A unified representation for interactive 3D modeling

Dragan Tubić, Patrick Hébert, Jean-Daniel Deschênes and Denis Laurendeau
Computer Vision and Systems Laboratory, University Laval, Québec, Canada
[tdragan,hebert,laurendeau]@gel.ulaval.ca

Abstract

Interactive 3D modeling is the process of building a 3D model of an object or a scene in real-time while the 3D (range) data is acquired. This is possible only if the computational complexity of all involved algorithms is linear with respect to the amount of data. We propose a new framework for 3D modeling where a complete modeling chain meets with this requirement. The framework is based on the use of vector fields as an implicit surface representation. Each modeling step, registration, surface reconstruction, geometric fusion, compression and visualization is solved and explained using the vector fields without any intermediate representations. The proposed framework allows model reconstruction from any type of 3D data, surface patches, curves, unorganized sets of points or a combination of these.

1. Introduction

Three dimensional modeling from 3D (range) data is the process of building a 3D model of a measured object. Depending on the type of 3D data (unorganized sets of points, surface curves or surface patches) 3D modeling consists of several steps: surface reconstruction [6, 10, 13] and/or geometric fusion[1] [3, 5, 8, 12, 14, 15], registration [1, 12, 14], compression and visualization. Due to the computational complexity, the model reconstruction and the acquisition processes are usually performed separately.

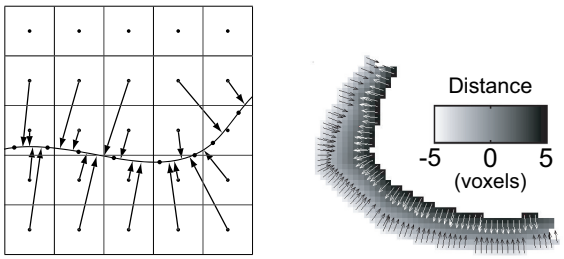
Current trends in 3D modeling indicate that the next challenge is building interactive 3D modeling systems, where a model is reconstructed on-line, in real-time, during the acquisition of data. A reconstructed model provides a visual feedback to the user so that unobserved regions can be readily identified and scanned. This greatly reduces required acquisition time especially in cultural heritage applications where the access to the artefacts is restrained. Also, by providing the model, some errors such as scanner miscalibration, can be detected and corrected immediately instead of returning to the site and repeating the acquisition. Although interesting work has been published [2, 7, 9, 11] there were no complete real-time modeling systems due to the computational complexity with respect to the quantity of measured data which can be as large as several billion 3D points. To be able to handle the 3D data in real-time, each step in an interactive modeling system has to be of linear computational complexity. Until now this was not the case and we believe that the solution for this problem is finding an appropriate surface representation rather than developing more sophisticated modeling algorithms.

The surface representation in 3D modeling is usually not considered as a problem in its own right. In some approaches, polygonal meshes are used throughout the modeling chain [12] which does not allow linear computational complexity at each step. In others, a signed distance field is used for surface reconstruction [6, 10] or geometric fusion [3, 5] while polygonal meshes are used for registration and for visualization, with notable exception of Masuda [8] who uses signed distance fields for registration too. However, this method is limited to model building from partial surfaces only, it is not of linear complexity (uses k-d trees) and the final visualization is performed by extracting a polygonal mesh. Using different representations results in a loss of efficiency and unnecessary computational steps. Furthermore, this prevents generalization of the model reconstruction with respect to the type of 3D data.

As a solution for the above problems, we propose an implicit surface representation as a vector field. In the following, a framework based on this representation is presented. Unlike existing methods, the proposed framework enables the building of a complete modeling chain from surface reconstruction to the final visualization using only vector fields as the surface representation. Furthermore, it will be shown that the same representation provides linear complexity in all modeling steps. Besides the complexity advantage, our approach simplifies surface reconstruction and in some cases allows reconstruction not previously possible such as reconstruction from surface curves and unified reconstruction from combined 3D data.

In our recent work [13, 14], we have presented algo-

[1] Since the surface patches already represent a surface, building a model from patches consists in removing redundant data and is usually referred to as geometric fusion. Building a model from curves or unorganized sets of points is referred to as surface reconstruction.



