Investigations of Tensor Voting Modeling

Joanna Beltowska†‡ Ken Museth‡ David Breen†

†Computer Science Department
Drexel University
Philadelphia, PA USA

‡Science and Technology Department
Linköping University
Norrköping, Sweden

ABSTRACT

Tensor voting (TV) is a method for inferring geometric structures from sparse, irregular and possibly noisy input. It was initially proposed by Guy and Medioni [GM96] and has been applied to several computer vision applications. TV generates a dense output field in a domain by dispersing information associated with sparse input tokens. In 3-D this implies that a surface can be generated from a set of input data, giving tensor voting a potential application in surface modeling. We study the tensor voting methodology in a modeling context by implementing a simple 3-D modeling tool. The user creates a surface from a set of points and normals. The user may interact with these tokens in order to modify the surface. We describe the results of our investigation.

Keywords: implicit surface modeling, surface reconstruction, tensor voting

1 INTRODUCTION

As the use of implicit 3D representations gains popularity, the need for modeling software that provides implicit model editing capabilities increases. New surface editing approaches are continually being explored and much research is being conducted to find ways to interactively modify implicit models. Towards these ends we investigated tensor voting as a possible technology that could provide a new and interesting approach for editing implicit models.

Tensor voting is a method for grouping geometric features [MLT00]. It can be used to generate surfaces from a sparse set of possibly noisy and irregular input data, and therefore may provide novel editing capabilities within a 3-D modeling context. The goal of our work was to investigate TV as a technique for interactive surface modeling. We were interested in determining if TV can be used to model simple objects, edit them interactively and control their shape via TV "input tokens." To achieve this, we developed a TV modeling system (TVMS) based on an already existing TV framework and conducted experiments to evalute TV in a 3-D modeling context. Spheres were used to examine the parameters of TVMS and two models were sculpted in order to get an evaluate the potential capabilities and limitations of the TV modeling process.

1.1 Related Work

We looked to other novel modeling systems for ideas and inspiration. In general, these are systems that model 3-D objects with geometry other than polygonal meshes. The Vector Field Based Shape Deformations of von Funck et al. [vFTS06, vFTS07] is an interactive method for shape modeling with volume preservation. Guo et al.[GHQ04] use scalar–fields to drive a point-set surface editing process. The point-sample based shape modeling presented by Pauly, Keiser, Kobbelt and Gross [PKKG02] is a hybrid system which couples the benefits of implicit and parametric surface representations, using the implicit surface definition of moving least squares projection along with point-based visualization. Pointshop 3D is a framework developed by Zwicker et al. [ZPKG02] that transfers the functionality of 2D image editing operations to 3D, Dewaele and Cani [DC04] present a virtual clay modeling approach, which implements interactive shape modeling capabilities which may be based on haptic feedback. Museth et al. [MBWB02, MBW$^+$05] developed interactive editing operators for a relatively new type of implicit surface, level set models.

In our approach, as in many of the approaches described above, the object is represented using control points, which can be modified by the user in order to edit the surface. By using the TV methodology in a novel context, we have employed the ideas of local and global editing operators, point-based surface representation and a physical analogy for modeling surfaces.

2 TENSOR VOTING

This section provides an introduction to the Tensor Voting methodology. More details about the methodology can be found in [GM97, TM98, MLT00, TMMS01, TTMM04].

TV has its background in early computer vision problems where the available data often is sparse and noisy, making it difficult to extract relevant information and structures. TV identifies local feature descriptions by spreading the information associated with shape-related input within a neighborhood while enforcing a smoothness constraint. This process refines the information and accentuates local features. By doing so, coherent, locally smooth, geometric features are defined and noise is discarded. Each data point communicates its information in a neighborhood through a voting process. The more information that is received at each data point, the stronger is the likelihood of a geometric feature being present at a certain location. This likelihood is expressed through a confidence measure, saliency, which is used in the feature extraction process.

TV is based on two elements; data representation, which is obtained by means of tensor calculus, and communication of data through linear voting, a process similar to linear convolution. The input elements, referred to as input tokens, are encoded into tensor form and communicate their information to their neighboring tokens via pre-calculated tensor voting fields. After this initial voting step, each token has its confidence and surface orientation encoded into a generic second order symmetric tensor. The tokens vote a second time to propagate their information throughout a neighborhood. The result is a dense tensor field which assigns a measure of confidence and saliency to each point in the domain. This dense map is decomposed into three dense maps, each representing a geometric feature (junction, curves or surface), which are analyzed during feature extraction. TV can be generalized to N-D [TMMS01]. The 3-D case, specifically surface voting, is sufficient for our needs and will therefore be the focus of this paper.

