Grouping •, –, →, ⊖, into regions, curves, and junctions

Abstract

We address the problem of extracting segmented, structured information from noisy data obtained through local processing of images. A unified computational framework is developed for the inference of multiple salient structures such as junctions, curves, regions, and surfaces from any combinations of points, curve elements and surface patch elements inputs in 2-D and 3-D. The methodology is grounded in two elements: tensor calculus for representation, and non-linear voting for data communication. Each input site communicates its information (a tensor) to its neighborhood through a predefined (tensor) field, and therefore cast a (tensor) vote. Each site collects all the votes casted at its location and encodes them into a new tensor. A local, parallel routine such as a modified marching cube/square process then simultaneously detects junctions, curves, regions, and surfaces. The proposed method is non-iterative, requires no initial guess or thresholding, can handle the presence of multiple curves, regions and surfaces in a large amount of noise while still preserves discontinuities, and the only free parameter is scale.

Keywords: shape inference, perceptual grouping, robust estimation

1 Introduction

In computer vision, we often face the problem of identifying salient and structured information in a noisy data set. From greyscale images, edges are extracted by first detecting subtle local changes in intensity and then linking locations based on the noisy signal responses. From binocular images, surfaces are inferred by first obtaining depth hypotheses for points and/or edges using local correlation measures and then selecting and interpolating appropriate values for all points in the images. Similarly, in image sequence analysis, the estimation of motion and shape starts with local measurements of feature correspondences which give noisy data for the subsequence computation of scene information. As computer vision systems move from controlled laboratory settings to real applications, the need for robust techniques for inferring salient structures becomes more apparent. In this paper, we present a unified methodology for the robust inference of multiple salient structures such as junctions, curves, regions, and surfaces from any combination of points, curve elements and surface patch elements inputs in 2-D and 3-D.

For a salient structure estimator to be useful in computer vision, it has to able to handle the presence of multiple structures, and the interaction between them, in noisy, irregularly clustered data set. It is therefore necessary to address the issues of interpolation, discontinuity detection and outlier identification when designing such an estimator. Most previous works on region inference from 2-D dot clusters[10], curve inference in 2-D[1][7] and surface reconstruction in 3-D[8] take either the fit-and-split or the fit-and-merge approach, attacking the 3 subproblems in a sequential manner. These divide-and-conquer approaches result in iterative algorithms which often require, and are sensitive to, initial guesses.

Recently, Guy and Medioni[3] have introduced a salient structure estimation approach that is able to handle the tasks of interpolation, discontinuity detection and outlier identification simultaneously. Applying their methodology to the inference of multiple curves in 2-D and surfaces in 3-D from sparse noisy data, they have produced impressive results, mostly on synthetic data. The strength of their method comes from the fact that, while interpolation can be accomplished by collecting orientation information in a local neighborhood, discontinuity detection and outlier identification can be realized as well by measuring the confidence (or saliency) and the consistency of the orientation estimations obtained from the neighbors. They have introduced a non-iterative technique called vector voting to gather orientation estimation in a local neighborhood. These orientation estimations are generated by rules easily captured by vector fields, allowing an efficient convolution-like implementation for data communication. The novelty of their method lies on the use of second order moments in accumulating and interpreting the vector votes, resulting in a method that can handle large amounts of noise, while preserving discontinuities, requires no initial guess or thresholding, and leaves scale as the only free parameter.

In this paper, we significantly extend their vector voting technique in a number of ways: by formulating their vector voting scheme in a tensorial framework, we augment their method with a mathematical foundation which accounts for many heuristic measures in their approach. The tensorial framework also provides a unified representation for various geometric features including points, curve elements and surface patch elements, which enables us to devise a tensor voting scheme to handle different instances of shape inference uniformly. In order to handle region inference, we incorporate ***polarity*** into the tensorial framework. Equipped with tensor voting, we develop a salient structure inference engine which combines the robustness and the efficiency of the original method and the representational power of tensors.

In this paper, we present in section 2 our unifying tensorial framework for salient structure inference. We then discuss the inference of curves and surfaces in section 3. Inference of regions by incorporating polarity into our salient structure inference engine is presented in section 4. We conclude with a discussion on future work in section 5.