(a) (b)

Figure 1. Example of the vector field computed on a regular grid. a) The values of the field represent the direction and the distance toward the closest point on the surface. b) The norm of the vector field is encoded as grey-levels with associated directions toward the surface.

rithms for 3D modeling from range images and curves. In this paper we present a new modeling framework where all the modeling steps are solved using a single implicit representation, vector fields, including reconstruction from unorganized points, unified reconstruction from all types of data as well as visualization. The principle of each modeling step is explained as well as how to achieve linear computational complexity. The remainder of the paper is organized as follows. Section 2 presents the proposed surface representation. Surface reconstruction from the three types of 3D data, as well as unified reconstruction, are detailed in Sections 3 and 4, followed by registration in Section 5. Compression and visualization are presented in Sections 6 and 7, respectively.

2. Vector field surface representation

Explicit surfaces, such as triangulated meshes, are the most commonly used surface representations for 3D modeling and computer graphics. However, a major drawback of explicit representations is their inappropriate organization of the data, which makes it computationally expensive to perform proximity operations such as the search for closest points on the surface[2]. Remarkably, the search for closest points is the corner stone of all modeling steps and it is used in one form or another throughout the modeling chain. For the purpose of interactive modeling, linear complexity is an absolute requirement. Otherwise the modeling system slows down as the amount of data increases, and eventually comes to a halt. As we will show later, linear complexity can be achieved by merely changing the representation of the surface. Moreover, most of the modeling tasks can be greatly simplified and unnecessary computations (such as the computation of k-d trees) can be avoided.

The proposed representation is an implicit surface representation as a vector field. The vector field at each point in 3D space represents the direction and the distance toward the closest point on the surface, as illustrated in Figure 1a. The surface itself is represented as the zero set of the vector field norm. More formally, let a surface S be defined as a differential map $S : U \in R^2 \rightarrow R^3$. The vector field $\mathbf{f} : V \subset R^3 \rightarrow R^3$ representing the surface S is defined as

$$\mathbf{f}(\mathbf{p}) = \underset{\mathbf{q} \in S}{\operatorname{argmin}} \|\mathbf{p} - \mathbf{q}\| - \mathbf{p} \tag{1}$$

and

$$V = \{\mathbf{p} \in R^3 \,|\, \mathbf{p} = \mathbf{c} + \|\mathbf{f}(\mathbf{p})\| \, \mathbf{n}(\mathbf{c}), \mathbf{c} = \underset{\mathbf{q} \in S}{\operatorname{argmin}} \|\mathbf{p} - \mathbf{q}\|\}$$

where $\mathbf{n}(\mathbf{c})$ denotes the unit normal on the surface at point $\mathbf{c}$. The restriction on the set V prevents the field from being computed in regions near the surface discontinuities, i.e. in regions where the surface normal does not coincide with the direction of the vector field (hatched region in Figure 5).

In order to reduce computation costs, the field is computed on a regular lattice of points (voxels) and only in the neighbourhood of the surface, i.e. at points that are closer to the surface than a predefined constant distance $\epsilon > 0$. This neighbourhood is said to be enclosed by a volumetric envelope. Being defined as the zero-crossing of the norm (which occurs between neighbouring voxels), the surface is adequately represented as long as ϵ is larger than the voxel's diagonal (the largest distance between two neighbouring voxels). This representation can also be considered as an augmented signed distance field where the direction is added to the distance (see Figure 1b). It should also be noted that, unlike scalar distance fields, the vector field at each point defines exactly a point on the surface. This can be used to improve surface extraction from the vector field.

3. Surface reconstruction

In the proposed framework, surface reconstruction means building the vector field representation from measured 3D data. As shown in Figure 2a, to compute the vector field that represents a surface, a local *orientation* of the surface has to be estimated, that is, tangent planes. The vector field is then computed relative to these tangent planes as illustrated in Figure 2b.

A local estimate of the surface orientation has to be inferred from relative positions of the measured 3D points. The value of the vector field at a voxel is estimated using points in its neighbourhood, for example by fitting a plane. However, if the 3D data consists of multiple views, a small positioning error can cause an inaccurate estimate of surface orientation. To circumvent this problem, another

[2]Using a plain triangulated surface, the complexity of the single closest point search is $O(N)$ with respect to the number of points on the surface. Using a specialized data structure, i.e. a different representation, such as kd-trees, this complexity can be reduced to $O(logN)$, but not to the minimum $O(1)$ which is required for interactive modeling.

