geometry of a 3-D scene: *dense depth*, *surface-based* and *volumetric* representations. Much of the discussion is devoted to surface-based representations; the field of computer graphics has a vast literature in this area covering a variety of powerful approaches, such as *polygon meshes, nonuniform rational B-spline (NURBS), subdivision surfaces,* and *point sets*. Section V deals with *texture representation* schemes; a 3-D scene is characterized not only by its geometry but also by the texture of the objects that it contains. Alternatively, 3-D views of a scene can be composed directly from its images without using any explicit geometry and texture representation; these so-called *pseudo-3D* techniques are discussed in Section VI. Section VII addresses the basic tasks involved in object-based dynamic 3-D scene modeling: *representation, animation* and *rendering*. Section VIII addresses head and body representations as scenes involving humans are common in 3DTV applications. Section IX deals with recent standardization activities aimed at achieving interoperability between different 3-D scene representation technologies. Concluding remarks are finally provided in Section X.

II. Dense Depth Representations

The fundamental representation of a sole point in 3-D space could be obtained by a vector of three dimensions (or four dimensions in *homogeneous coordinates*). However, 3DTV applications usually do not require a representation for such a sole point in space. In a typical *free-viewpoint TV* scenario, the users freely select their viewing angles by generating the desired virtual views from the delivered multiview video. Thus, the camera distances (depth) of the scene points, whose projections give the pixel locations on the image, are essential to render an arbitrary view of the scene. Therefore, it is better to examine, not a single point, but a regular dense-depth representation of a scene. The distances of the points in a 3-D scene from the camera are stored in a lattice, defined by the reference image of the scene and denoted, as a *depth map*.

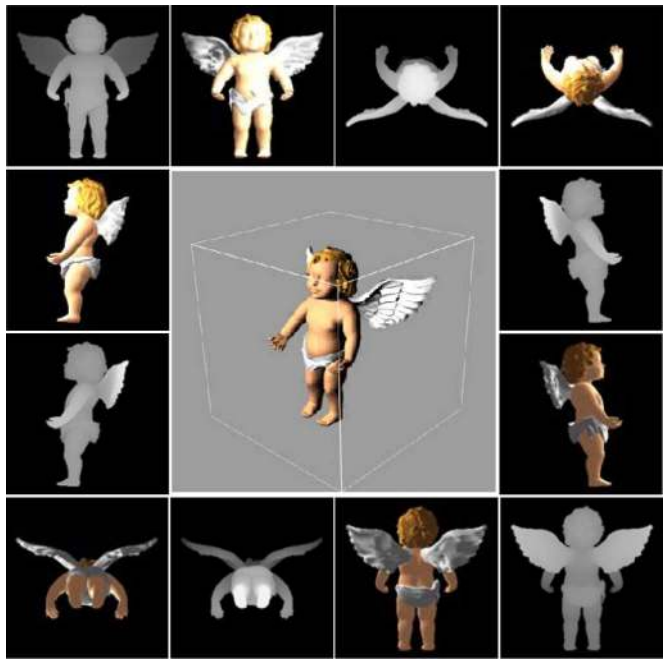

Fig. 1. Example of AFX DIBR [2]. Copyright © 2005, IEEE, Inc. Reprinted by permission.

A. Dense Depth Representations

In 1998, Shade *et al.* proposed the concept of layered depth images (LDI) [1]. In this approach, a 3-D object (or a scene) is represented by a number of views with associated depth maps, as shown in Fig. 1 [2].

Using appropriate scaling and information from camera calibration, it is possible to render virtual intermediate views, as illustrated in the center image of Fig. 1. The quality of the rendered views and the possible range of navigation depend on the number of original views and camera settings. In case of simple camera configurations (such as a conventional stereo-rig or a multibaseline video system), LDI can even be utilized for fully automatic real-time depth reconstruction in 3-D video or 3DTV applications, which could be denoted as depth image-based rendering [3], [4].

A data and rendering format for LDI is included in the recent computer graphics extension of MPEG-4. It is called as Animation Framework eXtension (AFX) [5] and makes it easy to use LDI in a standardized way [6]. Thus, LDI or depth image-based rendering method represents an efficient and attractive alternative to classical 3-D mesh representations of 3-D scenes and objects. Since LDI represents a highly attractive representation format for 3-D scenes and objects, the 3 DAV group of MPEG investigates LDI as a standard format for 3DTV applications [7].

Apart from the problem of reliable dense-depth extraction, another serious problem often arises at depth discontinuities, such as object boundaries. These boundary effects can be reduced by a technique, called *alpha-matting*, where over-blending of depth values is used over object boundaries.

Fig. 2. Point-based representation of a dense depth map extracted from multiview video.

Once the discontinuities have been identified over the whole image, alpha-matting technique can be applied over a certain region; this can significantly improve the rendered output [8].

B. Rate-Distortion Optimal Dense Depth Representations

The representation of a 3-D scene by dense depth map(s) will face a bandwidth problem, since in any 3DTV system, this redundant information is usually delivered over a capacity-limited channel. Hence, this information should be optimally represented and compressed by minimizing both its rate and distortion together. The conventional strategies encode the available depth field by lossy image or video compression methods [6]; there is also a novel approach for extracting a depth field, whose representation is optimal in the rate-distortion sense [9]. In other words, the depth field is extracted in such a way that the resulting dense representation is easier to compress, yielding minimum distortion compared to the ground-truth depth field.

In the literature, the most popular methods for obtaining a dense depth field are approaches based on Markov random field (MRF) [9], [10] and partial differential equations (PDE) [11], [12]. Although these approaches derive from two different hypotheses, they end up with similar formulations, in which a cost function is minimized to arrive at the unknown dense depth field. A typical cost function consists of two terms; one favors intensity similarity for the desired depth values of different views, whereas the other implies smoothness between neighboring depth values. These two terms also approximate the depth distortion and number of bits to encode the resulting depth [9]. Therefore, minimizing such cost functions also yields rate-distortion optimal dense depth representations, addressing the requirements stated in Section I for accuracy and perceptual quality, and for compression. A typical example is shown in Fig. 2.

The multiview dense depth maps can efficiently produce 3-D replica of real scenes. They represent the whole scene with a single surface, making no distinction between separate objects; hence, they are easy to construct and space-efficient but incapable of modeling the scene semantics. Graphical realism, progressive modeling, level of detail scalability and animation are fundamental functionalities which are hard to achieve using dense depth representations.

III. Surface-Based Representations

In this section, we provide a comparative survey of various techniques and approaches which constitute the state-of-the-art in computer graphics for representing 3-D surface geometry. These representation techniques can address most of the 3DTV requirements stated in Section I, including graphical realism, progressive modeling, level of detail scalability, and animation. We discuss four different surface representation paradigms: *polygonal meshes, NURBS, subdivision surfaces*, and *point-based modeling*.

A. Polygonal Meshes

Polygonal meshes are currently the most common 3-D representations in the manufacturing, architectural and entertainment industries. Polygons are also the basic primitives of hardware rendering technologies. The increasing demand for realism in computer graphics and the developments in 3-D scanning technologies result in more complicated object meshes, containing millions of polygons; these can satisfactorily represent any geometric surface detail with almost no topological restrictions. Such complex meshes are very expensive to store, transmit and render; this has led to the development of many mesh simplification and compression techniques resulting in flexible representations with different levels of detail and progressive quality.

Progressive Meshes: Most of the state-of-the-art mesh representation techniques are based upon the progressive meshes (PM) scheme [13]. In the PM scheme, an arbitrary triangular mesh can be stored as a coarser mesh along with a sequence of mesh refinement operations referred to as *vertex splits*. A vertex split is a local elementary mesh transformation that adds a single vertex to the mesh. The PM representation of a surface defines a *continuous sequence of meshes* with increasing accuracy. Each mesh of the sequence corresponds to a LoD approximation specified by a single vertex split operation.

The PM scheme naturally supports progressive transmission. However, as in all polygon-based representations, there is no inherent smoothness embedded in its description and polygonal artifacts appear along the silhouette boundaries, especially in the case of zooming and/or low resolution representations. One partial remedy to this problem, as proposed by Hoppe in [14], is to incorporate a view-dependent rendering and transmission strategy that can selectively refine a progressive mesh along its silhouette boundaries for a given view angle by making use of the locality of vertex split operations.

Progressive Forest Split Compression: The PM representation is not well suited for compression; the cost of a vertex split depends heavily on the size of the initial mesh, making the PM scheme impractical for very large meshes. The progressive forest split (PFS) [15] and the compressed progressive meshes (CPM) [16] are basically space-efficient versions of the PM scheme. In both methods, the vertex splits are grouped into

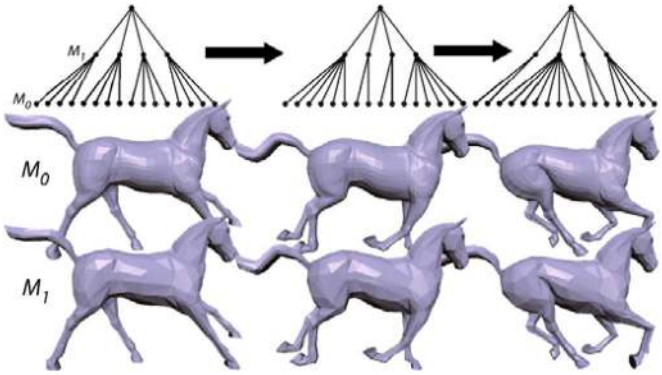

Fig. 3 Progressive time-varying meshes [22]. The 3-D Horse sequence at two levels of detail. Copyright © 2005, ACM, Inc. Reprinted by permission.

batches. In this way, the granularity of the progressive representation is limited while the storage cost per triangle for encoding connectivity changes becomes independent of the initial mesh size.

3DMC coding of MPEG–4 v.2 [17] is mainly based on the PFS scheme, which encodes an arbitrary mesh, as a coarse mesh along with forest splits. The coarse mesh is compressed using topological surgery [18], which was also included in the VRML compressed binary format [19]. In the PFS scheme, the atomic refinement record is a group of vertex splits, that is, a forest split. This causes a compromise between compression and LoD granularity. In this way, the totality of vertex split operations is encoded at a much lower cost, but the resulting number of levels of detail is limited and the geometry cannot be locally refined. A direct consequence of this limitation is that a flexible view-dependent based rendering or transmission scheme cannot be implemented with 3DMC. We should finally note that both PM and PFS schemes have been extended to handle nonmanifold triangulations [20], [21].

Progressive Time-Varying Meshes: When modeling time-varying or deforming surfaces with meshes, it is much more space efficient to use a fixed connectivity for all frames of the animation and to modify only the vertex positions, rather than using a separate mesh for each time instant. However, the use of static connectivity often yields inadequate modeling of a deformable surface. Very recently, a progressive scheme has been proposed, which can efficiently produce incremental LoD approximations for all frames of a time-varying surface [22]. The scheme uses edge splits (or contractions) to refine (or simplify) the geometry of a given mesh. The edge contractions are clustered according to a base hierarchy that produces LoD approximations for the initial frame. The base hierarchy is then incrementally adapted to the geometry of the subsequent frames by using edge swap operations (see Fig. 3). The whole deforming surface can thus be encoded in terms of the initial vertex positions and the base hierarchy along with the swap sequence and the vertex displacements for each frame.

B. NURBS

Many shapes can be described by NURBS surfaces without loss of mathematical exactness. Since they are smooth and easily manipulated, the NURBS representation has long been a common choice in computer aided design (CAD) or manufacturing (CAM) and computer animation. A NURBS surface patch is represented by a function of two parameters, u and v, which defines a mapping of a 2-D region into the 3-D Euclidean space. It is usually expressed as the tensor product of some piecewise-polynomial basis functions (B-splines) and specified by a mesh of 3-D control points and two knot vectors (specifying the domain) over which the B-spline basis functions are defined. A point on a NURBS surface S is given by

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{ip}(u)N_{jq}(v)w_{ij}\mathbf{B}_{ij}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{ip}(u)N_{jq}(v)w_{ij}}, 0 \le u,v \le 1 \tag{1}$$

where $\{N_{ir}\}$ denotes the B-spline basis functions; p, q: degrees (order) of the surface in u and v directions; $\mathbf{B}_{ij}$: a mesh of $n \times m$ control points; w_{ij}: weights. The knot sequences, U and V, are two nondecreasing sets of real numbers (knots), and partition the parameterization domain into subintervals: $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_m\}$. The B-spline basis functions $N_{ir}(u)$ are defined over these knot sequences and can be calculated in a recursive manner. Each knot of a knot sequence is associated to a control point and to a basis function calculated as above.

One of the key characteristics of a NURBS surface is that its shape is primarily determined by the positions of its control points, and hence, the influence of each control point is local. This property is very desirable because it allows the operator to make localized changes by moving only individual control points, without affecting the overall shape of the surface. The shape of a NURBS surface and its smoothness can be controlled by the choice of knot vectors. In the most general case, the knot vectors can be nonuniform in the sense that the interval between two consecutive knots can vary inside a knot vector, yielding a *nonuniform* representation. The effect of a given control point might also be different relative to another, depending on its weight; this is why NURBS is *rational*. In this respect, tensor-product *uniform B-spline surfaces* can be seen, as a special case of the general class of NURBS representation with uniform knot vectors and equally weighted control points. A comprehensive mathematical description of NURBS can be found in [23]; another article [24] provides an intuitive understanding of the functionality of NURBS (in particular curves) in practice.