2 Tensorial framework for salient structure inference

Inspired by Guy and Medioni's work[3], we attempt to derive a general salient structure estimator which interpolates, detects discontinuities, and identifies outliers ***simultaneously***. We introduce the use of a construct called ***saliency tensor*** which is able to signal the presence of a salient structure, or a discontinuity, or a outlier at any location. The design of the saliency tensor is based on the observation that a discontinuity is indicated by the disagreement in surface or curve orientation estimation. As orientations are represented by vectors, it is therefore the *variations* in the vector estimation that signify a discontinuity, which obviously cannot be represented by a vector. An appropriate choice hence is a (second order symmetric) ***tensor***, which is commonly used for capturing variations of direction in the study of dynamics, elasticity, fluids, and electromagnetic, among others. On the other hand, outliers can be detected by insufficient support in the estimation of surface or curve orientation and therefore can be detected by the number of the consistent votes, which we called saliency. This orthogonality of feature property and the confidence (or saliency) of the estimation has been noticed by Knutsson *et al.* [2], who have introduced the used of tensors to the estimation of local structure and orientation in the context of signal processing for computer vision. Combining the efficiency of voting with the strength of tensor representation, we develop a salient structure inference engine which make use of tensor voting to extract salient junctions, curves, regions, and surfaces from noisy data.

In the following, we first outline the mechanism of our salient structure inference engine in section 2.1. We then define the saliency tensor in section 2.2. The details of tensor voting and voting interpretation are given in section 2.3. We conclude this section with a discussion on performance in section 3.1. In this paper, scalars are denoted by italic letters, e.g. l, tensors are denoted by bold capital letters, e.g. $\boldsymbol{T}$, vectors are denoted by bold lower-case letters, e.g. $\boldsymbol{e}$. The unit length vector for $\boldsymbol{e}$ is denoted as $\hat{\boldsymbol{e}}$.

2.1 Overview of the inference engine

Our salient structure inference engine is a non-iterative procedure that makes use of tensor voting to infer salient geometric structures from local features. Figure 1 illustrates the mechanisms of this engine. In practice, the domain space is digitized into discrete cells, which are pixels in 2-D and voxels in 3-D. We use a construct call a saliency tensor, to be defined in section 2.2, to encode the saliency of various surface features such as points, curve elements and surface path elements. Local feature detectors provide an initial local measure of surface features. The engine then encodes the input into a sparse saliency tensor field. Saliency inference is achieved by allowing the elements in the tensor field to communicate through voting. A voting function specifies the voter's estimation of local orientation/discontinuity and its saliency regarding the collecting site, and is represented by tensor fields. The vote accumulator uses the information collected to update the saliency tensor at locations with features and establish those at previously empty locations. From the densified saliency tensor field, the vote interpreter produces saliency maps for various features from which salient structures can be extracted. By taking into account the different properties of different features, the integrator locally combines the features into a consistent salient structure. Once the generic saliency inference engine is defined, it can be used to perform many tasks, simply by changing the voting function. In the following, we focus our discussion on the use of tensor in salient structure inference. The process of structure integration is the same as that used in Guy and Medioni's method, which can be found in [9].

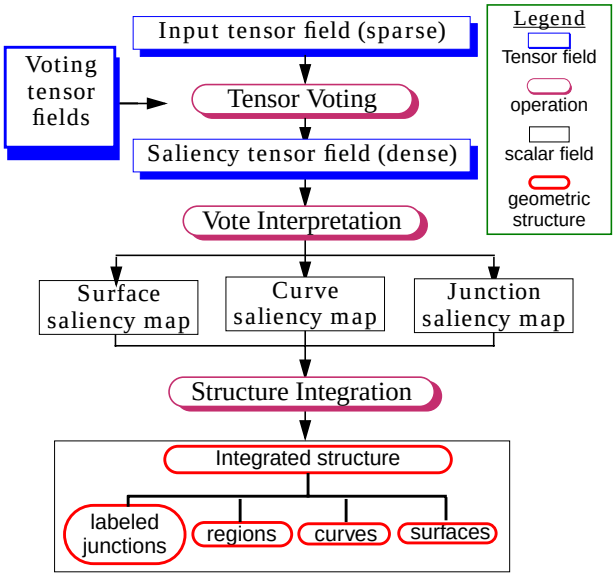


Figure 1 Flowchart of the Salient Structure Inference Engine

2.2 Saliency Tensor

The design of the saliency tensor is based on the observation that discontinuity is indicated by the variation of orientation when estimating smooth structure such as surface or curve locally. Intuitively, the variation of orientation can