2.1 Tensor Representation

Diagonalizing a second order symmetric tensor, which can be represented by a 3×3 matrix, produces the associated characteristic equation. Solving this equation leads to a representation based on the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ (in descreasing order) and the associated eigenvectors $\hat{e_1}, \hat{e_2}, \hat{e_3}$ of the tensor. A second order symmetric tensor may be graphically represented as an ellipse in 2-D or an ellipsoid in 3-D. The eigenvalues describe the general size and shape of the ellipsoid and the eigenvectors describe its principal directions. Because of the properties of second order symmetric tensors, the eigenvalues are real and positive, or zero, and the eigenvectors form an orthonormal basis. The tensor can be decomposed into three components defined by

$$\begin{aligned} \mathbf{T} = (\lambda_1 - \lambda_2)\hat{e_1}\hat{e_1}^T + \\ (\lambda_2 - \lambda_3)(\hat{e_1}\hat{e_1}^T + \hat{e_2}\hat{e_2}^T) + \\ \lambda_3(\hat{e_1}\hat{e_1}^T + \hat{e_2}\hat{e_2}^T + \hat{e_3}\hat{e_3}^T). \end{aligned} \quad (1)$$

The first term corresponds to a *3-D stick tensor*, which implies a surface patch with normal $\hat{e_1}$. The second term corresponds to a *plate tensor* and implies a curve or surface intersection with tangent $\hat{e_3}$ that is perpendicular to the plane defined by $\hat{e_1}$ and $\hat{e_2}$. The last term corresponds to a *3-D ball tensor* and implies a structure with no orientation preference.

2.2 Tensor Communication

The voting process is similar to convolution with the difference that convolution produces scalar values and tensor voting produces tensors. Voting kernels which encode certain constraints such as smoothness and proximity are used. These voting kernels are continuous tensor fields which assign a value to every point within the domain. Any voting kernel, regardless of dimension, can be derived from the 2-D stick tensor, which is therefore referred to as the fundamental 2-D stick voting field (VF) and is presented in Figure 1 (left). For its derivation, see [MLT00] and [TMMS01]. Given an input point and normal at O, the most likely normal at P to a curve passing through O and P is defined by the osculating circle connecting O and P, because it keeps curvature constant. The 3-D VF is produced by rotating the 2-D VF around N.

The magnitude of the field is described by a decay function, which is expressed in spherical coordinates in Equation 2 and is presented in Figure 1 (right),

$$DF(\gamma, \varphi, \sigma) = e^{(\frac{\gamma^2 + c\varphi^2}{\sigma^2})}. \quad (2)$$

γ is the arc length of the curve OP, φ the curvature, c the curvature scale factor and σ the scale of analysis. c provides additional control on the influence of curvature. σ determines the size of the voting neighborhood and is the only free parameter adjusted by a user. In practice the range of influence of a particular VF is eliminated once its magnitude drops below some value, e.g. 1%. Therefore the value of σ determines the extent of the voting field.

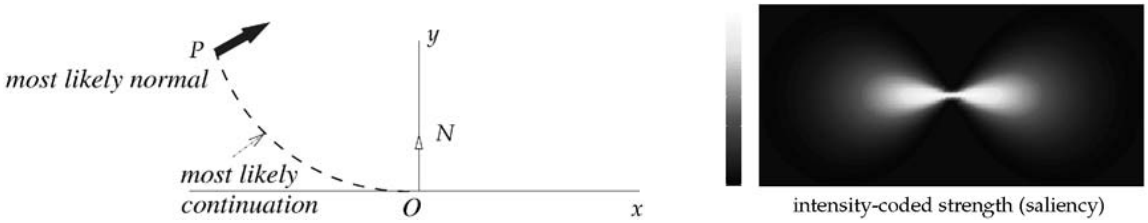

Figure 1: (left) The design of fundamental 2D stick voting field. (right) Magnitude (saliency) of the fundamental 2D stick voting field. From [TMMS01]

The field has an analogy with particle physics, motivating the choice of a Gaussian function for the energy decay function. The voting kernel shows the same behavior as an anisotropic energy field, where an emitting particle P_s, radiates energy in all directions and a receiving particle P_r receives energy from all directions, but with a preference for a straight path. Therefore received energy is inversely proportional to distance traveled and curvature, producing a loss of energy along curved paths.