Topological Limitations: The NURBS representation, as a tensor product surface, can represent only planar, cylindrical or toroidal topologies. In order to overcome this restriction in practice, a surface of arbitrary topological type is modeled as a network of NURBS or B-spline patches that are stitched together. One of the challenges in NURBS modeling is to define a patchwork structure and then merge the resulting patches seamlessly. A common tool to create NURBS models of arbitrary topology is NURBS trimming [25]; this is the most difficult and unreliable part of NURBS modeling, since seamless stitching of patches requires much labor and human intervention.

Surface Fitting: Manual, semi-automated, or automated techniques can be used to fit NURBS surfaces to scanned 3-D objects of arbitrary topology. *Automated* techniques use constrained optimization to construct the surface patchwork and the parameterization. However, the automated techniques

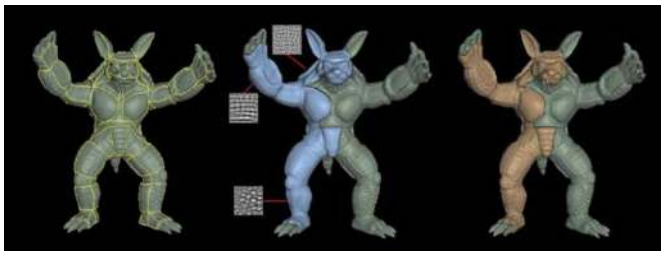

Fig. 4. Uniform B-spline surface fitting [29]. Original polygonal mesh painted with patch boundaries, shaded B-spline surface patches (right half of the figure is the original mesh), and displacement mapped B-spline patches. Copyright © 1996, ACM, Inc. Reprinted by permission.

in the literature generally either have severe topological restrictions [26], [27] or suffer from parameterization distortion and computational complexity [28]. *Semi-automated* schemes can partially address these drawbacks. Such a scheme is proposed in [29], which allows human interaction in both patch placement and B-spline patch fitting. In this scheme, the user roughly selects the outline of the rectangular patch boundaries by picking up successive vertices of an initial dense triangulation. These boundaries are automatically optimized, smoothed and represented in terms of B-spline curves. A uniform grid of 3-D points is then sampled over each patch and a B-spline surface is fit via unconstrained optimization by using a coarse to fine framework, as demonstrated in Fig. 4.

Smoothness: A NURBS surface has controllable smoothness within a patch; in other words, the C^n continuity (i.e., n-times differentiability at every surface point) can be controlled by the degree of the basis B-spline functions and knot multiplicities. However, obtaining smoothness along patch boundaries is not straightforward. One solution is to define a built-in G^1 (tangent-plane) continuity at patch boundaries as in [28], which is visually sufficient for a seamless patchwork structure. This is achievable only if uniform B-splines are employed, or if all patches are forced to have the same knot vector and the same order; but these restrictions limit the ability of the general NURBS representation to deal with fine surface details.

Representing Fine Detail: One of the major drawbacks of NURBS modeling is its inefficiency in representing fine surface details. Local refinement of a NURBS surface necessitates large-scale modification. In order to add a single control point within a patch, an entire column or row of control points must be split to preserve the desired quadrilateral grid structure. This situation may even produce a more global effect, which propagates into the whole patchwork structure if, for example, there is a built-in continuity setting. One solution is to make use of displacement maps as proposed in [29]; these model and store the fine detail as if it were a kind of texture information, and then map it onto the surface during rendering. The schemes based on displacement maps are also useful to separate fine detail from coarser semantics for animation and editing purposes.

Another possibility for modeling fine detail is to use hierarchical B-splines [26]. However the schemes based on hierarchical B-splines are not sufficiently generalized to work with arbitrary complexity and topology; they seem to be more efficient in editing and refining computer generated models for which they can provide valuable coarse manipulation semantics.

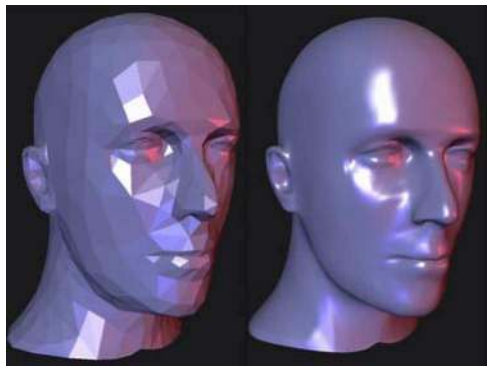

Fig. 5. Interpolating subdivision [33]. The initial semi-regular coarse mesh and the smooth limit surface obtained by modified Butterfly subdivision. Copyright © 1996, ACM, Inc. Reprinted by permission.

Level of Detail: Theoretically, a NURBS surface has infinite resolution. In practice however, it is tessellated into either a triangular or quadrilateral representation before rendering. This is achieved by stepping through the u- and v-domains and evaluating the NURBS equation for points on the surface. Such an evaluation produces a grid of sample points at a desired level of detail; these can then be easily converted into a mesh representation. The built-in LoD control of NURBS surfaces does not however imply a real incremental LoD representation. A NURBS surface is a compact representation and its LoD hierarchy can be constructed only if the complete representation is available.

C. Subdivision Surfaces

Subdivision surfaces have gained attention in the last decade, as a possible alternative to NURBS and traditional polygonal representations. The main challenge is to unify these two extremes of 3-D modeling in an infrastructure that allows representation of arbitrary topology and any fine detail with a more controllable smoothness. In subdivision schemes, the basic idea is to construct a surface from an arbitrary polygonal mesh by recursively subdividing each face. If the subdivision is done appropriately, the limit of this sequence of successive subdivision will be a smooth surface. For example, the well-known Catmull–Clark [30] subdivision scheme yields a bicubic B-spline as the limiting surface.

Subdivision schemes in the literature can be classified on the following three criteria: the pattern of the refinement rule (vertex insertion [30]–[33] or corner cutting [34]), the type of generated mesh (triangular [31], [33] or quadrilateral [30], [32]), whether the scheme is approximating [30], [31] or interpolating [32], [33]. One of the simplest subdivision schemes is the Loop scheme for triangular meshes, which uses vertex insertion for refinement [31]. The refinement proceeds by splitting each triangular face into four subfaces. The vertices of the refined mesh are then repositioned by using weighted averages of the vertices in the initial mesh. Vertex insertion schemes can be interpolating or approximating; in the first approach, the original control points, i.e., the mesh vertices, are also points of the limit surface. Interpolating schemes are attractive, since they allow control of the limit surface in a more intuitive way (see Fig. 5). On the other hand, the quality of surfaces produced by approximating is higher and they converge to the limit surface faster.

Subdivision schemes give at least C^1 smoothness, even in irregular settings [35] and the smooth limit surface can be computed explicitly without need for infinite subdivision recursion [36]. To deal with irregular meshes, semi-uniform subdivision schemes make a distinction between regular and irregular (*extraordinary*) vertices. Extraordinary vertices of a semi-regular mesh (i.e., a mesh made up of mostly regular vertices) are those with valence other than 6 for the triangular case and 4 for the quadrilateral case. In order to guarantee C^1 smoothness, extraordinary vertices are treated differently and the subdivision coefficients at these vertices vary depending on their valences [33].

Multiresolution Analysis-Synthesis: With subdivision schemes, it is possible to build a multiresolution mesh pyramid that allows one to coarsen a given mesh, and later refine it as desired, in order to always recover the same mesh with the same connectivity, geometry and parameterization. By reversing the subdivision process, i.e., by repeatedly smoothing and downsampling, an irregular coarse base mesh can be obtained from an initial dense mesh along with its smooth intermediate representations.

The geometrical information which is lost due to smoothing can be incorporated into the subdivision process by encoding the difference as *detail offsets* [37], [38]. Once this is achieved, the initial mesh can be recovered from the coarse mesh by repeated subdivisions and by adding the detail offsets. This process is referred to as *multiresolution analysis* and *synthesis*. The recorded details can be introduced at any time independent of the others and propagated smoothly, as the surface is refined or coarsened.

Multiresolution analysis also allows the construction of an efficient progressive representation which encodes the original mesh with a coarse mesh and a sequence of wavelet coefficients expressing the detail offsets between successive levels. There are several methods for building wavelets on semi-regular meshes [39], [40]. These schemes are particularly effective for compression of densely sampled, highly detailed surfaces. However, when the input geometry to be compressed is already well-described by a compact simplified mesh, the space efficiency of subdivision schemes becomes questionable.

Subdivision Connectivity and Remeshing: A mesh, generated from an initial mesh of arbitrary connectivity by successive subdivision, is said to have *subdivision connectivity*. Such meshes are semi-regular meshes, made up of mostly regular vertices except for some isolated extraordinary vertices. Most of the techniques based on subdivision surfaces are applicable only to the meshes having subdivision connectivity, such as the multiresolution analysis-synthesis scheme described in the previous subsection. However, mesh representations, especially those generated from scanned 3-D data, do not have this property. The literature has various remeshing techniques [41]–[43], which can convert an arbitrary mesh into a form with subdivision connectivity. This necessitates finding a new parameterization of the underlying surface, represented by the initial arbitrary mesh over a much coarser simplified version (base domain). Hence, by remeshing, the connectivity but not the geometry of the mesh is modified. Remeshing is a computationally costly task, and in practice, cannot be achieved without introducing some distortion to the geometry of the original mesh.

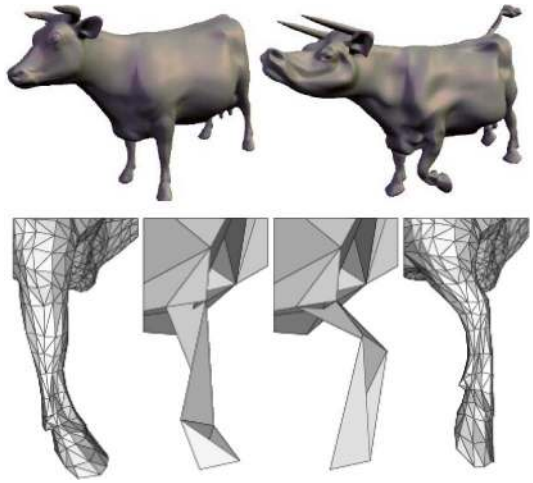

Fig. 6 Multiresolution mesh editing [37]. (Above) Original cow and its edited version. (Below) Editing sequence for the leg: original, coarsest scale, edit, and reconstruction with multiresolution details. Copyright © 1997, ACM, Inc. Reprinted by permission.

Multiresolution Editing: Once a multiresolution mesh pyramid has been constructed for a subdivision surface, it can be edited at any available resolution level in the same manner that the surface details are introduced [37] (see Fig. 6). The changes (e.g., by clicking on a vertex and dragging with the mouse) introduced on a coarser level propagate smoothly towards finer levels, or vice versa. This provides valuable manipulation semantics that can be used in animation, deformation or surface design. Subdivision surfaces are already being used in commercial applications, especially for animation purposes. A famous example is the animated short film *Geri's Game*, by *Pixar* [44]; won Oscar for the best animated short film in 1997.

Nonuniform Subdivision: The subdivision connectivity property, a requirement for most multiresolution techniques based on subdivision, is an important limitation. The pioneering work presented in [45] combines a variety of the techniques in the literature and extends the multiresolution analysis-synthesis scheme to irregular meshes with no need for remeshing; thus multiresolution signal processing techniques, such as editing, smoothing, enhancement and texture mapping, also become applicable in the irregular setting. This generality is basically achieved by defining a nonuniform subdivision scheme, where the weighting coefficients depend not only on the connectivity, but also on the geometry. In practice, these techniques work quite well, however, their analytical smoothness is not completely known at present.

D. Point-Based Modeling

Surface points [46], [47], particles [48], or surfels [49] can be used instead of triangles (or polygons), as simpler display primitives for surface representation. The terms "particle" or "surfel" are used in the literature to denote a dimensionless space point that has context-dependent surface properties, such as surface normal or color. In point-based schemes, the surface geometry is represented by a set of points sampled from the surface; no topology or connectivity information is explicitly stored. Hence,

point sets do not have a fixed continuity class, nor are they limited to a certain topology, as are many other surface representations; therefore they represent any shape with arbitrary topology and complexity. In the most general setting, the distribution of sampled points is nonuniform; however the point set can easily be forced to be sampled on a discrete grid. Uniformly sampled point sets are much easier to handle, compress and render, since they implicitly impose some neighborhood information.

The idea of using points, instead of triangle meshes and textures, was first proposed by Levoy *et al.* [46]. For more than a decade, point-based modeling was used mostly to model phenomena that are difficult to model with polygons, such as smoke, fire and water. With the recent advances in scanning technologies, the ability of 3-D models to represent real objects has increased tremendously but processing such huge meshes produces bottlenecks with current technology. As a result, point-based representation systems have regained attention in the last half-decade. Rendering complexity is one of the most problematic issues. When a highly detailed complex 3-D triangle model is rendered, the projected size of individual triangles is often smaller than the size of a pixel in the screen image. In this case, the polygon rasterization process at the rendering pipeline becomes unnecessarily costly, whereas rendering individual points, rather than polygons, can be much more efficient.