be captured by an ellipsoid in 3-D, and an ellipse in 2-D. Note that it is only the *shape* of the ellipsoid/ellipse that describe the orientation variation, we can therefore encode saliency, i.e., the confidence in the estimation, as the *size* of the ellipsoid/ellipse. One way to represent an ellipsoid/ellipse is to use the distribution of points on the ellipsoid/ellipse as described by the covariance matrix. By decomposing a covariance matrix $\boldsymbol{T}$ into its eigenvalues λ_1, λ_2, λ_3 and eigenvectors $\hat{\boldsymbol{e}}_1$, $\hat{\boldsymbol{e}}_2$, $\hat{\boldsymbol{e}}_3$, we can rewrite $\boldsymbol{T}$ as:

$$\boldsymbol{T} = \begin{bmatrix}\hat{\boldsymbol{e}}_1 & \hat{\boldsymbol{e}}_2 & \hat{\boldsymbol{e}}_3\end{bmatrix}\begin{bmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{e}}_1^T\\ \hat{\boldsymbol{e}}_2^T\\ \hat{\boldsymbol{e}}_3^T\end{bmatrix} \qquad (2.1)$$

Thus, $\boldsymbol{T} = \lambda_1\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T+\lambda_2\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T+\lambda_3\hat{\boldsymbol{e}}_3\hat{\boldsymbol{e}}_3^T$ where $\lambda_1\geq\lambda_2\geq\lambda_3$ and $\hat{\boldsymbol{e}}_1$, $\hat{\boldsymbol{e}}_2$, $\hat{\boldsymbol{e}}_3$ are the eigenvectors correspond to λ_1, λ_2, λ_3 respectively. The eigenvectors correspond to the principal directions of the ellipsoid/ellipse and the eigenvalues encode the size and shape of the ellipsoid/ellipse. $\boldsymbol{T}$ is a *linear* combination of outer product tensors and, therefore a tensor.

Every saliency tensor thus has 6 parameters, λ_1, λ_2, λ_3, θ,ϕ, and ψ, where θ,ϕ, ψ are the orientation of the principal axes. This saliency tensor is sufficient to describe feature saliency for unpolarized structure such as non-oriented curves and surfaces. When dealing with boundaries of regions or with oriented data (edgels), we need to associate a polarity to the feature orientation to distinguish the inside from the outside. Polarity saliency, which relates the polarity of the feature and its significance, will be addressed when we discuss region inference in section 4.

According to the spectrum theorem [2], a general saliency tensor $\boldsymbol{T}$ can be expressed as a *linear* combination of a stick tensor, which describes a perfect orientation estimation, a plate tensor, which describes orientation uncertainty except in one direction, and a ball tensor, which describes total orientation uncertainty as:

$$\begin{aligned}\boldsymbol{T} = {} & (\lambda_1-\lambda_2)\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T + (\lambda_2-\lambda_3)(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T + \hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T)\\ & + \lambda_3(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T + \hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T + \hat{\boldsymbol{e}}_3\hat{\boldsymbol{e}}_3^T)\end{aligned} \qquad (2.2)$$

where $\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^{\mathrm{T}}$ describes a stick, $(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^{\mathrm{T}}+\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^{\mathrm{T}})$ describes a plate, and $(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^{\mathrm{T}}+\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^{\mathrm{T}}+\hat{\boldsymbol{e}}_3\hat{\boldsymbol{e}}_3^{\mathrm{T}}t)$ describes a ball. Figure 2 shows the shape of a general saliency tensor as described by the stick, plate and ball components. A typical example of a general tensor is one that represents the orientation estimates at a corner. Using orientation selective filters, multiple orientations $\hat{\boldsymbol{u}}_i$ can be detected at corner points with strength s_i. We can use the tensor sum $\Sigma s_i^2\hat{\boldsymbol{u}}_i\hat{\boldsymbol{u}}_i^{\mathrm{T}}$ to compute the distribution of the orientation estimates. The addition of 2 saliency tensors simply combines the distribution of the orientation estimates and the saliencies. This linearity of the saliency tensor enables us to combines votes and represent the result efficiently. Also, since the input to this voting process can be represented by saliency tensors, our saliency inference is a closed operation whose input, processing token and output are the same.