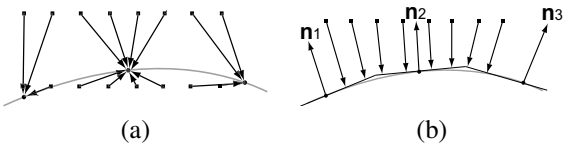


Figure 2. Surface reconstruction from 3D data a) The vector field cannot be built by pointing vectors toward closest points. b) The surface is reconstructed by locally estimating surface orientation, i.e. tangent planes (whose normals are vectors $\mathbf{n}_i$ in this example). The vector field is computed relative to those tangent planes.

way to proceed is to use additional information that is contained in the connectivity of points in curves and surfaces patches. For triangulated surfaces, the orientation of the reconstructed surface is estimated as an average of normals at vertices in the neighbourhood of a voxel. The effect of the registration error is then much smaller. Similarly, curve tangents can be used to improve reconstruction and reduce the effect of registration errors.

Another reason to use connectivity information in 3D data (tangents or normals) is the ambiguity that leads to completely wrong surface reconstruction. One example of such an ambiguity is shown in Figure 3 where a single curve is measured on a cylindrical object. By considering this curve as an unorganized set of points, the reconstructed surface will inevitably be a plane that contains the curve and will be perpendicular to the real surface. The method proposed in Section 3.3 for surface reconstruction from curves eliminates this problem by using curve tangents.

In the following it is explained how to reconstruct the surface (as the vector field) from unorganized sets of points, curves, surface patches as well as how to combine these three types of data. The additional information about surface orientation, curve tangents and surface normals, are exploited for improving the quality of reconstruction.

3.1. Surface reconstruction from partial views (range images)

Partial views (surface patches or range images) are represented as a 2D grid, where each point represents the distance between the sensor and the object. Being in a regular parametric grid, the points can be easily connected to their immediate neighbours in order to construct a triangulated surface patch. Since the initial data is already a surface, model reconstruction is usually referred to as geometric fusion, with the purpose of merging the sections of the surface that are observed more than once.

Building the vector field from a single triangulated surface is straightforward: each voxel within the volumetric envelope contains the vector pointing to the closest point of

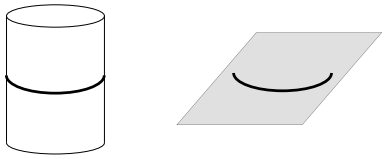


Figure 3. Considering all 3D data as unorganized sets of points might lead to wrong surface reconstruction. In this example, the surface reconstructed from a measured curve (left) is a plane perpendicular to the real surface (right).

the closest triangle. Although finding the closest triangles for each voxel can be performed using brute-force search or more efficient k-d trees, it is not of linear complexity. This problem is solved by using the inverse approach where the closest voxels are found for each triangle. For this purpose, each triangle defines a so-called fundamental cell [13, 14] which is the region influenced only by that triangle. The shape of the cell can be computed and, therefore, the voxel within the cell can be found. This way, the complexity is linear with respect to the number of triangles.

3.2. Geometric fusion

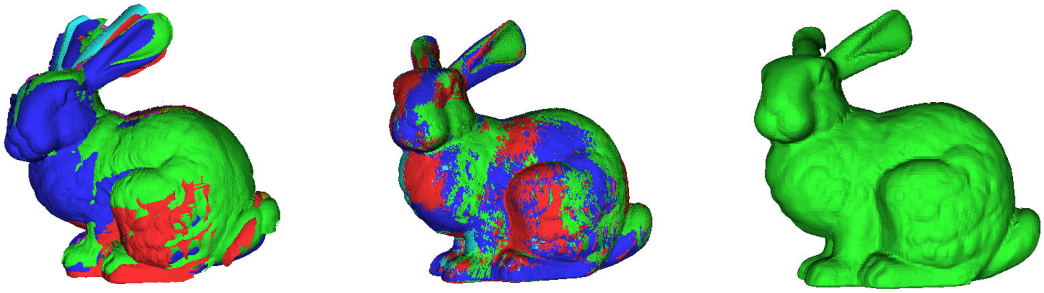
In overlapping (volumetric) regions of input surfaces, voxels can be influenced by two or more surfaces (see Figure 5). These voxels should contain the direction and distance toward the new surface which is the local average of all nearby input surfaces. In our framework, the new value of the field is the vector that is the most parallel to the vectors of the individual fields, i.e. normal to the surfaces at the closest points. This vector is a least squares estimate obtained as follows. Let $\mathbf{v}$ be a voxel within the envelope of N surfaces as in Figure 5, and let $\mathbf{c}_i$ be the closest points on each surface. The unit normal $\mathbf{n}$, i.e. the opposite direction of $\mathbf{f}(\mathbf{v})$ at $\mathbf{v}$, is obtained as the eigenvector corresponding to the largest eigenvalue of the covariance matrix:

$$C = \sum_{i=1}^{N} \frac{(\mathbf{v}-\mathbf{c}_i)(\mathbf{v}-\mathbf{c}_i)^T}{\|(\mathbf{v}-\mathbf{c}_i)\|^2} \qquad (2)$$