Octree Particles and Qsplat: A progressive point-based surface-modeling technique was first proposed by Yemez *et al.* [50], [51]. This technique is based upon a hierarchical octree structure, which first voxelizes the surface geometry at different levels of detail and then encodes it in terms of octree particles that are uniformly sampled over the surface. The particles are encoded in such an order that the viewer, or the decoder, can progressively reconstruct the surface information and visualize it by on-the-fly triangulation and polygon rendering. In two very closely related works, Rusinkiewicz *et al.* [52], [53] followed the framework presented in [50], and proposed a point-based technique to render their large data sets resulting from the *Digital Michelangelo Project* [54]. Rather than an octree representation, they constructed a sphere hierarchy and rendered the resulting representation via *splatting*. Splatting allowed them to implement a progressive interactive representation that can be locally refined with respect to viewing parameters.

Since then, several other progressive schemes have been proposed, all based on hierarchical data structures [51]–[53], [55]. When sampled on a regular grid, such as octree, the point sets can easily support progressive transmission/visualization and LoD scalability with no additional explicit refinement information. Very recently, the authors of [56] have proposed a progressive point-based scheme that is also applicable to nonuniformly sampled surface points in a framework that is very similar to subdivision surfaces and the PM scheme.

Compared to triangle meshes, point-based representations seem to be more efficient regarding storage and memory for progressive modeling of high resolution surface geometry; this is because only sample point coordinates need to be stored and they do not require additional structural information such as connectivity. For instance, the storage requirements for a uniformly distributed high resolution point dataset can be reduced by up to about 2 bits per sample by using hierarchical structures [55], [57]. However, low-resolution approximations of point sets do not generally produce realistic rendering results.

Fig. 7. Surface splatting examples [58]. Copyright © 2001, ACM, Inc. Reprinted by permission.

Splatting: Point rendering can be viewed as a resampling problem. When surface samples are projected from the object space onto the image space, the projections do not usually coincide with the regular grid of the output image. If this process is not properly handled, visual artifacts, due to aliasing and undersampling, might appear. Point rendering basically consists of reconstruction of a continuous surface from the input samples, filtering the continuous representation, and sampling it (or evaluating it) at the output positions.

Most point-based rendering systems use splatting to achieve high quality renderings [49], [52], [55], [58]–[61] (see Fig. 7). The basic idea in splatting is to associate each surface point with an oriented tangential disc. The shape and size of the disc may vary; if it is circular, it projects as an elliptical splat on the image plane and its radius can be adjusted with respect to the local density. The shade or color of the point is warped accordingly so that its intensity decays in the radial direction from the center. The shape of the splat together with its intensity distribution defines the reconstruction kernel, or so-called *footprint*. Often a single image pixel is influenced by several overlapping splats; in this case the shade of the pixel is computed by the intensity-weighted average of the splat colors. Proper filtering is needed at the end to avoid aliasing artifacts. The choice of the splat shape and distribution often yields a compromise between rendering quality and performance; Gaussian circular splats usually perform well. Other more sophisticated non-Gaussian kernels could also be used at the cost of an increase in rendering time [62]. Even without special hardware support, current splatting techniques can achieve very high quality rendering of point-based surface models at a speed of millions of points per second.

Point-Based Animation: Points (or particles) have been successfully used to animate complex phenomena, such as smoke, fire and water, using dynamic particle systems with inter-particle forces. This literature already offers a framework to those interested in point-based computer graphics for animation and editing. Currently, there are very few works and systems that address point-based animation [63] (see Fig. 8) and editing [64]. These systems are far from mature, but worthy of attention. For example, *Pointshop* 3-D is an interactive editing system [64], which is also available as a modular software platform and implementation test-bed for point-based graphics applications. It provides a set of kernel functions for loading, storing, modifying, and rendering point-sampled surfaces.

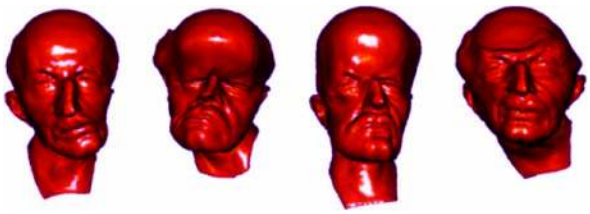

Fig. 8. Point-based animation with an elasticity model driven from continuum mechanics [63] Copyright © 2004, ACM, Inc. Reprinted by permission.

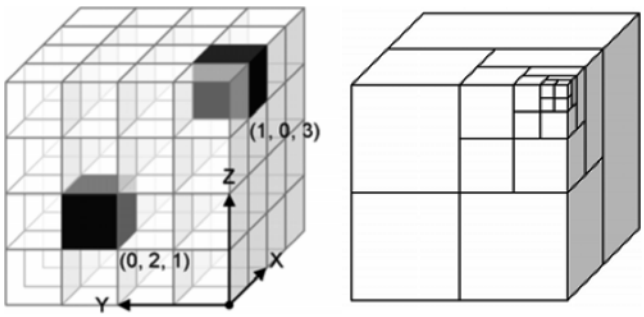

Fig. 9. Illustration of data structures implementing volumetric representations. (left): voxel buffer. (right): octree.

IV. Volumetric Representations

The notion of volumetric representation refers to parameterization of the reconstruction volume in a world reference frame. The representation may contain data associated to any location within a volume of interest. The unit volume, defined by a regular parameterization of the space, is called a *voxel* and corresponds to the smallest representable amount of space [65]; therefore, it determines the precision of the representation..

The data associated with a voxel refers to the properties of the surface segment that occurs within it, or else, occupies it. Empty voxels are typically not associated with any information other than a Boolean flag indicating that they are vacant. Thus, voxel representations might be quite memory-inefficient, since the utilized data usually take up only a small portion of the representation capacity. Straightforward implementation of a voxel space is through a volume buffer [1], or cuberille [66], employing a 3-D array of cubic cells. Other implementations utilize data structures that omit the representation of voxels that are empty, or cannot be observed.

A common implementation of the above idea is called an *octree* [67] (see Fig. 9). Starting from a coarse resolution, it facilitates the detection of large empty areas; the resolution is refined only for the areas that contain surfaces. The data structure consists of a tree that grows in depth, but only at the nodes that correspond to occupied voxels; it recursively subdivides the initial voxel into eight parts of equal size and terminates at a predefined resolution. In this process, a tree is generated that represents the occupied voxels. Octrees exhibit greater representational efficiency over volume buffers and they are the most common method of storing voxels, providing both good data compression and ease of implementation. Some variations of this approach are linear octrees [68], PM-octrees [69], kd-trees [70], and interval trees [71]. In addition, nested multiresolution hierarchies are also utilized in [72].

Conceptually, binary space-partitioning trees [73] are similar to octrees except that each subdivision is binary and segments the volume by a plane of arbitrary orientation. The subdivision terminates at a predetermined threshold of spatial resolution. This approach requires larger memory capacity than octrees but less than volume buffers. However, it is not inherently compatible with a regular parameterization of the reconstruction volume, which facilitates transmission and rendering.

Multiview stereo techniques aim to reconstruct a 3-D scene from multiple images that are simultaneously acquired from different viewpoints. In multiview stereo, there is no notion of a single "depth" dimension; for arbitrary camera locations, there is no semantic difference between x, y, and z directions. Volumetric representations are valuable, especially in multiview stereo-reconstruction algorithms, because they provide a common reference frame in which the results obtained from different views are combined [74]–[76]. Volumetric representations also include neighborhood information, meaning that they facilitate direct access to neighboring voxels. This property is essential for efficient computation of visibility- the requirement that voxels lying between the camera and an imaged surface be empty. This computation is essential in space-carving [77] and voxel-coloring [78] approaches, where the reconstruction result is obtained implicitly by detecting the empty voxels. Moreover, the access to neighborhood information facilitates operations such as 3-D convolution and detection of connected components; these are both useful operations for computation of local geometric properties (e.g., curvature [79]), as well as for noise-filtering of the reconstruction results. Based on a volumetric representation of the reconstruction, radial basis functions (RBF) can provide smooth interpolations and noise-filtering of the data [80], [81]. In RBF approaches, after a functional is computed for each voxel, the surface is extracted as an iso-surface by using either the Marching Cubes algorithm [82] or a surface-following [81] variation of this technique.

An advantage of voxel-based representations is their linear access time for the structured data. This property could be useful for efficient rendering of surface representations, independently from the complexity of the object. Furthermore, depth-sorting of voxels for application of the GPU-accelerated Z-buffering algorithm [83] is handled well. On the down side, voxels are rendered as cubes, resulting in poor visualization when the rendering viewpoint is close to the surface. In contrast, polygons produce continuous renderings of the reconstructed surface. In addition, the architecture of commodity-graphics hardware is designed to render polygons; mesh-based representations are therefore handled better. The straightforward conversion of a volumetric representation to a mesh is possible, e.g., by replacing each facet of the voxel with two triangles [84]; this is not however very useful since the result is not optimized in terms of memory capacity and the resulting geometrical structure is still shaped as an arrangement of cubes. Finally, voxel-based representations facilitate data-partitioning into subvolumes for parallel processing by more than one CPU [85] and exhibit a predetermined accuracy proportional to the third power of the resolution of the model stored in memory.

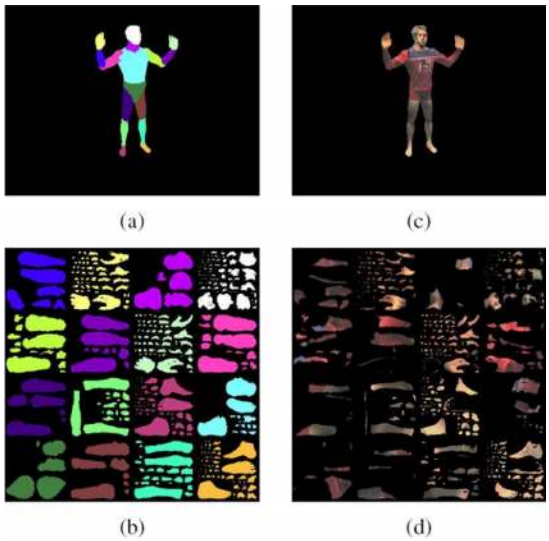

Fig. 10. Single texture atlas representation: (a) color coded body parts, (b) corresponding regions in texture space, (c) input frame, (d) resampled texture frame considering visibility [86]. Copyright © 2004, IEEE, Inc. Reprinted by permission.

V. Texture Representations

Texturing of 3-D geometric objects is essential for realistic scene rendering in 3DTV applications. The texture of a real object is usually extracted from a set of images that capture the surface of the object from various angles. These images can either be processed and merged into a single texture representation or stored separately to be used as multitextures.

A. Single Texture Representation

Single textures can occur in closed form, representing the appearance of the entire object surface by a single connected area. This is achieved by unwrapping the true object surface, i.e., by transforming it to a 2-D plane approximation. The texture information is stored in the form of a 2-D rectangular image which can then be coded and transmitted along with surface geometry information and eventually mapped onto the object surface during rendering. Single textures may also exist in open form, such as a texture atlas, where the image texture contains isolated texture patches for each part of a 3-D object [86], as shown in Fig. 10. Although closed-form texture representations can achieve higher compression ratios due to high correlation within the 2-D texture image area, they cannot be generalized to handle objects with arbitrary topology. A texture atlas on the other hand, can model the appearance of a surface with no restriction on topology. However, it is not as space efficient as closed form textures, since the 2-D texture image contains isolated texture parts with limited correlation. For both single texture representations, the appearance is static in terms of different lighting and reflection effects during navigation.

B. Multitexture Representation

Multitexturing was originally developed by the computer graphics community to apply environmental effects such as special lighting or reflection properties, to an image texture. In this case, the utilized set of textures consists of the original texture, plus a number of artificial textures, representing illumination and reflection effects. In contrast to this approach, recent developments in multicamera scenarios have led to the concept of multitexturing as a computer vision approach, where "multitexture" refers to a number of original camera views of a common 3-D scene object [87]. Multitextures include the environmental effects, as they appear in the original camera views. Hence, the naturalness of the rendered object mainly depends on the camera density and the interpolation strategy between original views. Fig. 11 shows a multitexturing example. For view-dependent texturing, there are different approaches, such as light field rendering [88], [89], light field mapping [90] or unstructured lumigraph rendering [91]. An overview of view-dependent texturing approaches can be found in [92].

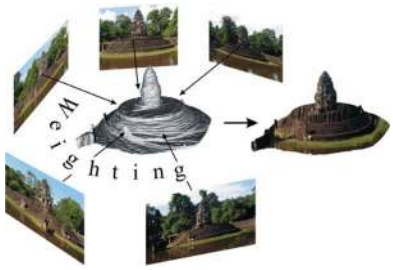

Fig. 11. Multitexture example with (view-dependent) weighted original camera views mapped onto 3-D geometry.