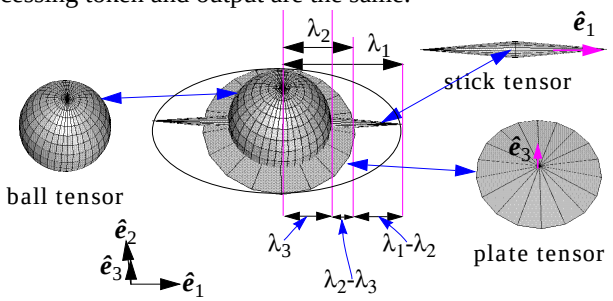


Figure 2 A general saliency tensor

2.3 Tensor Voting

Given a sparse feature saliency tensor field as input, each active site has to generate a tensor value at all locations that describe the voter's estimate of orientation at the location and the saliency of the estimation. The value of the tensor vote cast at a given location is given by the voting function which characterizes the structure to be detected. Due to the general properties of the function, this vote generation process can be implemented as a convolution with three saliency tensor fields, one encodes the votes due to the stick component of the voter, one encodes the votes due to the plate component of the voter, and one encodes the votes due to the ball component of the voter. Moreover, the plate voting field and the ball voting field are generated from the stick voting field, which unifies the process for handling various orientation information.

After all the input site saliency tensors have cast their votes, the vote interpreter builds feature saliency maps from the now densified saliency tensor field by computing the eigensystem of the saliency tensor at each site. According to equation (2.2), each saliency tensor can be broken down into three components, $(\lambda_1-\lambda_2)\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^{\mathrm{T}}$ corresponds to an orientation, $(\lambda_2-\lambda_3)(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^{\mathrm{T}}+\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^{\mathrm{T}})$ corresponds to a locally planar junction whose normal is represented by $\hat{\boldsymbol{e}}_3$, and $\lambda_3(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^{\mathrm{T}}+\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^{\mathrm{T}}+\hat{\boldsymbol{e}}_3\hat{\boldsymbol{e}}_3^{\mathrm{T}})$ corresponds to an isotropic junction.

For surface inference, surface saliency therefore is measured by $(\lambda_1-\lambda_2)$, with normal estimated as $\hat{\boldsymbol{e}}_1$. The curve junction saliency is measured by $(\lambda_2-\lambda_3)$, with tangent estimated as $\hat{\boldsymbol{e}}_3$. For curve inference, the curve saliency is measured by $(\lambda_1-\lambda_2)$, with tangent estimated as $\hat{\boldsymbol{e}}_1$. The planar junction saliency is measured by $(\lambda_2-\lambda_3)$, with normal estimated as $\hat{\boldsymbol{e}}_3$. In both case, isotropic (or point) junction saliency is measured by λ_3.

Notice that all the feature saliency values do not depend only on the saliency of the feature estimation, but also are determined by the distribution of orientation estimates.

Since the voting function is smooth, sites near salient structures will have high feature saliency values. The closer the site to the salient structure, the higher the feature saliency value the site will have. Structure features are thus located at the local maxima of the corresponding feature saliency map. The vote interpreter hence can extract features by non-maxima suppression on the feature saliency map. To extract structures represented by oriented salient features, we use the method developed in [4]. Note that a maximal surface/curve is usually not an iso-surface/iso-contour. By definition, a maximal surface is a zero-crossing surface whose first derivative will change sign if we traverse across the surface. By expressing a maximal surface as a zero-crossing surface, an iso-surface marching process can be used to extract the maximal surface. Similar discussion applies to maximal curve extraction. Details for extracting various structures from feature saliency maps can be found in [4][9].

3 Inference from Symmetric Features

While the salient structure inference engine defined above performs structure inference efficiently, it is the definition of the voting function that determines the effectiveness of the inference. For curve and surface inference, the voting function specifies how curve or surface orientation is estimated by a voting feature regarding each location in the domain space.

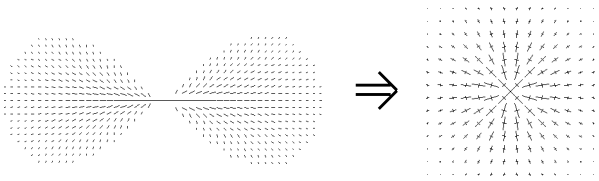
Since the orientations of the votes should change gradually in a local neighborhood, any curve element with an orientation, say $\hat{\boldsymbol{u}}$, must be connected to points in a local neighborhood through non-crossing, C_1-continuous paths with initial orientations equal to $\hat{\boldsymbol{u}}$. From a energy consumption point of view, circular path, which is the only C_1-continuous path that has constant curvature and is defined for all pairs of points and orientations, is preferred over other smooth paths. Similarly, surface patch elements are connected to other points through spherical surfaces.