The distance to the reconstructed surface is obtained as the average projection of the vectors $\mathbf{v}-\mathbf{c}_i$ on $\mathbf{n}$. To be able to update the field incrementally, matrix C is stored at each voxel. Being a simple sum, matrix C can be updated for each new surface without reconsidering already processed surfaces. An example of geometric fusion and model reconstruction from partial views is shown in Figure 4.

3.3. Surface reconstruction from curves

Optical triangulation in conjunction with a projected curvilinear light (laser) pattern is commonly used to build 3D scanners, and hand-held scanners in particular since the mechanical part of the scanner is very light. These scanners



Initial range images Registered range images Reconstructed model.

Figure 4. Example of geometric fusion and registration.

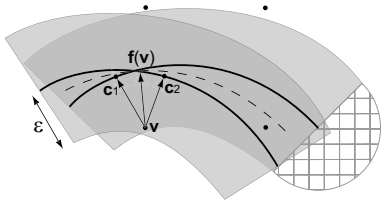


Figure 5. Geometric fusion. The vector field $\mathrm{f}(\mathrm{v})$ at point v within the envelope of multiple surfaces is obtained as a least squares estimate. The "average surface" is represented as a dashed line. Volumetric envelopes are shaded in gray. The field should not be computed in the regions (hatched area) near surface discontinuities.

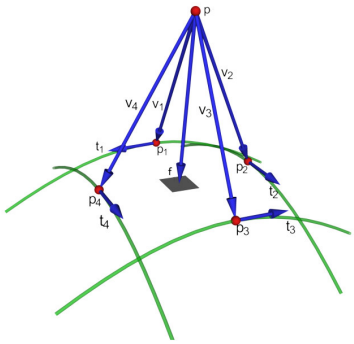


Figure 6. Reconstruction from multiple curves. The normal is obtained as a least squares estimate of the "most perpendicular" vector to the tangents $\mathbf{t}_i$, $i = 1, 2, 3, 4$ at the closest points $\mathbf{p}_i$ of the four curves.

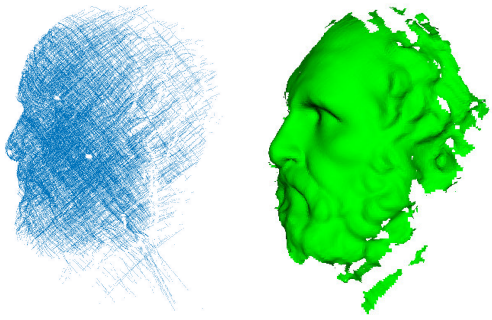
produce curves measured on the surface of an object. To reconstruct the surface, the curves have been typically considered either as unorganized sets of points or constrained only to the curves contained in parallel planes (profiles). Former approaches result in a loss of quality as explained in Section 3. In the latter approach [5], the neighbouring profiles are simply connected to form surface patches. To be able to do so, the scanner has to be moved without abrupt changes of the speed or the orientation which is very inconvenient, especially for hand-held scanners. Furthermore, surface profiles do not intersect, which implies that they cannot be registered in order to remove scanner positioning errors.

Surface reconstruction from curves is based on a fundamental theorem of differential geometry stating that the tangents of all surface curves passing through a single point are contained in the tangent plane at the same point. A corollary of this theorem is that the tangent plane can be computed at the intersection point of at least two curves if their tangents are known and are not parallel. In the case of two curves, the normal on the tangent plane is given as the vector product of the two tangents.