VI. Pseudo-3-D Representations

The term *pseudo-3D* refers to image-based representations that avoid using any explicit 3-D geometry to obtain a 3-D impression from 2-D video. Following the taxonomy in [93], a 3-D view can be composed from the input sequences by using either implicit geometric information or no geometric information at all. In this section, some representations obtained by image interpolation and image warping, as well as light field representations, are briefly summarized. Actually, the borderline between other representations is quite vague, since some of these methods employ techniques described in the previous sections.

Chen *et al.* create virtual views of still images by image-based interpolation [94], whereas Seitz *et al.* present physically correct view-morphing in their seminal paper [95]. In contrast, in image warping the virtual viewing position is not restricted to the baseline, which is the line between two camera centers. An example of such a method is the trifocal transfer that allows wide extrapolations from two or three closely positioned cameras [96]. All the aforementioned methods implicitly employ the available geometric information by utilizing the corresponding feature points between the images.

More recently, the domain of time-varying representations has also been studied; image interpolation of the objects in dynamic scenes is investigated in [97] and [98]. Some of the interpolation methods could be used for applications in sports TV transmissions and can thus be regarded as a pioneering pseudo-3DTV application [99]. In [100], natural video objects obtained by image warping are augmented with synthetic content by the help of MPEG-4 standard; this could also be an important tool in 3DTV applications (see Fig. 12).

Fig. 12. Pseudo 3-D video object augmented with a synthetic environment.

In a different approach, a light field or lumigraph is a 4–D simplification of the plenoptic function, as described by Adelson and Bergen in [101]. This concept was independently introduced both by Levoy *et al.* in [88] and Gortler *et al.* in [102]. In this technology, virtual views are obtained by interpolation from a parameterized representation that uses coordinates of the intersection points of light rays with two known surfaces. Light fields are often sampled by using large camera arrays. Dynamic light fields extend the function by the time dimension, which is essential for 3DTV applications. The most crucial issue in sampling dynamic light fields is the large volume of data. Hence, compression of such representations is investigated in [103] and [104]; the latter addresses the problem by using distributed light field rendering. However, the research direction in light field rendering leans towards using more sparsely arranged cameras and additional scene information, such as geometric proxies; this obviously leads to other scene representation methods that are covered in the other sections of this paper.

VII. Object-Based 3-D Scene Modeling

3-D dynamic scene modeling involves three basic tasks: *representation, animation*, and *rendering* (see Fig. 13). The research efforts in the field of computer graphics have already produced diverse techniques for realistic scene modeling that are amenable to real-time implementations and thus well suited for 3DTV applications.

A. Representation

A typical 3-D scene may contain moving objects of different types and complexities. The choice of the best surface representation scheme for a given object depends on its geometry, appearance and motion. A typical example for such 3-D scenes is an *outdoor environment* that comprises static urban scenery with living moving objects. Once a static scene for an outdoor environment is constructed, only any additional living objects need to be modeled, as dynamic components of this scene. In the literature, there are different building representations that can be used to model urban scenery and different ways to populate them to construct urban scenery [105], [106]. Similarly, there are different representations for terrain data, such as height fields and triangulated irregular networks [107]. These components could be integrated for a time-varying representation of the scenes captured using multiple cameras and/or multiple views.

As explained in Section III, the surface of any 3-D object can be approximated via polygons, parametric surfaces or point sets at increasing levels of accuracy. However, the methods based on Euclidean geometry, such as polygonal approximations, cannot describe typical natural objects, such as mountains, clouds, and trees, since these objects do not have regular shapes; their irregular or fragmented features cannot be realistically modeled by these methods [108]. As a solution to this shortcoming, *fractal-geometry* describes procedures to model natural objects, such as mountains, clouds, trees [109]. Lindenmayer systems (L-systems) can also be used for the realistic modeling of plants, since they define complex objects by successively replacing different parts of simple initial objects in parallel using a set of rewriting rules [110].

B. Animation

Computer animation makes use of computers to generate both key-frames and the frames in between. Animating a 3-D model can be regarded as generating the values of the model parameters over time. The models have various parameters, such as polygon vertex positions, joint angles, muscle contraction values, colors, and camera parameters. Animation is a process of varying the parameters over time and rendering the models to generate the frames. After the key-frames are generated, either by directly manipulating the models or by editing the parameter values, interpolation techniques, such as linear interpolation and spline interpolation, can be used to generate in-between positions [111], [112].

Recently, physically based modeling and animation has emerged as a new approach in computer animation [113]. The methods for modeling the shape and appearance of objects are not suitable for dynamic scenes where objects are moving. These models do not interact with each other or with external forces. In real life, the behavior and form of many objects are determined by such physical properties as mass, damping, and the internal and external forces acting on them. The deformability of the objects is determined by the properties of elasticity and inelasticity (such as internal stresses and strains) inherent in the material.

For realistic animation, one should model the physical properties of the objects to follow predefined trajectories and interact with the other objects in the environment, just like real physical objects. Physically based techniques achieve this kind of natural animation by adding physical properties to the models, such as forces, torques, velocities, accelerations, mass, damping, kinetic and potential energies, etc. Physical simulation is then used to produce animation based on these properties. The *initial value problems* must be solved so that the new positions and velocities of the objects are determined by the initial positions and velocities, and by the forces and torques applied to the objects.

Constraints provide a unified method to build and animate objects by specifying their physical behavior in advance without specifying their exact positions, velocities, etc., [114], [115].

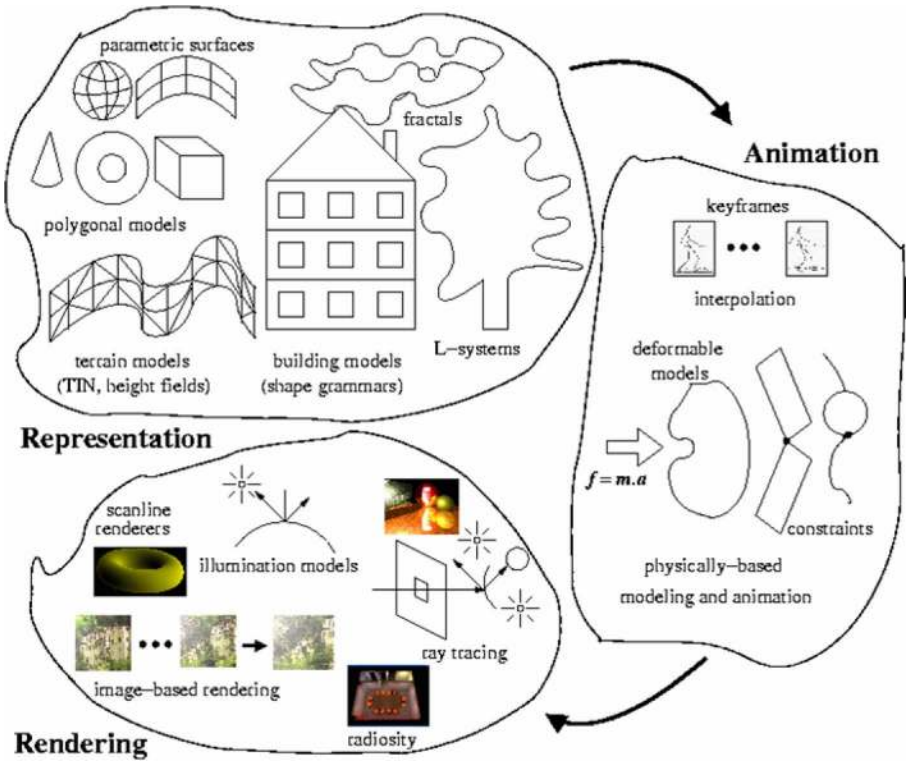

Fig. 13. Basic tasks involved in modeling of a dynamic 3-D scene that includes objects of different types: representation, animation, and rendering.

Thus, given a constraint, one must determine the forces to meet and maintain the constraint. Some examples of constraints used in animation are the *point-to-nail constraint*, which is used to fix a point on a model to a user-specified location in space; the *attachment constraint*, which is used to attach two points on different bodies to create complex models from simpler ones; the *point-to-path constraint*, which requires some points on a model to follow an arbitrary user-specified trajectory; the *orientation constraint*, which is used to align objects by rotating them, etc., [116].

An important aspect in realistic animation is modeling the behavior of *deformable objects*. This uses methods from elasticity and plasticity theory. However, such techniques are computationally demanding, since their simulations involve the numerical solution of the partial differential equations that represent the shape and motion of the deformable object through time [117]. For simulating the behavior of deformable objects, one should approximate a continuous model by using discretization methods. The trajectories of the points are determined by the properties of the deformable object. For example, in order to obtain the effect of an elastic surface, the grid points can be connected by springs. In fact, mass-spring systems are one of the simplest, yet most effective ways, to represent deformable objects.

There are two well established formulations to simulate the dynamics of elastically deformable models- *primal* [117] and *hybrid* [118]. These formulations use concepts from elasticity and plasticity theory; they represent the deformations by using such quantities from differential geometry as metric and curvature tensors. There are other approaches for deformable models: mathematical constraint methods, based on physics and optimization theory [114]; nonrigid dynamics, based on modal analysis that discards high-frequency modes having no effect on linear deformations and rigid body dynamics to reduce the dimensionality and stiffness [119]; nonrigid dynamics, based on simple linear global deformations with relatively few degrees of freedom [120]; constraint methods for connecting dynamic primitives that can be globally deformed (bends, tapers, twists, shears, etc.,) to make articulated models [121].

C. Rendering

The rendering literature is enormous; discussing all the rendering techniques is beyond the scope of this paper. For the purposes of 3DTV, we will concentrate on the techniques that are suitable in hardware implementations.

Rendering techniques try to model the interaction between light and the environment to generate pictures of scenes. This could be in the form of the Phong Illumination Model [122], which is a first order approximation to the rendering equation [123], or it could be very sophisticated techniques, such as ray tracing, radiosity, or photon mapping. Simple local illumination models do not consider the interaction of light with the environment, such as object-to-object light interactions (reflections, transmissions, etc.,). They only calculate the direct illumination from light sources on object surfaces.

Global illumination models calculate object-to-object interreflections, transmission, etc., to generate more realistic renderings, but they are computationally expensive. Ray tracing [124], [125] can only handle specular reflections, where the light

sources are point light sources. There are however some variations of ray tracing, such as *distributed* ray tracing, that increase the realism of the rendering by firing more rays to add spatial antialiasing, soft shadows, depth-of-field effects [126]. Radiosity [127] can only handle diffuse reflections and is useful for rendering room interiors; this is important for architectural walkthroughs. There are attempts to combine ray tracing and radiosity, but these attempts are only partially successful depending on the assumptions about the rendered scenes [128].

There are also some new approaches to global illumination of scenes, such as photon mapping that can make realistic rendering more affordable. Photon mapping uses forward ray-tracing (i.e., sending rays from light sources) to calculate reflecting and refracting light for photons [129]. It is a two-step process (distributing the photons and rendering the scene) that works for arbitrary geometric representations, including parametric and implicit surfaces; it calculates the ray-surface intersections on demand.

In general, rendering techniques are classified into two groups: *object-space* and *image-space*. Object-space techniques calculate the intensity of light for each point on an object surface (usually represented by polygonal approximations) and then use interpolation techniques to interpolate the intensity inside each polygon. Scan-line renderers, such as flat shading, Gouraud shading [130], and Phong shading, are in this category and they use local illumination models, e.g., the Phong Illumination Model, to calculate intensities at points. Radiosity is also an object-space technique. However it is a global illumination algorithm that solves the rendering equation only for diffuse reflections. Image-space techniques calculate intensities for each pixel on the image. Ray tracing is an image-space algorithm, which sends rays to the scene from the camera through each pixel and recursively calculates the intersections of these rays with the scene objects. The techniques, such as texture mapping [131], [132], environment mapping [133], and bump mapping [134], add realism to 3-D scenes, since the scenes are not usually uniformly shaded. Such techniques are not computationally intensive compared to global illumination algorithms and can be used together with scan-line renderers. Moreover, they can be implemented in real-time on GPUs.

In order to render a 3-D scene, the parts that are visible for different views must be calculated. This step requires the elimination of hidden surfaces. Some rendering algorithms, such as ray tracing and radiosity, handle the visible surface problem implicitly. It is handled explicitly by scan-line renderers, such as Gouraud and Phong shading, (e.g., using z-buffer).

VIII. Head and Body Specific Representations

Human faces and bodies form an integral part of most dynamic 3-D scenes. Therefore, the 3-D representation of the human face and body merit special attention in different scene representation technologies for 3DTV.

A. Modeling the Skeleton and the Body Appearance

Several articulated 3-D representations and mathematical formulations have been proposed to model the structure and movement of the human body. A human body model (HBM) can be represented as a chain of rigid bodies, called *links*, interconnected to one another by *joints*. Links are generally represented by sticks [135], polyhedrons [136], generalized cylinders [137] or superquadrics [138]. A joint connects two links by means of rotational motions around their axes. The number of independent rotation parameters defines the degrees of freedom (DOF) associated with a given joint. Development of a highly realistic HBM is a computationally expensive task, involving a problem of high dimensionality. In computer vision, where models need to be only moderately precise, articulated structures with low DOF are generally adequate [138], [139]. But, in computer graphics, highly accurate representations consisting of more than 50 DOF are usually desired [135].