After voting, the saliency tensor $\mathbf{T}(i,j,k)$ at each location (i,j,k) is the tensor sum of the contributions from each voting token within range,

$$\mathbf{T}(i,j,k) = \sum_{m,n,o} V(\mathbf{S}(m,n,o), ((i,j,k)-(m,n,o))) \quad (m,n,o) \in N(i,j,k), \quad (3)$$

where $N(i,j,k)$ returns the neighboring locations of (i,j,k) and V is a function that returns a tensor field corresponding to the ball, stick and surface voting kernel components of $\mathbf{S}$ at (m,n,o) using the vector $(i,j,k)-(m,n,o)$, which is expressed in the $(\hat{e_1},\hat{e_2},\hat{e_3})$ coordinate system of $\mathbf{S}$.

2.3 Feature Extraction

After the voting process, each point in the domain has been assigned a second order symmetric tensor that estimates the structure of the feature type(s) and the associated saliency. Thus, information is collected at each location in the domain, building a saliency map for each feature type. Salient features can be located by finding local extrema in this dense tensor map, which in 3-D is decomposed into three dense vector maps, expressing the occurance of junctions, curves and surfaces in the domain. For each feature type, the dense tensor map can be broken into a scalar field, expressing the saliency (strength) of the tensor field at any given point, and a vector field, expressing the direction of the feature corresponding to the saliency value. Thus, at any given location there is a scalar and a unit vector $< s, \hat{n} >$ present. These dense saliency and vector maps are then used for vote interpretation.

Since we are only interested in the surface features produced by tensor voting, we only examine the the Surface Map $< s, \hat{n} >$ extracted from the tensor field. For this case,

$$s = (\lambda_1 - \lambda_2) \qquad \hat{n} = \hat{e_1}, \quad (4)$$

where s is the surface saliency and $\hat{n}$ is the normal to the most likely surface at each point in space.

The most likely surface produced by a set of input tokens (points and normals) is an *extremal surface* embedded in the Surface Map. A point is on the extremal surface when the saliency s is extremal in the normal direction $\hat{n}$ at that point, i.e. $ds/d\hat{n} = 0$. Reformulating this equation, the surface may be found by identifying zeros of the the scalar field $\hat{n} \cdot \nabla s$, defined as g field. For practical reasons the Surface Map is calculated on a regular grid and the zero crossings of the extremal surface are identified along grid lines.

3 TENSOR VOTING MODELING SYSTEM

The tensor voting modeling system, TVMS, is based on the TV3D framework for tensor voting developed by the USC Computer Vision Group[1], and uses QSplat[2] [RL00] for point-based visualization. It has been developed on a MacBook Pro with a 2.2 GHz Intel Core 2 Duo processor. Functionality has been added to both frameworks and only surface voting is used from the TV3D framework. Since our goal is to extract a surface from a given input, we have omited the point and curve voting functionality of TV3D.

TV3D uses an input of oriented or unoriented points, curvels and surfels. In TVMS, the input is restricted to points with associated normals. While a surface may be constructed from unoriented points only, we have chosen to include direction information in the input data set. The reason for this

[1] http://www.cs.ust.hk/∼cstws/research/TensorVoting3D
[2] http://graphics.stanford.edu/software/qsplat