Besides the orientation of the votes, the voting function also specifies the saliency of the votes. As stated in the beginning of section 2.3, the influence of a voter must be smooth, and decreases with distance. Moreover, low curvature is preferred over high curvature. We use the Gaussian function to model the decay *DF* of voter's influence with path length *s* and curvature ρ as:

$$DF(s,\rho) = e^{-\left(\frac{s^2 + c\rho^2}{\sigma^2}\right)}$$

where *c* is the constant that reflects the relative weight of path length and curvature and σ is the scale factor that determines the rate of attenuation.

Having defined the stick tensor voting field, the voting fields for plate and ball tensor are obtained by applying equation (2.6). Figure 3 depicts slices of the symmetrical voting fields correspond to the voting function defined in above for the inference of curves. Applying similar argument, the voting field for surface inference is derived with similar shape with individual votes oriented orthogonal to that of the curve voting field.

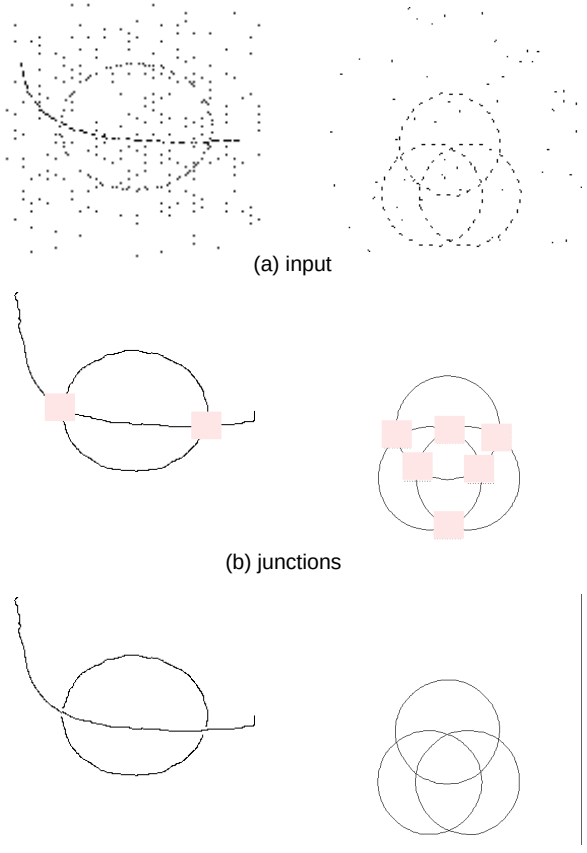


(a) stick tensor voting field

(b) ball tensor voting field

Figure 3 Voting fields for surface inference

Applying the voting functions defined in above, we have obtained results on noisy synthetic data. Figure 4 shows 2 examples of curve inference in 2-D.