The models proposed for the body appearance can be classified into four categories: stick figure models, surface models, volume models, and multilayer models. *Stick figure models* [140] are built by using a hierarchical set of rigid segments, connected by joints; they allow for easy control of movement, but realism is limited. *Surface models* are based on two layers: a skeleton, which is the backbone of the character animation, and a skin. The skin can use different types of primitives: points and lines, polygons [141], curved surface patches [142], [143], and subdivision surfaces [44]. In *volumetric models*, simple volumetric primitives, such as ellipsoids, spheres and cylinders [144] or implicit surfaces [141], [145] are used to construct the shape of the body. They perform better than surface models but it is difficult to control a large number of volumetric primitives during animation. *Multilayer models* consist of three layers: *skeleton, muscle* and *skin*. Complex motions are produced easily by building up the animation in different layers. Chadwick *et al.* were the first to use a muscle layer [146]. Nedel and Thalmann simulated muscles by a mass-spring system composed of angular springs [147].

B. Motion of the Skeleton

There are a number of ways to mathematically model an articulated human body using the kinematics and dynamics approaches. A mathematical model that describes the parameters of the links and the constraints associated with each joint is called a *kinematics model* and it can only describe the possible static states of a system [146], [148], [149]. In a *dynamic model*, the state vector includes positions, linear and angular velocities, accelerations, and the underlying forces and torques that act on this model [150], [151]. Dynamic model-based approaches are used to realistically animate walking models. However, dynamic model-based techniques are computationally more costly than kinematics-based techniques.

Determining the motion parameters explicitly at each frame, even for a simple motion, is not a trivial task. The solution is to specify a series of key-frame poses and interpolate the joint parameters between those key-frames [152]. Linear interpolation is the simplest method of generating the intermediate poses, but it produces a robotic motion due to discontinuous first derivatives in the interpolated joint angles. Obtaining smooth velocity and acceleration requires higher order interpolation methods, such as piecewise splines [153].

Since dynamics simulation cannot solve all animation problems, *motion capture* techniques have been introduced

to animate virtual characters from real human motion data. Motion capture methods are mainly used in the film and computer-game industries. The motion of a real actor is captured by tracking the 3-D positions and orientations of points located on him, using mechanical, electro magnetic or optical technologies [154], [155]. This method produces realistic and highly detailed motion in a short time. Since many application scenarios require no visual intrusion into the scene, researchers in computer vision have also investigated marker-free optical methods [156].

HBM can be described in various ways but for human body models to be interchangeable, a standard for animation is required. The Web 3-D H-anim [157] standards for human representation and the MPEG-4 representations for facial and body animation have been developed to meet this need [158].

C. 3-D Face Modeling and Animation

It is a major challenge to accurately model and animate the expressive human face using a computer [159]. Computer facial animation follows two basic steps: designing a 3-D facial mesh and animating that mesh to simulate facial movements. A face model with reasonable quality will typically consists of approximately 500 vertices. 3-D mesh can be designed interactively using computer graphics software or it can be captured via specific techniques. A general 3-D capturing technique uses photogrammetry, employing several images of the face, recorded from various angles [160]. Another technique is to place markers on the face that can be observed from two or more cameras. A more recent and accurate technique uses an automated laser scanner to digitize a person's head and shoulders in just a few seconds.

The literature on facial animation can be classified into three major methods: muscle-based [161], [162], rule-based (geometrical) [160], [163], and motion capture-based [164]. Muscle-based facial animation techniques can be further divided into two categories- physics-based muscle modeling and modeling with pseudo or simulated muscles. Physics-based muscle modeling mathematically describes the properties and behavior of human skin, bone and muscle systems by using mass-spring systems [165], vector representations [166], or layered spring meshes [167], [168]. Pseudo-muscle models mimic the dynamics of human tissue with heuristic geometric deformation of splines [169], [170] or free form deformations [171]. Physics-based muscle modeling produces realistic results by approximating human anatomy. However, the high computational cost of physics-based muscle modeling is a problem for real-time applications.

In *rule-based facial animation* [160], [172], a subset of the nodes of the geometry mesh, called feature points, is used to control the movement of the rest of the nodes of the geometry mesh. Although rule-based methods provide real-time deformations of the face, they may lack realism, as they are not based on any physical model. In [160], a rule-based animation method is used, where the face model consists of a dense geometry mesh and a sparse shape mesh. As part of the personalized 3-D face model, geometry mesh definitions conforming to each action state of the face are obtained. The facial expressions comply with the MPEG–4 standard [173] and are joy, sadness, anger, fear, disgust, and surprise. Visemes are defined manually; these define the lip shapes corresponding to the phonemes of speech. A shape interpolation approach animates the face over time by averaging and blending visemes and expressions according to predefined weights. Fig. 14 shows the facial expressions for a female character. Typical examples for lip-synchronized facial animations can be accessed in [174].

IX. Standards for 3-D Scene Representations

Interoperability is an important requirement, especially for the development of mass-consumer products in media-related industries. Therefore, open international standards play a key role in the success of new media technology. Standardized formats for media content enable interoperability between different systems, while still allowing for competition among equipment and service providers. The International Organization for Standardization (ISO) is an important body that provides format specifications for media content.

The Virtual Reality Modeling Language (VRML), released in 1997, is an ISO standard for 3-D scene representations [175]. It was mainly developed for the exchange of 3-D computer graphics data over the Internet. Although VRML provides a lot of useful elements, it has limited applicability for 3DTV applications because of limited real-time capabilities, and the lack of efficient point-based and other representations. Recently, ISO ratified the Extensible 3-D (X3D) [176] standard, as a successor to VRML; this was developed by the Web3D Consortium [177]. It provides improved real-time capabilities, including video and audio, as well as sophisticated computer graphics elements. The main focus of the design however, is still on computer-type systems and applications.

Standards for consumer electronics and telecommunication are developed by ISO/IEC JTC 1/SC 29/WG 11 (Moving Picture Experts Group—MPEG) and ITU-T SG 16 Q.6 (Video Coding Experts Group—VCEG) often in joint efforts resulting in joint specifications. With MPEG–4, for the first time, a universal multimedia framework has been created, which efficiently integrates natural video and audio with 3-D computer graphics [178]. MPEG–4 combines real-time streaming/display capabilities from a video/audio perspective with 3-D scene composition and rendering concepts from the computer graphics point of view. As such, it is perfectly suited for 3DTV applications.

A lot of computer graphics elements were already integrated in the initial versions of MPEG–4, including VRML as a subset, advanced tools for human face and body animation, and efficient compression of the data. A later addition, called Animation Framework eXtension (AFX) included sophisticated computer graphics tools such as depth image-based rendering and multitexturing [178]. Specific requirements for 3DTV have been studied in a subgroup, called 3DAV (for 3-D audio-visual) [179]; this group has initiated specific extensions of MPEG–4 to cover missing elements. Thus, MPEG–4 meets all the needs of 3-D scene representation for 3DTV as described in this paper, and will therefore very likely form the basis for a variety of 3DTV systems in the future [6].

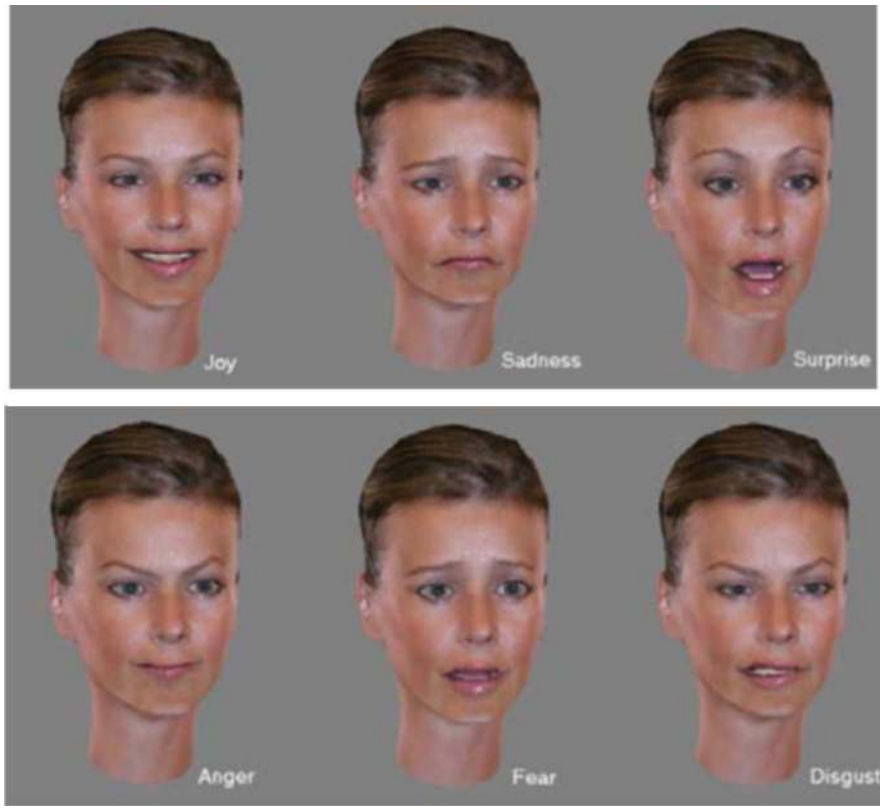

Fig. 14. Facial expressions for joy, sadness, and surprise, anger, fear, and disgust from left to right.

TABLE I
COMPARISONS BETWEEN DIFFERENT 3-D REPRESENTATIONS AND REQUIREMENTS (-: UNDEFINED, NUMBER OF STARS INDICATES DEGREE OF FULFILLMENT. *: "VERY POOR" & ******: "VERY GOOD")

	Dense depth	NURBS	Subdivision surfaces	Point sets	Pseudo-3D
Generality	**	*	****	*****	***
Accuracy	**	**	*****	*****	*
Perceptual quality	**	*****	*****	***	**
Level of detail	-	****	*****	***	**
Progressivity	-	-	*****	***	**
Compression	****	***	***	***	*****
Editing	-	*****	*****	**	-
Compatibility	**	***	****	**	***
Easiness to obtain	****	**	*	***	*****

X. CONCLUSION

A number of different scene-representation schemes for 3DTV have been examined in this paper. These schemes vary in the way they generate, transmit, and display dynamic 3-D scenes. A given representation depends strongly on the data available through recording and the type of display chosen for visualization. Table I presents an effort to summarize the performance of different representation strategies to fulfill various requirements for 3DTV systems.

Of the many 3-D scene-representation alternatives, dense point-based methods are most appropriate for free-view TV applications, since dense depth fields are good at rendering realistic novel views. Many auto-stereoscopic displays require multiviews of a scene to generate 3-D perception, which can efficiently be represented by dense depth maps. Hence, this type of representation is being standardized by ISO MPEG to serve for such auto-stereoscopic displays that could be assumed to be the first step towards 3DTV standardization.

Although dense depth representations are easy to construct and produce good visualization, they are incapable of modeling the scene semantics; they therefore fail to provide certain multimedia services and functionalities such as interaction, editing, animation, graphical realism, progressive modeling, and level of detail scalability. The computer graphics literature already offers a good variety of more capable surface-representation techniques. Each of these techniques has its own merits and disadvantages.

Polygonal meshes, which are currently the most common surface representations, are at one extreme. The polygons are the basic primitives of the hardware-rendering technologies; the state of the art for mesh-based schemes is quite mature, addressing several requirements of the emerging multimedia technologies. They can handle arbitrary topology and geometry in a very robust manner, but they have two major drawbacks. First, there is no inherent smoothness embedded in their description and their renderings may contain unrealistic artifacts. Second, is their lack of manipulation semantics, which limits the possibility of interactivity with and editing of highly detailed meshes that may contain millions of polygons.

At the other extreme are NURBS surfaces with embedded smoothness and efficient tools for editing and animation.

NURBS, however, has severe topological restrictions and limitations on the representation of fine surface detail. Subdivision surfaces offer a good compromise between polygonal meshes and NURBS surfaces. They can be regarded as a special case of the mesh representation with a specific type of connectivity; they yield truly progressive representations that converge to smooth surfaces at the limit. Subdivision surfaces can also be seen as a generalization of NURBS to arbitrary topology. In this sense, they unify the two extremes of 3-D modeling in an infrastructure that allows representation of arbitrary topology and any fine detail with controllable smoothness. The research on this area is still active and current techniques already satisfactorily address many problems of 3-D modeling. Their only limitation is the subdivision connectivity constraint which necessitates the use of remeshing techniques.

One recent and promising trend in surface representation is point-based modeling, i.e., the use of surface points as a simpler geometric primitive. Although many point-based techniques have addressed various functionalities, such as point-based animation, the state of the art is not yet mature. The main advantage of point-based schemes is that they require less preprocessing time, less storage, and enable fast rendering of highly detailed surfaces of arbitrary topology and geometry. Although the graphics-hardware technology does not yet support point-based rendering, existing software-based splatting techniques can achieve very high quality renderings at a speed of millions of points per second.