Building the vector field from a set of arbitrary (but not parallel) curves [13] is based on the same principle with one notable difference, that is: the tangent planes are not estimated only at the intersection points but also in their neighbourhood. At each voxel $\mathbf{v}$ the vector field is computed relative to the closest points $\mathbf{c}_i$ to all curves i within the envelope. The surface normal at the point closest to $\mathbf{v}$ is obtained as the most perpendicular vector to the tangents t_i at the points $\mathbf{c}_i$, see Figure 6. More formally, the normal is the eigenvector associated with the smallest eigenvector of the covariance matrix

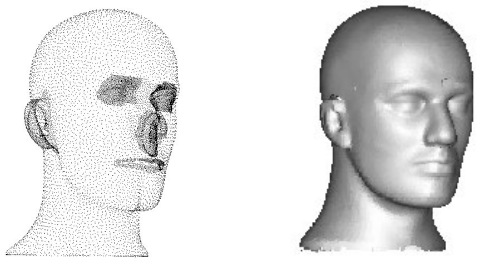
$$C = \sum_{i=1}^{N} t_i t_i^T. \tag{3}$$

If the field is to be updated incrementally, this covariance matrix is stored and updated at each voxel since it is a simple sum of covariance matrices of individual curves. In the same manner as it was done for partial surfaces, there is no need to search for closest points on curves for each voxel. If the curves are represented as sets of line segments, then a fundamental cell can be defined for each line segment [13]. Each line segment influences voxels in its cell thus resulting in linear complexity with respect to the number of points contained in curves. An example of surface reconstruction from curves is shown in Figure 7.



(a) (b)

Figure 7. Example of reconstruction from curves. a) Input data. b) Reconstructed surface.



(a) (b)

Figure 8. Example of reconstruction from an unorganized set of points [6]. a) Input data. b) Reconstructed surface.

3.4. Surface reconstruction from unorganized sets of points

Surface reconstruction from unorganized sets of points is very similar to the previous two cases. At each voxel, a covariance matrix is built from all points that are within $\epsilon > 0$ from the voxel. As before, an envelope is defined for each point, in this case a sphere. The covariance matrix is obtained as follows:

$$C = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{c}_i - \mu)(\mathbf{c}_i - \mu)^T = \frac{1}{N}\sum_{i=1}^{N}\mathbf{c}_i\mathbf{c}_i^T - \mu\mu^T \quad (4)$$

where

$$\mu = \frac{1}{N}\sum_{i=1}^{N}\mathbf{c}_i \quad (5)$$

is the centroid of closest points $\mathbf{c}_i$. Both μ and matrix C can be computed incrementally as before. The normal is obtained as the eigenvector associated with the smallest eigenvalue. An example of reconstruction from an unorganized set of points (using data from [6]) is shown in Figure 8.

3.5. Reconstruction from parallel curves

As explained above, the proposed method can only handle non-parallel curves, otherwise the covariance matrix would not have 3 non-zero eigenvalues and therefore the normal on the surface is not defined. To handle parallel curves, one can simply combine the reconstruction from the unorganized points and from curves by adding components that are not parallel to the curve tangents (but still tangent to the surface) to the covariance matrix defined in Eq. 3 (see Figure 9). These components are vectors $\mathbf{c}_i - \mu$ in Eq. 5.

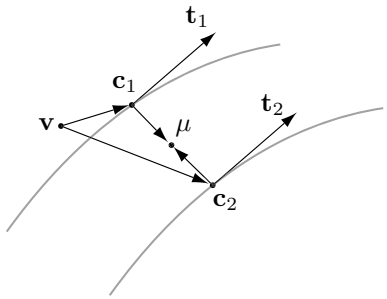


Figure 9. Reconstruction from parallel curves. If the tangents ($\mathbf{t}_1$, $\mathbf{t}_2$) are parallel the covariance matrix is not defined. Adding a nonparallel component $\mathbf{c}_i - \mu$ to the covariance matrix solves this problem.

As explained in Section 3, it is advantageous to use an orientation estimate from the connectivity of points in curves rather than from relative positions of curves. Therefore, the method in section 3.3 should be preferred for modeling unless the curves are parallel.

4. Unified reconstruction

The common ground of the reconstruction approach for the three types of 3D data is the use of covariance matrices. The only difference is the use of the largest eigenvalue for surface patches instead of the smallest as for curves and unorganized points. However, when computing the covariance matrix for a surface patch, the normal (vector $\mathbf{v} - \mathbf{c}_i$ in Eq. 2) can be used to compute two surface tangents using Gram-Schmidt orthogonalization, for instance, and the covariance matrix can be built using the two tangents in exactly the same manner as it was done for surface curves. That way the computation of the vector field is exactly the same for all types of data. This opens a new avenue for surface reconstruction from combined data. For example an initial model could be built using surface patches and then updated using curvilinear data (fixed scanner for patches, hand-held for curves).

5. Registration

When acquiring 3D data from multiple views, either the sensor or the object has to be moved. In order to merge the data from multiple views, the position of the object and/or the sensor has to be known in a single reference frame.