(a) input

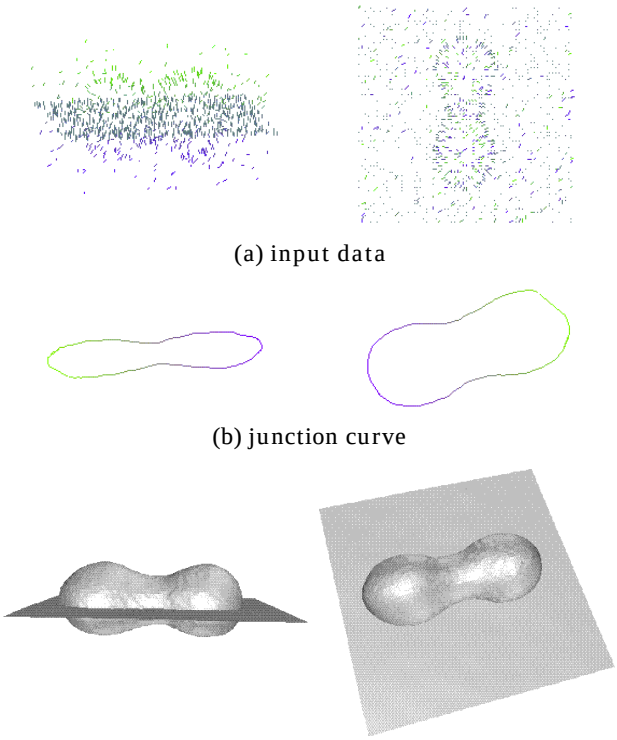
(b) junctions

(c) integrated output

Figure 4 2 examples of 2-D curve inference

Figure 5 illustrates an example of surface inference in 3-D, in which a peanut is intersected with a plane among noisy data. The junction curve extracted is shown in

Figure 5(b) and the integrated salient structure computed is shown in Figure 5(c).



(a) input data

(b) junction curve

(c) integrated description

Figure 5 An example of surface inference in 3-D

3.1 Complexity

The efficiency of the salient structure inference engine is determined by the complexity of tensor voting and vote interpretation. Given a voting function specified by the 3 discrete 3-D tensor fields with C voxels each, tensor voting requires $O(Cn)$ operations, where n is the number of input features. The size of C depends on the scale of the voting function and is usually much smaller than the size of the domain space, m. It is therefore the density of the input features that usually determines the time complexity of tensor voting. Since the density of input features is usually sparse, the total number of operations in tensor voting is normally a few times the size of the domain space. In vote interpretation, the saliency tensor in each location of the domain space is decomposed into its eigensystem, which takes $O(1)$ for each of the m elements in the domain space. The marching cube/square [5] only takes $O(m)$. Therefore the complexity of vote interpretation is $O(m)$.

In summary, the time complexity of the salient structure inference engine is $O(Cn+m)$, where C is the size of the tensor fields specifying the voting function, n is the number of input features, and m is the size of the domain space. Notice that all the processing is local and thus can be performed in parallel, which allows a $O(1)$ implementation on a parallel computer architecture.

4 Polarity Saliency and Region Inference

So far, we have only considered the inference of undirected structures from undirected features. However, in real applications, we often deal with regions in 2-D or 3-D. Computationally, it is economical to represent regions differentially, that is by their boundaries. By following the contour of a region, one knows which is the inside versus the outside. The simplest encoding for region therefore is to associate a polarity, which denotes the inside of the object, with each curve element along the region boundary. It is precisely this polarity information that is being expressed in the output of an edge detector, where the polarity of the directed edgel indicates which side is darker. As shown later, ignoring this information may be harmful. We therefore propose to augment our representation scheme to express this knowledge, and to use it to perform directed curve and region grouping.

4.1 Incorporating polarity

We first discuss how polarity is being incorporated into our salient structure inference engine for the inference of directed structure from directed features. In the spirit of our methodology, we attempt to establish at every site in the domain space a measure we called polarity saliency which relates the possibility of having a directed curve passing through the site. In particular, a site close to two parallel features with opposite polarities should have low polarity saliency, although the orientation saliency will be high.

Encoding polarity

Figure 6 depicts the situation when the voter has the polarity information. Since polarity is only defined for directed features, it is necessary to associate polarity only with the stick component of the saliency tensors. Thus, a scalar value ranging from -1 to 1 encodes polarity saliency. The sign of this value indicates which side of the stick component of the tensor is the intended direction. The size of this value measures the degree of polarization at a site. Initially, all sites with directed features have polarity saliency of -1, 0, or 1 only.

To propagate this polarity information, we need to modify the voting function accordingly. Note that polarity does not change along a path or a surface. Moreover, polarity is only associated with the stick component of the voting tensor which only casts stick tensor votes. Hence, to incorporate polarity into the voting function, we only need to assign either -1 or 1 to each stick tensor vote as polarity saliency, which is determined by the polarity of the voter

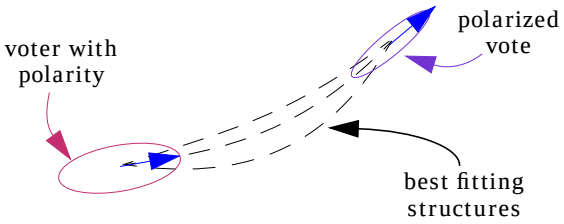


Figure 6Encoding Polarity Saliency

and the connection between the stick component of the voter and the site.