Another scene representation scheme considers volumetric approaches. Volumetric representations handle data structures that store intermediate results, facilitating the core functionality of certain algorithmic methods, rather than encoding and transmission of the final result. The most characteristic feature of these representations is the encapsulation of neighborhood information; this facilitates a wide variety of techniques for the reconstruction of surfaces. Volumetric data-structures also provide a common reference for multiview stereo algorithms and their parallel implementation, thus, providing a valuable representation for the implementation of full-blown 3DTV applications.

Texturing of 3-D geometric representations is essential for realistic scene rendering. In case of multicamera applications, two texturing methods can be considered. The first method is single texturing, where the camera views can be merged into a single texture or texture atlas, which is used for rendering. The second method is multitexturing, where the camera views are initially kept separate and later mapped onto an object in a view-dependent way. Texturing also provides a natural appearance in free viewpoint video applications due to inherent illumination properties.

A recent promising approach to render 3-D scenes is image-based rendering, so called *pseudo-3D*. Pseudo-3-D techniques directly render new views of a scene from the acquired images; they do not require any explicit geometry or texture representation. They often employ features from other methods, such as point-based or surface-based representations to reduce the amount of data. Some approaches produce good quality results which are already used in sports TV broadcasts as an important initial step towards real 3DTV systems.

One object-specific representation scheme targets 3-D human face and body modeling. Realistic rendering of human figures with reasonable computational complexity is important, since 3DTV might demand real-time animation of human bodies in time-varying scenes. Multilayer human models are useful for realistic rendering of the human face and body but they are still computationally demanding for real-time animation. It is expected that most of the current drawbacks will be solved with an increase in processing power in the next few years.

Physically based modeling and animation techniques produce impressive results but they are difficult to control, especially in terms of specifying motion and the way the movement should be performed. Researchers continue to present faster and simpler formulations to build and control the movement of models. Today, such approaches are used in the majority of feature films and games, using efficient techniques to approximate physics. Thus, physically based techniques also have potential for use in a 3DTV framework for modeling and animating dynamic scenes.

Rendering techniques, such as ray-tracing, radiosity and photon mapping, produce very realistic results but they cannot be implemented in real-time, since they are computationally intensive. Since scan-line renderers, such as Gouraud shading, are amenable to hardware implementations, they are more appropriate for 3DTV than sophisticated rendering techniques such as ray-tracing and radiosity given that 3DTV depends on real-time display capability. Hardware implementations enable the rendering of very complex models in real-time.

For all these formats and methods, open international standards supporting the presented 3-D scene representations are available and under development. X3D and MPEG-4 are particularly well suited for 3DTV applications and systems.

In conclusion, 3-D scene and texture representation technologies are mature enough to fulfill the requirements for 3-D extraction, transmission and display in a 3DTV scenario. The developments in those areas will determine the technologies to be utilized for representation.

ACKNOWLEDGMENT

The authors are grateful to Mrs. K. Ward for her proofreading and suggestions.

REFERENCES

[1] J. Shade, S. Gortler, L. W. He, and R. Szeliski, "Layered depth images," in *Proc. SIGGRAPH'98*, 1998, pp. 231–242.

[2] Y. Bayakovski, L. Levkovich-Maslyuk, A. Ignatenko, A. Konushin, D. Timasov, A. Zhirkov, M. Han, and I. K. Park, "Depth image-based representations for static and animated 3-D objects," in *Proc. IEEE ICIP'02*, 2002, pp. 22–25.

[3] C. Fehn, P. Kauff, M. Op de Beeck, F. Ernst, W. Ijsselsteijn, M. Pollefeys, L. Vangool, E. Ofek, and I. Sexton, "An evolutionary and optimised approach on 3D-TV," in *Proc.Int. Broadcast Convention (IBC)*, Sep. 2002, pp. 357–365.

[4] C. Fehn, E. Cooke, O. Schreer, and P. Kauff, "3-D analysis and image-based rendering for immersive TV applications," *Signal Processing: Image Comm.*, vol. 17, no. 9, pp. 705–715, Oct. 2002.

[5] Text of ISO/IEC 14496-16:2003/FDAM4. Awaji, Japan, Dec. 2002, ISO/IEC JTC1/SC29/WG11, Doc. N5397.

[6] A. Smolic and P. Kauff, "Interactive 3-D video representation and coding technologies," *Proc.IEEE*, vol. 93, no. 1, pp. 98–110, 2004.

[7] Y.-S. Ho, S.-U. Yoon, and S.-Y. Kim, "A Framework for Multi-View Video Coding Using Layered Depth Image," Hong Kong, ISO/IEC JTC1/SC29/WG11/M11582, Jan. 2005.

[8] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," in *Proc. SIGGRAPH'04*, 2004, pp. 600–608.

[9] A. A. Alatan and L. Onural, "Estimation of depth fields suitable for video compression based on 3-D structure and motion of objects," *IEEE Trans. Image Process.*, vol. 7, no. 6, pp. 904–908, 1998.

[10] S. H. Lee, Y. Kanatsugu, and J.-I. Park, "Hierarchical stochastic diffusion for disparity estimation," in *Proc. IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001, pp. 111–120.

[11] L. Alvarez, R. Deriche, J. Weickert, and J. Sanchez, "Dense disparity map estimation respecting image discontinuities: A PDE and scale space based approach," *J. Vis. Commun. Image Rep.*, vol. 47, pp. 229–246, 2002.

[12] C. Strecha, T. Tuytelaars, and L. Van Gool, "Dense matching of multiple wide-baseline views," in *Proc. IEEE Int. Conf. Comput. Vision*, 2003, vol. 2, pp. 1194–1201.

[13] H. Hoppe, "Progressive meshes," in *Proc. SIGGRAPH'96*, 1996, pp. 99–108.

[14] H. Hoppe, "View-dependent refinement of progressive meshes," in *Proc. SIGGRAPH'97*, 1997, pp. 189–198.

[15] G. Taubin, A. Gueziec, W. P. Horn, and F. Lazarus, "Progressive forest split compression," in *Proc. SIGGRAPH'98*, 1998, pp. 123–132.

[16] R. Pajarola and J. Rossignac, "Compressed progressive meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 6, no. 1, pp. 79–92, 2000.

[17] F. Pereira and T. Ebrahimi, *The MPEG-4 Book*. Upper Saddle River, NJ: Prentice Hall, 2002.

[18] G. Taubin and J. Rossignac, "Geometry compression through topological surgery," *ACM Trans. Graph.*, vol. 17, no. 4, pp. 84–115, 1998.

[19] G. Taubin, W. P. Horn, F. Lazarus, and J. Rossignac, "Geometry coding and VRML," *Proc.IEEE*, vol. 86, no. 6, pp. 1228–1243, Jun. 1998.

[20] J. Popovic and H. Hoppe, "Progressive simplicial complexes," in *Proc. SIGGRAPH'97*, 1997, pp. 217–224.

[21] A. Gueziec, F. Bossen, G. Taubin, and C. Silva, "Efficient compression of nonmanifold polygonal meshes," in *Proc. IEEE Vis.'99*, 1999, pp. 73–80.

[22] S. Kircher and M. Garland, "Progressive multiresolution meshes for deforming surfaces," in *Proc. ACM/EuroGraphics Symp. Computer Animation*, 2005, pp. 191–200.

[23] L. Piegl and W. Tiller, *The NURBS Book*. New York: Springer-Verlag, 1997.

[24] P. J. Schneider, "NURBS curves: A guide for the uninitiated," *Apple Tech. J.* no. 25, 1996 [Online]. Available: http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html

[25] S. Abi-Ezzi and S. Subramaniam, "Fast dynamic tessellation of trimmed NURBS surfaces," *Computer Graph. Forum*, vol. 13, no. 3, pp. 107–126, 1994.

[26] D. R. Forsey and R. H. Bartels, "Surface fitting with hierarchical splines," *ACM Trans. Graph.*, vol. 14, no. 2, pp. 134–161, 1995.

[27] F. J. M. Schmitt, B. A. Barsky, and W. H. Du, "An adaptive subdivision method for surface-fitting from sampled data," in *Proc. SIGGRAPH'86*, 1986, pp. 179–188.

[28] M. Eck and H. Hoppe, "Automatic reconstruction of B-spline surfaces of arbitrary topological type," in *Proc. SIGGRAPH'96*, 1996, pp. 325–334.

[29] V. Krishnamurthy and M. Levoy, "Fitting smooth surfaces to dense polygon meshes," *Proc. SIGGRAPH'96*, pp. 313–324, 1996.

[30] E. Catmull and J. Clark, "Recursively generated B-spline surfaces on arbitrary topological meshes," *Computer Aided Design*, vol. 10, no. 6, pp. 350–355, 1978.

[31] C. Loop, "Smooth Subdivision Surfaces Based on Triangles," M.S. Thesis, Dep. Math., Univ. of Utah, Salt Lake City, 1987.

[32] L. Kobbelt, "Interpolatory subdivision on open quadrilateral nets with arbitrary topology," *Proc. Eurographics'96 Comput. Graph. Forum*, vol. 15, no. 3, pp. 409–420, 1996.

[33] D. Zorin, P. Schröder, and W. Sweldens, "Interpolating subdivision for meshes with arbitrary topology," in *Proc. SIGGRAPH'96*, 1996, pp. 189–192.

[34] D. Doo and M. Sabin, "Analysis of the behavior of recursive division surfaces near extraordinary points," *Comput.-Aided Des.*, vol. 10, no. 6, pp. 356–360, 1978.

[35] U. Reif, "A unified approach to subdivision algorithms near extraordinary points," *Comput.-Aided Des.*, vol. 12, pp. 153–174, 1995.

[36] J. Stam, "Exact evaluation of Catmull–Clark subdivision surfaces at arbitrary parameter values," in *Proc. SIGGRAPH'98*, 1998, pp. 395–404.

[37] D. Zorin, P. Schröder, and W. Sweldens, "Interactive multiresolution mesh editing," in *Proc. SIGGRAPH'97*, 1997, pp. 259–268.

[38] A. Lee, H. Moreton, and H. Hoppe, "Displaced subdivision surfaces," in *Proc. SIGGRAPH'00*, 2000, pp. 85–92.

[39] A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *Proc. SIGGRAPH'00*, 2000, pp. 271–278.

[40] M. Lounsbery, T. DeRose, and J. Warren, "Multiresolution analysis for surfaces of arbitrary topological type," *ACM Trans. Graph.*, vol. 16, no. 1, pp. 34–73, 1997.

[41] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," in *Proc. SIGGRAPH'95*, 1995, pp. 173–182.

[42] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "MAPS: Multiresolution adaptive parameterization of surfaces," in *Proc. SIGGRAPH'98*, 1998, pp. 95–104.

[43] A. Khodakovsky, N. Litke, and P. Schröder, "Globally smooth parameterizations with low distortion," in *Proc. SIGGRAPH'03*, 2003, pp. 193–203.

[44] T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation," in *Proc. SIGGRAPH'98*, 1998, pp. 85–94.

[45] I. Guskov, W. Sweldens, and P. Schröder, "Multiresolution signal processing for meshes," in *SIGGRAPH'99*, 1999, pp. 325–334.

[46] M. Levoy and T. Whitted, "The use of points as a display primitive," Univ. North Carolina at Chapel Hill, Tech. Rep. TR-85022, 1985.

[47] J. Grossman and W. Dally, "Point sample rendering," *Proc. Rendering Techn.*, pp. 181–192, 1998.

[48] R. Szeliski and D. Tonnesen, "Surface modeling with oriented particle systems," *Proc. SIGGRAPH'92*, pp. 185–194, 1992.

[49] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," in *Proc. SIGGRAPH'00*, 2000, pp. 359–376.

[50] Y. Yemez and F. Schmitt, "Progressive multilevel meshes from octree particles," in *Proc. Int. Conf. 3-D Digital Imaging and Modeling*, 1999, pp. 290–299.

[51] Y. Yemez and F. Schmitt, "Multilevel representation and transmission of real objects with progressive octree particles," *IEEE Trans. Vis. Comput. Graph.*, vol. 9, no. 4, pp. 551–569, Oct.-Dec. 2003.

[52] S. Rusinkiewicz and M. Levoy, "QSplat: A multiresolution point rendering system for large meshes," in *Proc. SIGGRAPH'00*, 2000, pp. 343–352.

[53] S. Rusinkiewicz and M. Levoy, "Streaming QSplat: A viewer for networked visualization of large, dense models," in *Proc. Symp. Interactive 3-D Graph.*, 2001, pp. 63–68.

[54] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Periera, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, "The digital michelangelo project: 3-D scanning of large statues," in *Proc. SIGGRAPH'00*, 2000, pp. 131–144.

[55] M. Botsch, A. Wiratanaya, and L. Kobbelt, "Efficient high quality rendering of point sampled geometry," in *Proc. Eurographics Workshop on Rendering*, 2002, pp. 53–64.

[56] S. Fleishman, D. Cohen-Or, M. Alexa, and C. T. Silva, "Progressive point set surfaces," *ACM Trans. Graph.*, vol. 22, no. 4, pp. 997–1011, 2003.

[57] Y. Yemez and F. Schmitt, "3-D progressive compression with octree particles," in *Proc. VMV'02 Vision, Modeling and Vis.*, 2002, pp. 139–146.

[58] M. Zwicker, H. Pfister, J. van Baar, and M. H. Gross, "Surface splatting," in *Proc. SIGGRAPH'01*, 2001, pp. 371–378.