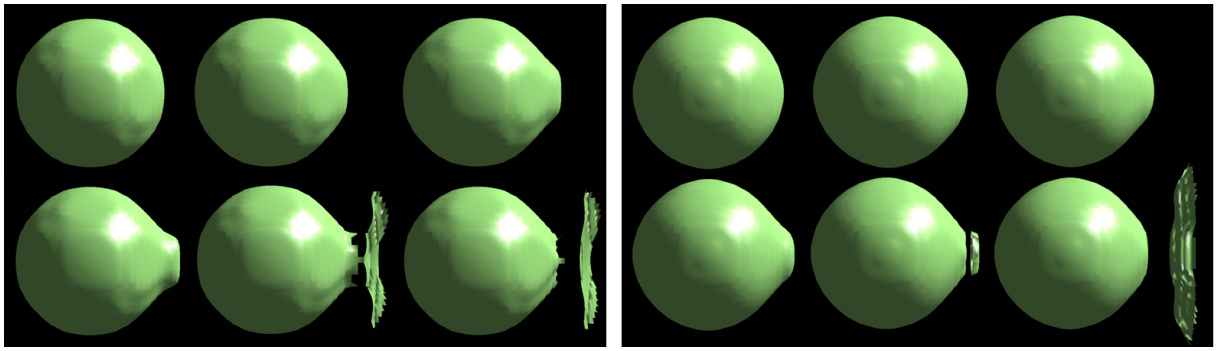

Figure 2: Translating the right-most token one unit at a time. Left: $\sigma = 7$. Right: $\sigma = 10$.

is that it gives more accurate results, allows us to omit the first voting step, which estimates and assigns a direction to all unoriented input tokens, and finally, it provides the user with an additional tool for manipulating the resulting surface.

These points can be translated and their corresponding normals rotated, as well as added to and removed from the scene. Translation is done by first rotating the normal of a selected point and then translating the point along the normal. This is a simple yet intuitive interaction that meets our needs. The user can either create an entirely new model on his or her own, or read an existing model from a file. A new point is inserted by clicking between two existing points; the new point is inserted at the position halfway between the two points closest to the user's mouse click. Each input token has a position, a normal, a weight and a local voting neighborhood size, which can be modified by the user.

Once the user is satisfied with the initial configuration of input tokens, (s)he selects an initial global sigma value and runs the tensor voting process to extract a surface. The surface is quickly rendered as points by default, but, given more time, the user can also render the surface as a mesh.

3.1 Modeling Parameters

The original TV framework only has one free parameter, namely σ, which is a coefficient in the saliency decay function and sets the scale of analysis. σ is a positive scalar which determines the size of the voting fields used in the voting process and is set with an initial global value. Having a single global σ value will create a model with a one level of detail. In TVMS, σ has been divided into two parameters, a local σ, σ_L which is assigned individually for each token and a global σ, σ_G, which is assigned to all tokens which have not had their local sigma value changed. By default, σ_L is set to -1 so that the system can distinguish modified tokens from unmodified. Instead of a single global voting field, up to n, where n is the number of input tokens, individual voting fields can be used to vote for a surface. This allow for the level of detail of the surface to be varied by using a smaller σ_L in those regions where the user desires finer details.

A weight has been added as a third modeling parameter. The weight is a scaler value associated with each input token. The token's normal is multiplied by the weight before voting and surface extraction. Increasing the weight value strengthens the influence of the plane defined by the token; thus flattening the resulting surface near the token. The default value is set to 1.0.

4 RESULTS

In order to investigate the geometric modeling potential of tensor voting we conducted a number of experiments. Our experiments were performed in two parts. In the first part, we looked at the local influence of the individual modeling parameters on a simple shape. In the second part, we examined overall effectiveness of the TV modeling process when attempting to model two specific shapes.

4.1 Influence of Modeling Parameters

The scale of analysis parameter σ affects the radius of the model and the cost of calculating a surface. Because we wish to extract the surfaces as quickly as possible, the size of the voting fields, and thus the size of the models, have to be kept small. We found that the radius of our models, in units of voxels, should be approximately the same as σ. Having a value of σ between 8 and 10 produced results in a reasonable amount of time, which implied a model radius of slightly more than 10 units. Using larger σ values would require bigger models, thus a higher resolution computational grid, and would