Inferring polarity

Since polarity is associated with an orientation, we need to take the orientation into account when we infer polarity saliency. Note that despite the association between polarity and orientation, they are independent of each other, and thus are inferred separately. An intuitive way to determine polarity is to compute the vector sum $u(x,y) = \sum \boldsymbol{v}_i(x,y)$ of these directed votes $\boldsymbol{v}_i(x,y)$'s. The length of the resulting vector gives the degree of polarization and the resulting direction relates the indented polarity. We thus assign to every site a polarity saliency $PS(x,y)$ as:

$$(PS(x,y) = \text{sgn}(u(x,y) \bullet \hat{e}_1(x,y))|u(x,y)|) \qquad (3.1)$$

where $\hat{e}_1(x,y)$ is the major direction of the saliency tensor obtained at the site (x,y).

Extracting directed features

Directed features are those which have both high orientation and polarity saliencies. The natural way to measure directed feature saliency is to take the product of the unpolarized feature saliency and the polarity saliency. Figure 7(b) shows the resulting directed curve saliency map for the pair of the directed circles that describe the filled circle in Figure 7(a). Compare to the curve saliency map shown in Figure 7(d) obtained for a pair of empty circles in Figure 7(a), the high agreement between the circles, which is acceptable for the empty circle pair but not the filled circle pair, is eliminated by properly incorporating the polarity information. Regions bounded by directed features can be extracted by the method described in section 2.3.

4.2 Boundary Inference

Here we address the problem of inferring boundaries from point input. A typical input is shown in Figure 9(a), where local boundary detection is hard. Again, we seek to compute at every site in the domain space a measure we call boundary saliency which relates the possibility of having the site being on the boundary of a region. A boundary

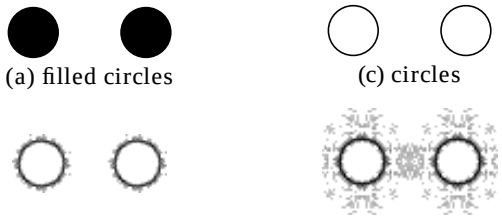


Figure 7 Empty and filled circle pairs

point has the property that most of its neighbors are on one side. We therefore can identify boundary points by computing the directional distribution of neighbors at every data point, as illustrated in Figure 8. This local discontinuity estimation is similar to the orientation estimation for salient structure inference.The voting function for boundary inference can be characterized as a radiant pattern with strength decays with distance from the center.

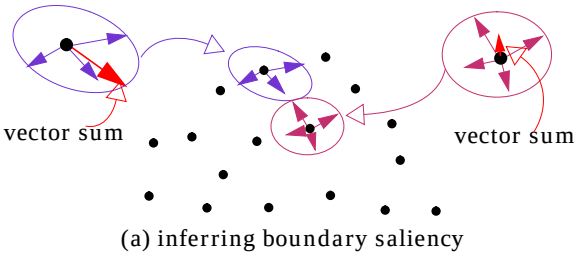
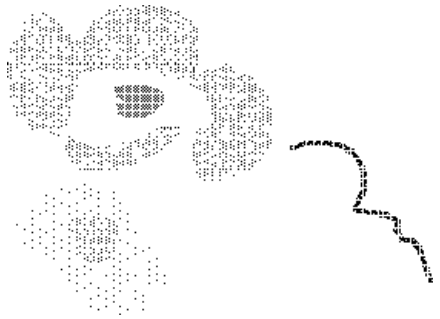


Figure 8 Boundary inference

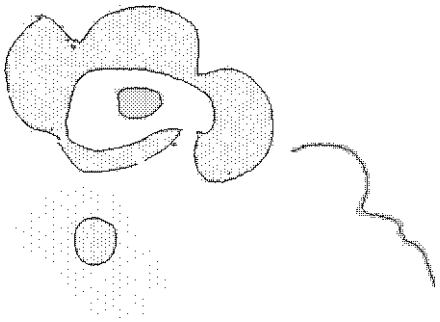
Since a point on the boundary will only receive votes from one side while a point inside the region will receive votes from all directions, the norm of the vector sum of the polarized votes should indicate the "boundariness" of a point, that is, the boundary saliency. Figure 9(b) presents the result of this boundary inference on the 2-D data set depicted in Figure 9(a). Observe that points are not labeled in absolute terms as borders or non-borders, but are presented with reservations as indicated by the boundary saliencies.