Since positioning devices cannot be made perfect, some registration error is inevitably introduced. This means that the data collected from multiple viewpoints is not perfectly aligned. This problem has been traditionally solved using a pose refinement (registration) algorithm usually based on the Iterated Closest Points (ICP) algorithm. ICP assumes that the data is already close to the exact position, which is suitable for this type of problem.

The registration using an ICP based principle requires finding correspondences (closest points) between the model and a rigid point set to be registered. Once correspondences are found, a rigid transformation aligning the two sets can easily be computed. The most computationally expensive step of this operation is finding closest points, which has $O(NM)$ complexity ($NlogM$ using k-d trees) with respect to N, the number of points to be registered and M, the number of points in the model. Since vector fields explicitly encode closest points, this operation becomes trivial: for each point to be matched, it is sufficient to find the closest voxel and, using the vector field value at the voxel, to compute the closest point on the model. Finding the closest voxel is also trivial since the voxels are organized on a regular 3D grid. This way, the computational complexity is linear, $O(N)$, and does not depend on the size of the model.

In general, regardless of the type of 3D data, registration can be performed as follows: first, a model is reconstructed from all available data and each rigid set of points (a curve, a surface patch or a set of points) is registered to it. Then, the model is recomputed and the data is again registered to the model. The whole procedure is repeated until convergence. Note that the model is initially distorted by registration errors but registering data to it still improves their poses. As the algorithm iterates, both model and pose of 3D data improve. An example of registration is shown in Figure 4.

Clearly, in order to allow registration, the positioning errors should be small enough so that the data that needs to be registered falls within the envelope of the model. On the other hand, increasing the size of the envelope increases computational costs considerably. Section 6 explains how to avoid additional computational cost for large registration errors.

6. Compression

The main disadvantage of volumetric representations is their large memory consumption when compared to the explicit representations, for example triangulated surfaces. Being computed on a regular grid, implicit representations always require a large number of points (voxels) even if the represented surface is a simple plane. For scalar distance fields, this is a problem usually solved by using compression schemes such as octrees or run-length encoding [3] to avoid the overhead due to the unoccupied voxels located outside the envelope. However, the compression rate depends largely on the position and the orientation of the surface within the volume. Moreover, these compression schemes do not take into account the shape of the surface, and thus even the representation of a single plane in the volume, may require a considerable amount of memory.

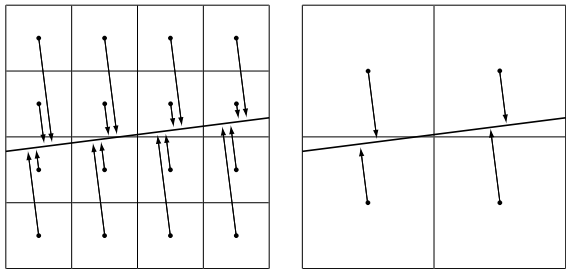


Figure 10. Principle of the compression of vector fields. In regions where the surface is planar or changes slowly (a), groups of voxels can be removed and replaced without loss by a smaller number of voxels at lower resolution (b).

As explained above, values in the vector field represent the direction and the distance to the closest points on the surface. In other words, vector fields encode tangent planes at the closest points on the surface. Therefore, if the surface is a plane, all voxels actually encode exactly the same plane and are therefore redundant. As shown in Figure 10, this allows a group of voxels in a planar region to be replaced with a smaller number of voxels at lower resolution. Reconstruction of higher resolution voxels can be performed without any loss since both lower and higher resolution voxels encode the same plane. If the surface changes shape, then the loss of information is inevitable, but the compression is still possible by imposing a maximal error produced by truncating higher level voxels. It should be noted that this compression scheme depends very little on the surface orientation or position.

Another useful property of the compression approach is that it is a simple way to allow for larger registration errors during the registration. As mentioned in section 5, the envelope should be large enough to accommodate registration errors but, increasing the size of the envelope significantly increases computational cost and should be avoided. This is solved by using lower resolution voxels since the voxels at a lower resolution level are twice as large and, consequently, the acceptable registration errors can be twice as large. As needed, the voxels at several levels below the highest resolution can be used to accommodate even larger errors.

An example of a compressed surface is shown in Figure 13 where the compression rate is 98% relative to the octree representation (which is already a compressed volume). The compression rate was 99.94% relative to the whole volumetric grid, i.e. 512x512x512 voxels.