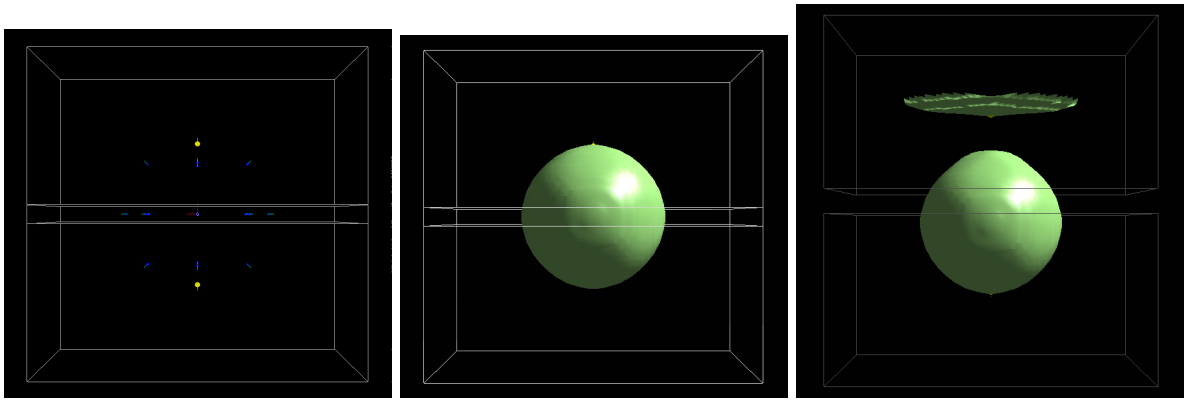

Figure 3: Left: The 18 input tokens needed to define a sphere. The top and bottom tokens have been highlighted and the extent of their voting fields ($\sigma_G = 10$) have been displayed. Center: The resulting sphere. Right: The extent of the voting fields for the top and bottom points, after point translation. The two fields no longer overlap and a disjoint surface has been formed.

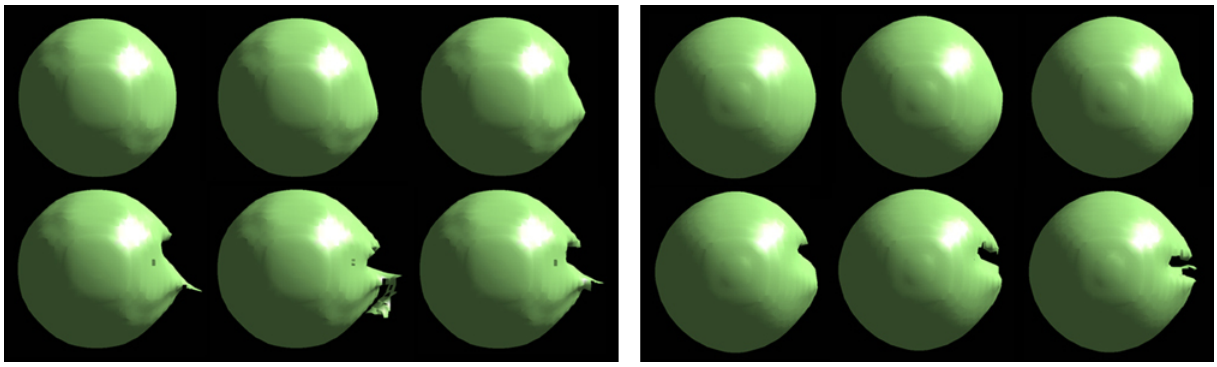

Figure 4: Rotating the normal of the right-most point and voting with $\sigma_G = 10$. Starting at the top left, the normal is rotated $0°$, $15°$, $30°$, $45°$, $60°$ and $75°$.

require significantly more time in the surface extraction step. Also, σ should not be set too large, since this causes votes to cancel each other out [MLT00] and produce surface holes, further motivating the use of small σ values.

The model used in our parameter influence experiments is a sphere with a radius of 12.5 units, that was defined by 18 input tokens. See Figure 3 (left).

To demonstrate the effect of translating input tokens, we translate one single token on the sphere, namely the right-most one, one unit at a time. The results are presented in Figure 2, which consists of spheres produced with $\sigma = 7$ and $\sigma = 10$. It can be seen that the surface does begin to bulge out to fit to the surface patch implied by the translated token. But if the token is translated too far the surface breaks up.

Insight into this unwanted artefact is provided in Figure 3. Here, the extents of two of the sphere's 18 tokens are displayed. As the top token is translated away from the others the two displayed extents no longer overlap. In this situation two disjoint surfaces are produced. The TV process interprets the translated token as an isolated point and tries to infer a separate plane from it. This clearly shows that in order to produce a closed mesh, one needs to ensure that there is significant overlap between the voting fields of neighboring tokens.