[59] M. Zwicker, H. Pfister, and J. van Baar, "EWA Splatting," *IEEE Trans. Vis. Comput. Graph.*, pp. 223–238, 2002.

[60] M. Botsch and L. Kobbelt, "High-quality point-based rendering on modern GPUs," *Proc. Pacific Graph.*, pp. 335–343, 2003.

[61] M. Zwicker, J. Räsänen, M. Botsch, C. Dachsbacher, and M. Pauly, "Perspective accurate splatting," *Proc. Graph. Interface*, pp. 247–254, 2004.

[62] R. Pajarola, M. Sainz, and P. Guidotti, "Confetti: Object-space point blending and splatting," *IEEE Trans. Vis. Comput. Graph.*, vol. 10, no. 5, pp. 598–608, Sep./Oct. 2004.

[63] M. Mueller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa, "Point-based animation of elastic, plastic, and melting objects," in *SIGGRAPH/Eurographics Symp. Computer Animation*, 2004, pp. 141–151.

[64] M. Zwicker, M. Pauly, O. Knoll, and M. Gross, "Pointshop 3D: An interactive system for point-based surface editing," in *Proc. SIGGRAPH'02*, 2002, pp. 322–329.

[65] A. Kaufman, R. Yagel, and D. Cohen, *Modeling in Volume Graphics*. New York: Springer-Verlag, 1993, pp. 441–454.

[66] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*. Reading, MA: Addison-Wesley, 1990.

[67] M. Gervautz and W. Purgathofer, "A simple method for color quantization: Octree quantization," in *New Trends in Computer Graphics*. New York: Springer, 1988, pp. 219–231.

[68] I. Garnagntini, "Linear octree for fast processing of three-dimensional objects," *Computer Graph. Image Process.*, vol. 20, pp. 365–374, 1982.

[69] I. Carlbom, I. Chakravarty, and D. Vanderschel, "A hierarchical data structure for representing the spatial decomposition of 3-D objects," *IEEE Comput. Graph. Appl.*, vol. 5, no. 4, pp. 24–31, 1985.

[70] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[71] M. de Berg, M. van Krevald, M. Overmars, and O. Schwarzkopf, *Comput. Geometry*. New York: Springer-Verlag, 2000.

[72] M. Ohlberger and M. Rumpf, "Hierarchical and adaptive visualization on nested grids," *Computing*, vol. 59, no. 4, pp. 365–385, 1997.

[73] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by *a priori* tree structures," in *Proc. SIGGRAPH'80*, 1980, pp. 24–133.

[74] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. SIGGRAPH'96*, 1996, pp. 303–312.

[75] M. Goesele, S. M. Seitz, and B. Curless, "Multi-view stereo revisited," in *Proc. CVPR*, 2006, vol. 2, pp. 2402–2409.

[76] X. Zabulis and K. Daniilidis, "Multi-camera reconstruction based on surface normal estimation," in *Proc. IEEE 3DPVT*, 2004, pp. 733–740.

[77] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space-carving," *Int. J. Comput. Vision*, vol. 38, no. 3, pp. 197–216, 2000.

[78] W. Culbertson, T. Malzbender, and G. Slabaugh, "Generalized voxel coloring," in *Vision Algorithms Theory and Practice Workshop*, 1999, pp. 100–114.

[79] G. Kamberov and G. Kamberova, "Topology and geometry of unorganized point clouds," in *Proc. IEEE 3DPVT*, 2004, pp. 743–750.

[80] G. Turk and J. F. O'brien, "Modelling with implicit surfaces that interpolate," *ACM Trans. Graph.*, vol. 21, no. 4, pp. 855–873, 2002.

[81] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3-D objects with radial basis functions," in *Proc. SIGGRAPH'01*, 2001, pp. 67–76.

[82] W. Lorenson and H. Cline, "Marching Cubes: A high resolution 3-D surface construction algorithm," in *Proc. SIGGRAPH'87*, 1987, pp. 163–170.

[83] E. Catmull, "A Subdivision Algorithm for Computer Display of Curved Surfaces," Ph.D. dissertation, Univ. Utah, Salt Lake City, 1974.

[84] S. W. Wang and A. E. Kaufman, "Volume-sampled 3-D modeling," *IEEE Comput. Graph. Appl.*, vol. 14, pp. 26–32, 1994.

[85] X. Zabulis and G. Kordelas, "Efficient, precise, and accurate utilization of the uniqueness constraint in multiview stereo," in *Proc. IEEE 3DPVT*, 2006.

[86] G. Ziegler, H. Lensch, N. Ahmed, M. Magnor, and H.-P. Seidel, "Multi-video compression in texture space," in *Proc. 2004 Int. Conf. Image Process. (ICIP'04)*, 2004, pp. 2467–2470.

[87] A. Smolic, K. Müller, P. Merkle, T. Rein, M. Kautzner, P. Eisert, and T. Wiegand, "Free viewpoint video extraction, representation, coding, and rendering," in *Proc. Int. Conf. Image Process. (ICIP'04)*, , Singapore, 2004, pp. 3287–3290.

[88] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. SIGGRAPH'96*, 1996, pp. 31–42.

[89] P. Debevec, C. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image based approach," in *Proc. SIGGRAPH'96*, 1996, pp. 11–20.

[90] W. C. Chen, J. Y. Bouguet, M. H. Chu, and R. Grzeszczuk, "Light field mapping: Efficient representation and hardware rendering of surface light fields," in *Proc. SIGGRAPH'02*, 2002, pp. 447–456.

[91] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proc. SIGGRAPH'01*, 2001, pp. 425–432.

[92] D. Vlasic, H. Pfister, S. Molinov, R. Grzeszczuk, and W. Matusik, "Opacity light fields: Interactive rendering of surface light fields with view-dependent opacity," in *Proc. Symp. Inter. 3-D Graph.*, 2003, pp. 65–74.

[93] H.-Y. Shum, S. B. Kang, and S.-C. Chan, "Survey of image-based representations and compression techniques," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 1020–1037, 2003.

[94] S. Chen and L. Williams, "View interpolation for image synthesis," in *Proc. ACM SIGGRAPH*, Aug. 1993, pp. 279–288.

[95] S. M. Seitz and C. M. Dyer, "View morphing," in *Proc. ACM SIGGRAPH'96*, New Orleans, LA, Aug. 1996, pp. 21–30.

[96] S. Avidan and A. Shashua, "Novel view synthesis by cascading trilinear tensors," *IEEE Trans. Vis. Comput. Graph.*, vol. 4, no. 4, pp. 293–306, 1998.

[97] R. A. Manning and C. R. Dyer, "Interpolating view and scene motion by dynamic view morphing," in *IEEE Conf. Comput. Vision and Pattern Recogn.*, 1999, vol. 1, pp. 388–394.

[98] J. Xiao, C. Rao, and M. Sha, "View interpolation for dynamic scenes," in *Proc. EUROGRAPHICS'02*, 2002, pp. 153–162.

[99] N. Inamoto and H. Saito, "Free viewpoint video synthesis and presentation from multiple sporting videos," in *Proc. IEEE Int. Conf. Adv. in Comput. Entertainment Tech.*, 2005, pp. 42–50.

[100] C. Weigel and L. L. Kreibich, "Advanced 3-D video object synthesis based on trilinear tensors," in *Proc. IEEE Int. Symp. Consumer Electron.*, Jun. 2006, pp. 1–5.

[101] E. H. Adelson and J. R. Bergen, , M. Landy and J. A. Movshon, Eds., "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*. Cambridge, MA: The MIT Press, 1991, ch. 1, Chapter.

[102] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," in *Proc. SIGGRAPH'96*, 1996, pp. 43–54.

[103] B. Wilburn, M. Smulski, K. Lee, and M. A. Horowitz, "The light field video camera," in *Proc. Media Processors'02*, 2002, SPIE Electronic Imaging.

[104] J. Yang, M. Everett, C. Buehler, and L. McMillan, "A Real-Time Distributed Light Field Camera," in *Proc. Eurographics Workshop on Rendering*. Pisa, Italy: , Jun. 2002, pp. 77–85.

[105] Y. I. H. Parish and P. Müller, "Procedural modeling of cities," in *Proc. SIGGRAPH'01*, 2001, pp. 301–308.

[106] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky, "Instant architecture," *ACM Trans. Graph*, vol. 22, no. 3, pp. 669–677, 2003.

[107] R. J. Fowler and J. J. Little, "Automatic extraction of irregular network digital terrain models," in *Proc. SIGGRAPH'79*, 1979, pp. 199–207.

[108] D. Hearn and P. Baker, *Computer Graph. with OpenGL*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 2003, Edition.

[109] B. B. Mandelbrot, *The Fractal Geometry of Nature*. San Francisco, CA: Freeman, 1982.

[110] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants The Virtual Laboratory*. New York: Springer, 1996.

[111] R. Parent, *Computer Animation: Algorithms and Techniques*. San Francisco, CA: Morgan-Kaufmann, 2001.

[112] A. Witkin, "Animation," in *Computer Graphics I Course Notes*. Pittsburgh, PA: Carnegie-Mellon Univ., 1995.

[113] A. Witkin and D. Baraff, "Physically based modeling," presented at the ACM SIGGRAPH Course Notes, #25, 2001.

[114] J. Platt and A. H. Barr, "Constraint methods for flexible models," in *Proc. SIGGRAPH'88*, 1988, vol. 22, no. 4, pp. 279–288.

[115] A. Witkin and M. Kass, "Spacetime constraints," in *Proc. SIGGRAPH'88*, 1988, vol. 22, no. 4, pp. 159–168.

[116] R. Barzel and A. H. Barr, "A modeling system based on dynamic constraints," in *Proc. SIGGRAPH'88*, 1988, vol. 22, no. 4, pp. 179–188.

[117] D. Terzopoulos, J. Platt, A. H. Barr, and K. Fleischer, "Elastically deformable models," in *Proc. SIGGRAPH'87*, 1987, vol. 21, no. 4, pp. 205–214.

[118] D. Terzopoulos and A. Witkin, "Physically based models with rigid and deformable components," *IEEE Comput. Graph. Appl.*, vol. 8, no. 6, pp. 41–51, 1988.

[119] A. Pentland and J. Williams, "Good vibrations: Modal dynamics for graphics and animation," in *Proc. SIGGRAPH'89*, 1989, vol. 23, no. 3, pp. 215–222.

[120] A. Witkin and W. Welch, "Fast animation and control of nonrigid structures," in *Proc. SIGGRAPH'90*, 1990, vol. 24, no. 4, pp. 243–252.

[121] D. Metaxas and D. Terzopoulos, "Dynamic deformation of solid primitives with constraints," in *Proc. SIGGRAPH'92*, 1992, vol. 26, no. 2, pp. 309–312.

[122] B.-T. Phong, "Illumination for computer generated pictures," *Commun. ACM*, vol. 18, no. 6, pp. 311–17, 1975.

[123] J. T. Kajiya, "The rendering equation," in *Proc. SIGGRAPH'86*, 1986, vol. 20, no. 4, pp. 143–150.

[124] A. Glassner, Ed., *An Introduction to Ray Tracing*. New York: Academic, 1989.

[125] T. Whitted, "An improved illumination model for shaded display," *Commun. ACM*, vol. 23, no. 6, pp. 343–349, 1980.

[126] L. R. Cook, T. Porter, and L. Carpenter, "Distributed raytracing," in *Proc. SIGGRAPH'84*, 1984, pp. 137–145.