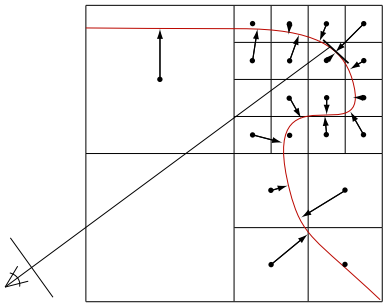


Figure 11. Ray-tracing the vector field. The normal on the surface at the intersection of the ray and the model is obtained from the vector field. The algorithm traverses the volume to reach the first voxel where the intersection of the ray with the tangent plane (encoded in the voxel) remains within voxel boundaries.

7. Visualization: ray-tracing the vector field

Although graphic hardware has advanced considerably in recent years, the maximum complexity of the rendered model is still limited. Once again, the reason is the use of polygonal (triangulated) surfaces. When rendering a triangulated surface, there is no simple way to decide which triangles are visible and which are not. This leads to a computational complexity which depends directly on the number of triangles. There is a number of techniques employed to accelerate rendering of large models, for example progressive meshes or octrees. Even though these techniques give good results, there is also significant pre-processing and implementation complexity involved while the complexity is still not constant. In order to display models of arbitrary size, the computational complexity has to be proportional only to the number of pixels in the rendered image, i.e. the complexity should be constant.

Since the vector fields encode the surface normal at points on a regular grid, the ray-tracing of the vector field is trivial. It is sufficient to traverse the volume using a variant of the Bresenham's algorithm as shown in Figure 11. The normal on the surface at the intersection of the ray and the model can be obtained as the value of the vector field at the voxel where the intersection of the ray with the tangent plane (encoded in the voxel) remains within voxel boundaries. If the field is compressed, then the voxel used will still be the first traversed voxel at highest resolution available that satisfies the above condition. Also, a variable level of rendering details can be simply achieved by choosing the appropriate level of the octree. By doing direct ray-tracing of the vector field, the computational complexity depends only on the resolution of the volume and the resolution of the rendered image. In other words, it is constant with respect to the size of the model. Examples of rendering for a compressed vector field are shown in Figures 12 and 13.

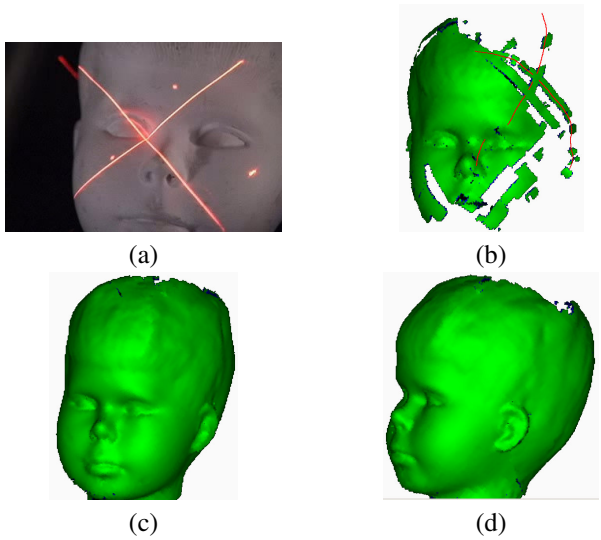


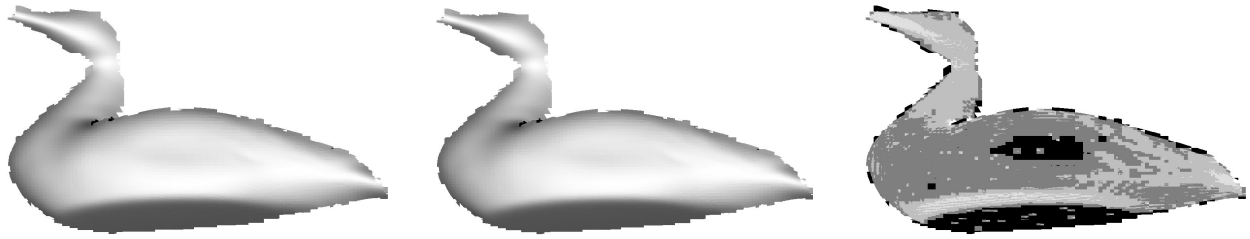
Figure 12. a) The real-time modeling system is based on a hand-held sensor that projects a laser crosshair pattern on the object. b) The motion of a crosshair pattern in 3D space produces multiple intersecting curves used to incrementally build the model. The partially reconstructed model is displayed in real-time by ray-tracing the vector field. c) and d) Final reconstructed model from two viewpoints.