Figure 2 also highlights the effect of σ on the surface. On the left, with $\sigma = 7$, the resulting surface has flatter, somewhat sharper, features. When σ is increased to 10, more voting fields overlap because the are larger. The increased number of contributing VF's at any location creates a smoothing effect on the surface. This is most evident in the top-right and bottom-left surfaces in each of the examples. As the right-most token is translated, the resulting bump has a smoother transition to the sphere when σ is greater.

To demonstrate the effect of rotating normals, Figure 4 presents the changes to the sphere produced by rotating the normal of the right-most to-

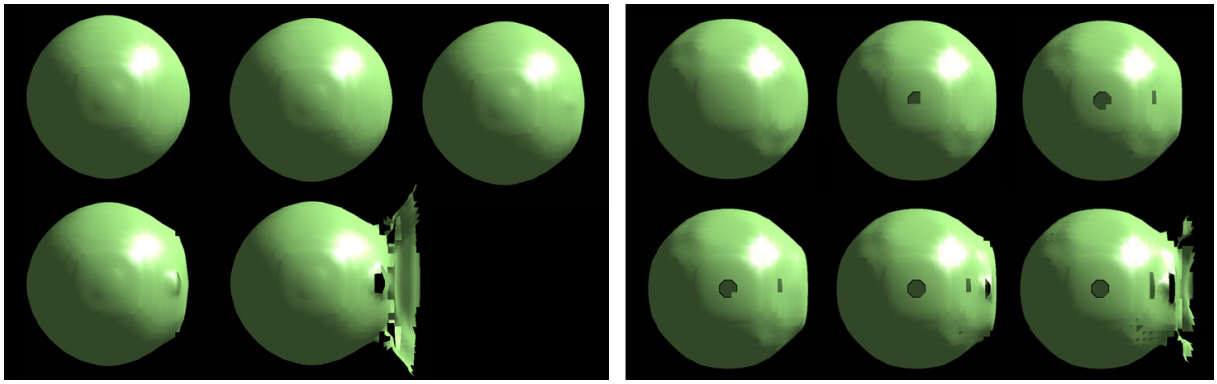

Figure 5: To the left: Incrementing the weight while voting with $\sigma_G = 10$. Weight values, from left to right, are 1.0, 1.5, 2.0, 2.5, 3.0 and 3.5. To the right: Incrementing the weight while voting with $\sigma_G = 7$. Weight values, from left to right, are 1.0, 2.0, 3.0, 4.0, 5.0 and 6.0.

.

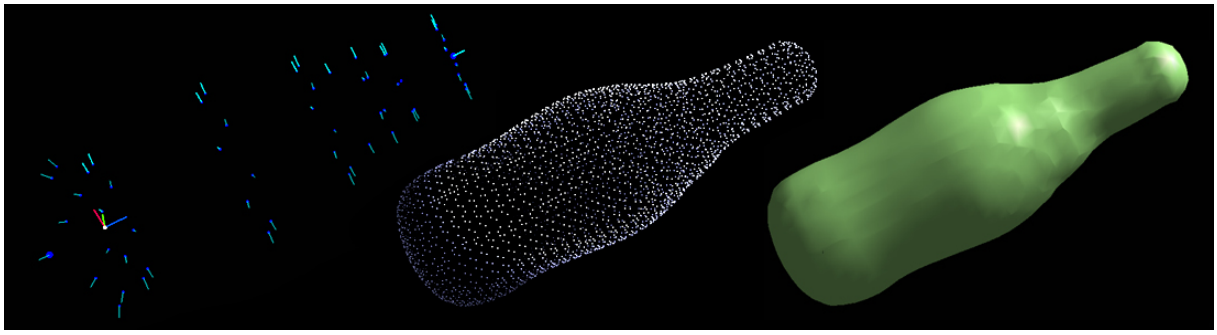

Figure 6: The bottle model created from 55 tokens. Left: Input tokens. Center: Resulting TV surface displayed as points. Right: TV surface displayed as a mesh.

ken. It can be seen that applying small rotations (angles less than 45°) to a token can be easily accommodated by the tensor modeling system. The surface smoothly blends in the disoriented surface patch implied by the rotated token. As the rotation angles increases past 45° the surface breaks up. Using a higher σ gives slightly better results, but this still cannot resolve the conflicting input information that implies that a piece of surface is directed into the sphere.