[127] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modeling the interaction of light between diffuse surfaces," in *Proc. SIGGRAPH'84*, 1984, pp. 213–222.
[128] J. R. Wallace and K. A. Elmquist, "A ray tracing algorithm for progressive radiosity," in *Proc. SIGGRAPH'90*, 1989, vol. 23, no. 4, pp. 315–324.
[129] H. W. Jensen, *Realistic Image Synthesis Using Photon Mapping*. Reading, MA: Addison-Wesley, 2001.
[130] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Trans. Comput.*, vol. 20, no. 6, pp. 623–628, Jun. 1971.
[131] J. F. Blinn and M. E. Newell, "Texture and reflection in computer generated images," *Commun. ACM*, vol. 19, no. 10, pp. 542–547, 1976.
[132] P. Heckbert, "Survey of texture mapping," *IEEE Comput. Graph. Appl.*, vol. 6, no. 11, pp. 56–67, 1986.
[133] N. Greene, "Environment mapping and other applications of world projections," *IEEE Comput. Graph. Appl.*, vol. 6, no. 11, pp. 21–29, 1986.
[134] J. F. Blinn, "Simulation of wrinkled surfaces," in *Proc. SIGGRAPH'78*, 1978, vol. 12, no. 3, pp. 286–292.
[135] A. Aubel, R. Boulic, and D. Thalmann, "Real-time display of virtual humans: Levels of details and impostors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 2, pp. 207–217, Feb. 2000.
[136] M. Yamamoto, A. Sato, S. Kawada, T. Kondo, and Y. Osaki, "Incremental tracking of human actions from multiple views," presented at the IEEE Int. Conf. Computer Vision and Pattern Recogn., Santa Barbara, CA, 1998.
[137] I. Cohen, G. Medioni, and H. Gu, "Inference of 3-D human body posture from multiple cameras for vision-based user interface," in *Proc. World Multiconf. on Syst., Cybern. Informatics*, 2001.
[138] D. M. Gavrila and L. Davis, "3-D model-based tracking of humans in action: A multiview approach," in *Proc. IEEE Int. Conf. Comput. Vision Pattern Recogn.*, 1996.
[139] Q. Delamarre and O. Faugeras, "3-D articulated models and multiview tracking with physical forces," *Comput. Vision and Image Understanding*, vol. 81, pp. 328–357, 2001.
[140] N. Badler and S. Smoliar, "Digital representations of human movement," *ACM Computing Surveys*, vol. 11, no. 1, pp. 19–38, 1979.
[141] N. Magnenat-Thalmann, J. Shen, and E. Cahuvineau, "Fast human body deformations for animation and VR applications," *Proc. Comput. Graph. Int.*, pp. 166–174, 1996.
[142] N. Magnenat-Thalmann and D. Thalmann, *Synthetic Actors in 3-D Computer Generated Films*. New York: Springer-Verlag, 1990.
[143] K. Komatsu, "Human skin model capable of natural shape variation," *The Visual Computer*, vol. 3, no. 5, pp. 265–271, 1988.
[144] S. Yoshimoto, "Ballerinas generated by a personal computer," *Vis. Comput. Animation*, vol. 3, no. 1, pp. 85–90, 1992.
[145] J. Bloomenthal, Hand crafted: Modeling, Visualization and Animating Implicit Surfaces ACM SIGGRAPH Course Notes-#25, 1993.
[146] J. Chadwick, D. Haumann, and R. Parent, "Layered construction for deformable animated characters," *ACM Comput. Graph.*, vol. 23, no. 3, pp. 243–252, Jul. 1989.
[147] L. Nedel and D. Thalmann, "Real time muscle deformations using mass-spring systems," *Proc. Computer Graph. Int. CGI'98*, pp. 156–165, 1998.
[148] N. Badler, K. Manoochehri, and G. Walters, "Articulated figure positioning by multiple constraints," *IEEE Comput. Graph. Appl.*, vol. 7, no. 6, pp. 28–38, 1987.
[149] M. Girard and A. Maciejewski, "Computational modeling for computer generation of legged figures," in *Proc. SIGGRAPH'85*, 1985, pp. 263–270.
[150] J. Wilhelms, , N. Badler, B. Barsky, and D. Zeltzer, Eds., *Making Them Move: Mechanics, Control and Animation of Articulated Figures*. San Francisco, CA: Morgan-Kaufmann, 1991, ch. 13, pp. 265–280.
[151] H. Ko and N. Badler, "Animating human locomotion in real-time using inverse dynamics, balance and comfort control," *IEEE Computer Graph. App.*, vol. 16, no. 2, pp. 50–59, 1996.
[152] P. Hanrahan and D. Sturman, "Interactive animation of parametric models," *The Visual Computer*, vol. 1, pp. 260–266, 1987.
[153] J. Steketee and N. Badler, "Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control," in *Proc. SIGGRAPH'85*, 1985, pp. 255–262.
[154] M. C. Silaghi, R. Plaenkers, R. Boulic, P. Fua, and D. Thalmann, "Local and global skeleton fitting techniques for optical motion capture," *Lecture Notes on Artif. Intell.*, no. 1537, pp. 26–40, 1998.
[155] A. Menache, *Understanding Motion Capture for Computer Animation and Video Games*. San Francisco, CA: Morgan Kaufmann, 1995.
[156] D. Gavrila, "The visual analysis of human movement," *Comput. Vis. and Image Understanding*, vol. 73, no. 1, pp. 82–98, 1999.
[157] Specification of a Standard VRML Humanoid. (2007) [Online]. Available: http://h-anim.org
[158] *"MPEG-4 Animation Framework eXtension (AFX) VM 9.0,"* M. Preda, Ed., ISO/IEC JTC1/SC29/WG11 N5245, 2002.
[159] F. I. Parke and K. Waters, *Computer Facial Animation*. Wellesley, MA: A. K. Peters, 1996.
[160] A. T. Erdem, "A new method for 3-D face model generation and animation of personalized game characters," *Int. J. Intell. Games Simul.*, vol. 3, no. 1, pp. 20–28, 2004.
[161] D. Terzopoulos and K. Waters, "Analysis and synthesis of facial image sequences using physical and anatomical models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 6, pp. 569–579, Jun. 1993.
[162] X. Wei, Z. Zhu, L. Yin, and Q. Ji, "A real time face tracking and animation system," *Proc. IEEE Workshop on Face Process. in Video*, 2004.
[163] A. M. Tekalp and J. Ostermann, "Face and 2-D mesh animation in MPEG-4," *Signal Process. Image Commun.*, vol. 15, no. 4–5, pp. 387–421, 2000.
[164] J. Chai, J. Xiao, and J. Hodgins, "Vision based control of 3-D facial animation," in *Proc. ACM Symp. Comput. Animation*, 2003, pp. 193–206.
[165] S. M. Platt, "A structural model of the human face," Ph.D. dissertation, Univ. Pennsylvania, Philadelphia , 1985.
[166] K. Waters, "A muscle model for animating three-dimensional facial expressions," *Proc. SIGGRAPH'87*, vol. 21, no. 4, pp. 17–24, 1987.
[167] D. Terzopoulos and K. Waters, "Physically based facial modeling, analysis and animation," *J. Vis. Comput. Animation*, vol. 1, no. 4, pp. 73–80, 1990.
[168] Y. C. Lee, D. Terzopoulos, and K. Waters, "Realistic face modeling for animation," in *Proc. SIGGRAPH'95*, 1995, pp. 55–62.
[169] M. Nahas, H. Hutric, M. Rioux, and J. Domey, "Facial image synthesis using skin texture recording," *The Visual Computer*, vol. 6, pp. 337–343, 1990.
[170] C. L. Y. Wang and D. R. Forsey, "Langwidere: A new facial animation system," *Proc. Comput. Animation*, pp. 59–68, 1994.
[171] P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann, "Simulation of facial muscle actions based on rational free form deformations," *Proc. Eurographics*, vol. 11, no. 3, pp. 59–69, 1992.
[172] J. Dalong, L. Zhiguo, W. Zhaoqi, and G. Wen, "Animating 3-D facial models with MPEG-4 facedeftables," in *Proc. 35th Annu. Simulation Symp.*, 2002, pp. 395–400.
[173] I. S. Pandzic and R. Forchheimer, Eds., *MPEG-4 Facial Animation—The Standard, Implementations, and Applications*. Hoboken, NJ: Wiley, 2002.
[174] A. S. Momentum [Online]. Available: http://www.momentum-dmt.com
[175] *Virtual Reality Modeling Language (VRML)*, ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004, Dec. 2003.
[176] *Extensible 3-D (X3D)*, ISO/IEC 19775:2004, Nov. 2005.
[177] Web 3-D Consortium. [Online]. Available: http://www.web3d.org
[178] M. Bourges-Sevenier and E. S. Jang, "An introduction to the MPEG-4 animation framework extension," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 7, pp. 928–936, Jul. 2004.
[179] A. Smolic and D. McCutchen, "3DAV exploration of video-based rendering technology in MPEG," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 3, pp. 348–356, Mar. 2004.

A. Aydın Alatan (S'91–M'07) received the B.S. degree from Middle East Technical University, Ankara, Turkey, in 1990, the M.S. and D.I.C. degrees from Imperial College of Science, Medicine, and Technology, London, U.K., in 1992, and the Ph.D. degree from Bilkent University, Ankara, Turkey, in 1997, all in electrical engineering.

He was a Postdoctoral Research Associate at the Center for Image Processing Research at Rensselaer Polytechnic Institute between 1997 and 1998 and at New Jersey Center for Multimedia Research, New Jersey Institute of Technology between 1998 and 2000. In August 2000, he joined the faculty of the Electrical and Electronics Engineering Department at Middle East Technical University. His research interests include 3–D scene analysis, content-based video indexing and retrieval, data hiding and watermarking for visual content, image/video compression and robust transmission, automatic target detection, tracking and recognition.

Yücel Yemez (M'03) received the B.S. degree from Middle East Technical University, Ankara, Turkey, in 1989, and the M.S. and Ph.D. degrees from Bogaziçi University, Istanbul, Turkey, in 1992 and 1997, respectively, all in electrical engineering.

From 1997 to 2000, he was a Postdoctoral Researcher in the Image and Signal Processing Department, Télécom Paris (Ecole Nationale Supérieure des Télécommunications), Paris, France. Currently, he is an Assistant Professor of the Computer Engineering Department, Koç University, Istanbul, Turkey. His current research is focused on various fields of computer vision and graphics.

Uğur Güdükbay (M'00–SM'05) received the B.S. degree in computer engineering from Middle East Technical University, Ankara, Turkey, in 1987, and the M.S. and Ph.D. degrees in Computer Engineering and Information Science from Bilkent University, Ankara, Turkey, in 1989 and 1994, respectively.

He conducted research as a Postdoctoral Fellow at Human Modeling and Simulation Laboratory, University of Pennsylvania, Philadelphia, PA. Currently, he is an Associate Professor at Department of Computer Engineering, Bilkent University. His research interests include physically based modeling, human modeling and animation, multiresolution modeling, visualization, multimedia databases, computational geometry, cultural heritage, and electronic arts.

He is a Professional Member of ACM.

Xenophon Zabulis received the B.A., M.S., and Ph.D. degrees in computer science degree from the University of Crete, Crete, Greece, in 1996, 1998, and 2001, respectively.

He has worked as a Postdoctoral Fellow at the Institute for Research in Cognitive Science and at the General Robotics, Automation, Sensing and Perception Laboratory, both at the Computer and Information Science Department, University of Pennsylvania, Philadelphia. He is currently a Research Fellow at the Institute of Informatics and Telematics—Centre of Research and Technology Hellas, Greece.

Karsten Müller (M'98–SM'07) received the Dipl.-Ing. and Dr.-Ing. degree in electrical engineering from the Technical University of Berlin, Berlin, Germany, in 1997 and 2006, respectively.

He has been with the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut, Berlin, Germany, since 1997, where he is currently Project Manager. His research interests are mainly in the field of representation, coding, and reconstruction of 3-D scenes in free viewpoint video scenarios and coding, multiview applications and combined 2D/3-D similarity analysis. He has been involved in MPEG activities, where he contributed to visual MPEG-7 descriptors and MPEG-4.

Çiğdem Eroglu Erdem (S'93–M'03) received the Ph.D. degree from the Department of electrical and electronics engineering, Bogaziçi University, Istanbul, Turkey, in 2002.

From September 2000 to June 2001, she was a Visiting Researcher in the Department of Electrical and Computer Engineering, University of Rochester, NY. Between 2003–2004, he was a Postdoctoral Fellow at the Faculty of Electrical Engineering, Delft University of Technology, The Netherlands, where she was also affiliated with the video processing group at Philips Research Laboratories, Eindhoven. She is currently a Senior Researcher at Momentum Digital Media Technologies Inc., Istanbul, Turkey. Her research interests are in the areas of digital image, video and speech processing, including motion estimation, video segmentation, object tracking, 3-D scene reconstruction, and speech processing for natural face animation.

Christian Weigel is currently working toward the Ph.D. degree at the Institute for Media Technology of the Technische Universität Ilmenau, Ilmenau, Germany, focusing on the scalable representation of 3-D video objects.

He is currently doing research on the fields of video technology with focus on 3-D video objects. He has been working in the IAVAS (Interactive Audiovisual Application Systems) project concerning the application of the MPEG–4 standard in interactive virtual environments. He published papers regarding applications employing the MPEG-4 standard and 3-D video production.

Aljoscha Smolic received the Dipl.-Ing. degree in electrical engineering from the Technical University of Berlin, Berlin, Germany in 1996, and the Dr.-Ing. degree in electrical and information engineering from Aachen University of Technology (RWTH), Aachen, Germany, in 2001.

He joined the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut (HHI), Berlin, Germany, in 1994 and has been Scientific Project Manager since 2001. He has been involved in several national and international research projects, where he conducted research in various fields of video processing, video coding, computer vision and computer graphics and published more than 80 referred papers in these fields. In this context, he has been involved in ISO standardization activities where he contributed to the development of the multimedia standards MPEG-4 and MPEG-7. In current projects, he is responsible for research in free viewpoint and 3-D video processing and coding, augmented reality, 3-D reconstruction and video-based rendering. Since 2003, he has been an Adjunct Professor at the Technical University of Berlin, Berlin, Germany, and teaches multimedia communications and statistical communications theory. He chaired the MPEG ad hoc group on 3DAV pioneering standards for 3-D video. Currently , he is the Editor of the Multiview Video Coding (MVC) standard and co-chairs a working group of the JVT for MVC.

Dr. Smolic received the "Rudolf-Urtlel-Award" of the German Society for Technology in TV and Cinema (FKTG) for his dissertation in 2002. He is Area Editor for *Signal Processing: Image Communication* and Guest Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and IEEE *Signal Processing Magazine.* He is a Committee Member of several conferences, including ICIP, ICME, and EUSIPCO.