Using these boundary saliencies as the initial saliencies, the boundariness of the data points can be further verified by locating the surfaces/curves that describe the region boundaries. Figure 9(c) depicts the resulting region inferred from the boundary points shown in Figure 9(b). Note that we not only find the region, but also accurately find the corners.

Figure 9 Region inference from sparse data



(a) input



(b) salient boundary inferred

Figure 10 Inference of multiple boundaries

Depicted in Figure 10 and Figure 11 are the results of applying our method to inferring multiple junctions, curves and region boundaries from noisy data. Notice that regions with very sparse data is not detected. We believe this problem has to be solved by using multiple scales in detection.

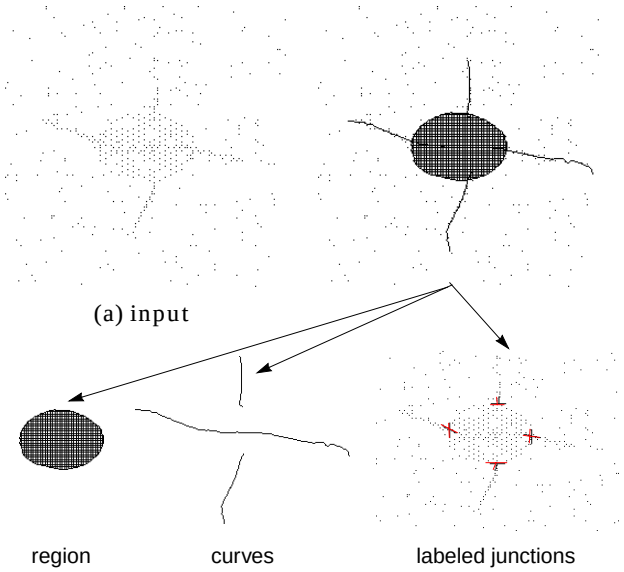
5 Conclusion

We have presented a unifying computational framework for the inference of multiple salient structures including junctions, curves, regions and surfaces from a sparse noisy set of points, curve elements, and surface patch elements. The methodology is grounded on two complementary aspects:

- the data is represented as a tensor, which allows to capture not only the local information, but also its associated confidence (saliency) and uncertainty;
- the communication between sites is performed by voting, which is efficiently implemented as a convolution-like operation.

The method is non-iterative, requires no initial guess or thresholding, and the free parameter is scale.

We have demonstrated here the approach on a small number of examples in 2-D and 3-D. Our current research involves the integration with a state-of-the-art edge detector, and thorough validation through many more examples.



(b) detected salient structures

Figure 11 Overlapping Salient features automatically detected by the method

References

[1] J. Dolan and R. Weiss, "Perceptual Grouping of Curved Lines", *Proc. Image Understanding Workshop*, 1989, pp. 1135-1145.

[2] G.H. Granlund and Knutsson, *Signal Processing for Computer Vision*, Kluwer Academic Publishers, 1995.

[3] G. Guy and G. Medioni, "Inference of Surfaces, 3D Curves and Junctions from Sparse 3-D Data", *Proc. IEEE Symposium on Computer Vision*, Coral Gable, 1995, pp. 599-604.

[4] G. Guy, *Inference of Multiple Curves and Surfaces from Sparse Data*, Ph.D. dissertation, Technical Report IRIS-96-345, Institute for Robotics and Intelligent Systems, University of Southern California.

[5] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3-D Surface Reconstruction Algorithm", *Computer Graphics*, vol. 21, no. 4, 1987.

[6] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman and Co., San Francisco, 1982.

[7] P. Parent and S.W. Zucker, "Trace Inference, Curvature Consistency, and Curve Detection", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no.8, 1989, pp. 823-839.

[8] C.V. Stewart, "MINPRAN: A new robust estimator for computer vision", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 10, 1995, pp. 925-938.

[9] C.K. Tang and G. Medioni, "Integrated Surface, Curve and Junction Inference from Sparse 3-D Data Sets", *Proc. ICCV 98*, India, Jan. 1998

[10] S.W. Zucker and R.A. Hummel, "Toward a Low-Level Description of Dot Clusters: Labeling Edge, Interior, and Noise Points", *Computer Vision, Graphics, and Image Processing*, vol. 9, no.3, 1979, pp.213-234.