The weight parameter provides another way to modify the influence of a token. The weight parameter may either enlongate or shorten the length of the normal $\hat{n}$ associated with a token after the voting process. Given a large weight the plane associated with the token's normal has greater influence on the surface; thus locally attracting and flattening the surface, as seen in Figure 5. Tokens with small weights lose some of their influence and nearby regions are smoothed by the other tokens. Also, as a weight is lowered, the nearby surface region becomes less sensitive to changes in the weight. This is probably due to the increased influence of neighboring tokens.

4.2 Creating Complete Models

To explore the overall tensor voting modeling process, two complete models were created, a bottle and a car. Both are fairly simple objects, but have geometric properties, such as varying curvature and level of detail, that potentially could reveal any weaknesses or problems with our TV-modeling system. Both models were created by initially defining the tokens needed to produce the basic shape. The tokens were then modified to add details or fix problem. To change the curvature of the surfaces, a number of normals were rotated. To flatten regions of the surfaces, weights of nearby tokens were increased. To fill holes, new tokens were added or σ_L of the neighboring tokens were increased. For the car, the body was first created. The four wheels were then added as details later.

The bottle model, defined with 55 tokens, is presented in Figure 6. The TV3D system evaluated its

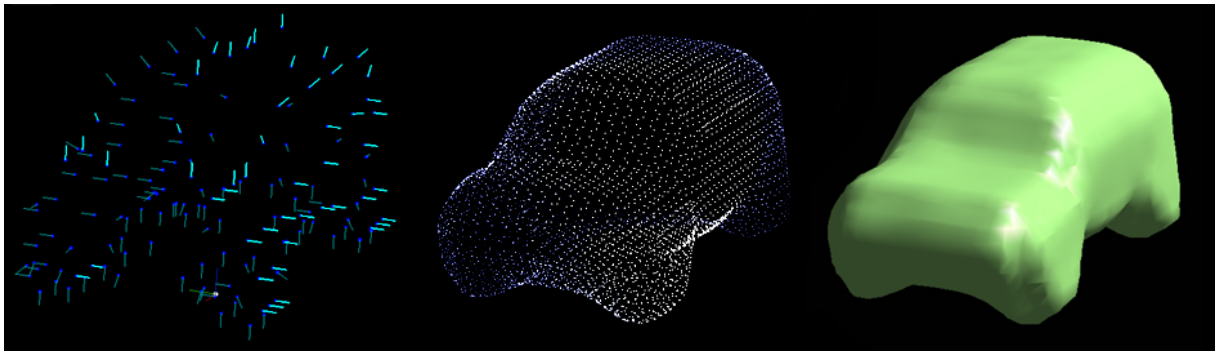

Figure 7: The car model created from 149 tokens. Left: Input tokens. Center: Resulting TV surface displayed as points. Right: TV surface displayed as a mesh.

Surface Map on X x X x X grid in FOO cpu-seconds. The resulting surface may be displayed with points. These are the zero-crossings of the computational grid used to calculate the associated g field. The Marching Cubes algorithm [LC87] may also be applied to the g field grid to produce a mesh. The car model, defined with 149 tokens, is presented in Figure 7. The TV3D system evaluated its Surface Map on X x X x X grid in FOO cpu-seconds. The resulting surface is displayed as points and a mesh.

5 DISCUSSION

During the modeling process, a significant problem quickly emerged. Small, densely sampled areas, may produce asymmetric results, which we believe is due to the TV3D surface extraction implementation. The algorithm begins at the most salient input token and grows a surface from it until the saliency of the neighborhood drops below a certain threshold. It then goes to the second most salient input point and so on. For the bottle, the asymmetry can be seen on the neck, which is not perfectly symmetric. When modeling the car, the asymmetry problem imposed serious problems shaping the more finely detailed areas. After creating the body, one wheel at a time was added. When the second wheel was added, its points influenced the points of the first wheel, changing its shape. We changed our strategy and modeled a full half of the car in one go, which was then duplicated, mirrored and merged with the first half. The resulting mesh was not closed and after repairing one hole, new holes emerged on other parts of the model. The surface extraction of TV3D appears to propagate the influence of the tokens forward as they are processed; thus the order in which tokens are processed affects the final extracted surface. The current TV3D implementation is elegant, but may not be the correct computational engine for an interactive modeling application.