Being conceptually very simple, a complete ray-tracer can be implemented in only 200 lines of C++ code. It should also be noted that scalar (signed) distance fields cannot be used in such a simple manner, since the surface normal cannot be estimated from a single voxel, but rather by using locally a variant of marching cubes. In our implementation this approach was for an order of magnitude slower than the direct ray-tracing of the vector field.

8. Implementation

The framework has been validated by constructing a real-time modeling system based on a hand-held sensor. The sensor measures two curves at a time by projecting a crosshair laser pattern on the surface of the object and reconstructs the two curves in 3D using optical triangulation (Figure 12a). The sensor displacement generates numerous intersections between the set of crosshairs, thus allowing reconstruction as described in Section 3.3. The sensor is self-referenced, that is, it computes its position and orientation by observing reference points in the scene[4].

What distinguishes this modeling system from others is its ability to incrementally reconstruct the model while displaying it in real-time. This greatly facilitates the acquisition since the user has a visual feedback. Currently, the modeling system is efficient enough to perform image pro-



(a) (b) (c)

Figure 13. Example of ray tracing for a compressed vector field. (a) Rendering of the uncompressed field. (b) Rendering of the compressed field, with a 98% compression rate. (c) Size of voxels used for rendering; darker colour indicates larger voxels. Since the flat regions contain less information, they are represented with larger voxels and thus compressed more.

cessing (detection and estimation of laser patterns), optical triangulation, sensor positioning, curve reconstruction, model reconstruction and rendering (320×240) at 15fps on a laptop PC (Pentium 4, 2.4 GHz). The octree has 9 levels. As expected from a modeling system with linear complexity, its performance does not degrade as the amount of data increases. Regardless of the number of measured points the system has a constant frame rate.

9. Conclusion

A novel framework based uniquely on vector field implicit representation has been presented as a solution for 3D modeling problems. When compared with existing methods, the proposed framework offers the possibility to implement the whole modeling chain with linear complexity, making the design of real-time acquisition and modeling systems possible. Furthermore, the model can be built from any type of 3D data in the same framework, including previously unconsidered surface curves. Although current work concerns algorithm optimization, future work will focus on the quality analysis of the recovered model (sampling and resolution).

References

[1] P. Besl and N. McKay. A method for registration of 3-d shapes. *IEEE Transactions PAMI*, 14(2):239–256, February 1992.

[2] F. Blais, M. Picard, and G. Godin. Recursive model optimization using icp and free moving 3d data acquisition. In *Proceedings of 3DIM*, pages 251–258, 2003.

[3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96 Proceedings*, pages 303–312, 1996.

[4] P. Hébert. A self-referenced hand-held range sensor. In *Proceedings of 3DIM*, pages 5–11, May 2001.

[5] A. Hilton and J. Illingworth. Geometric fusion for a hand-held 3d sensor. *Machine vision and applications*, 12:44–51, 2000.

[6] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH '92 Proceedings*, volume 26, pages 71–78, July 1992.

[7] T. Koninckx, A. Griesser, and L. Van Gool. Real-time range scanning of deformable surfaces by adaptively coded structured light. In *Proceedings of 3DIM*, pages 293–300, 2003.

[8] T. Masuda. Registration and integration of multiple range images by matching signed distance fields for object shape modeling. *Comput. Vis. Image Underst.*, 87(1/2/3):51–65, 2002.

[9] V. Popescu, E. Sacks, and G. Bahmutov. The modelcamera: a hand-held device for interactive modeling. In *Proceedings of 3DIM*, pages 285–292, 2003.

[10] G. Roth and E. Wibowo. An efficient volumetric method for building closed triangular meshes from 3-d image and point data. In *Graphics Interface*, pages 173–180, May 1997.

[11] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *proceedings of SIGGRAPH 2002*, 2002.

[12] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions PAMI*, 17(4):344–358, 1995.

[13] D. Tubić, P. Hébert, and D. Laurendeau. 3d surface modeling from curves. In *Proceedings of CVPR 2003*, pages 842–849, 2003.

[14] D. Tubić, P. Hébert, and D. Laurendeau. A volumetric approach for interactive 3d modeling. *Comput. Vis. Image Underst.*, 92:56–77, 2003.

[15] G. Turk and M. Levoy. Zippered polygon meshes from range images. *SIGGRAPH '94 Conference Proceedings*, 26:311–318, 1994.