The local influence of an individual input token on the final surface is not as strong nor with the type of properties as had been desired. Changing the parameters of a particular input token will in most cases render a change in the extracted surface, but only if the parameters remain within a certain range. For instance, if a token is translated far away from the other tokens, it will lose its influence on its neighbors and will produce a surface separated from the rest of the model. Also, the influence of a single token token decreases exponentially with its distance. As a token is moved away from the remaining tokens, the influence of its neighbors increases and they smooth out the contribution of the moved token. Determing these acceptable parameter ranges and including them as constraints will be a neccessary part of a future TVMS.

Surface extraction and voting is costly. Each time the model is modified and the surface must be updated, TVMS recalculates the entire surface instead of just the part that will be affected by the change. If the system could re-vote only the modified input tokens and update just the affected areas of the surface, surface regeneration could possibly be done in real-time. This will require a redesign of the surface extraction algorithm.

6 CONCLUSION

This work has examined the tensor voting method, a technique for geometric feature grouping, in a 3D-modeling context. The new system, which employs the TV3D framework for tensor voting and the point-rendering functionality of QSplat for visualization, extracts surfaces from sparse input data, consisting of points and normals. These points and normals can be interactively edited by the user to change the resulting surface. The initial

goal of modeling simple objects using TV has been achieved and two new modeling parameters, a local σ, σ_L, and a weight w, have been introduced and implemented.

Though the approach is promising, several limitations have been found in our current system. There is an asymmetry in the extracted surfaces, probably due to the surface extraction implementation in TV3D. Modeling parameter values may be easily set to produce unstable and unwanted results. Voting and surface extraction is slow for large models and large scales of analysis. The curve and junction capabilities of tensor voting should be explored for modeling sharp features and fine details. Each of these limitations should be addressed for future versions of a TV-based modeling system.

Acknowledgements We would like to thank Wai-Shun Tong who provided us with the tensor voting library and much technical assistance, and Christoffer Westberg for programming our initial point rendering software.

References

[DC04] G. Dewaele and M.-P. Cani. Interactive global and local deformations for virtual clay. *Graphical Models*, 66(6):352–369, 2004.

[GHQ04] X. Guo, J. Hua, and H. Qin. Scalar-function-driven editing on point set surfaces. *IEEE Computer Graphcs & Applications.*, 24(4):43–52, 2004.

[GM96] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1-2):113–133, 1996.

[GM97] G. Guy and G. Medioni. Inference of surfaces, 3D curves, and junctions from sparse, noisy 3D data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1265–1277, 1997.

[LC87] W.E. Lorensen and H.E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *Proc. SIGGRAPH*, pages 163–169, July 1987.

[MBW+05] K. Museth, D. Breen, R. Whitaker, S. Mauch, and D. Johnson. Algorithms for interactive editing of level set models. *Computer Graphics Forum*, 24(4):821–841, 2005.

[MBWB02] K. Museth, D. Breen, R. Whitaker, and A. Barr. Level set surface editing operators. In *Proc. SIGGRAPH '02*, pages 330–338, 2002.

[MLT00] G. Medioni, M.-S. Lee, and C.-K. Tang. *Computational Framework for Segmentation and Grouping.* Elsevier Science Inc., New York, NY, USA, 2000.

[PKKG02] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *Proc. SIGGRAPH '02*, pages 322–329, 2002.

[RL00] S. Rusinkiewicz and M. Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *Proc. SIGGRAPH '00*, pages 343–352, 2000.

[TM98] C-K. Tang and G. Medioni. Inference of integrated surface, curve, and junction descriptions from sparse 3D data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1206–1223, 1998.

[TMMS01] C.-K. Tang, G. Medioni, and M.-S.Lee. N-dimensional tensor voting and application to epipolar geometry estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):829–844, 2001.

[TTMM04] W.-S. Tong, C.-K. Tang, P. Mordohai, and G Medioni. First order augmentation to tensor voting for boundary inference and multiscale analysis in 3D. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):594–611, 2004.

[vFTS06] W. von Funck, H. Theisel, and H.-P. Seidel. Vector field based shape deformations. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3):1118–1125, 2006.

[vFTS07] W. von Funck, H. Theisel, and H.-P. Seidel. Explicit control of vector field based shape deformations. In *Proc. Pacific Graphics*, 2007.

[ZPKG02] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3D: an interactive system for point-based surface editing. In *Proc. SIGGRAPH '02*, pages 322–329, 2002.