Are Iterations and Curvature Useful for Tensor Voting?*

Sylvain Fischer[1], Pierre Bayerl[2], Heiko Neumann[2], Gabriel Cristóbal[1], and Rafael Redondo[1]

[1] Instituto de Optica (CSIC), Serrano 121, 28006 Madrid, Spain
{sylvain,gabriel,rafa}@optica.csic.es
[2] Universität Ulm, Abt. Neuroinformatik, D-89069 Ulm, Germany
{pierre,hneumann}@neuro.informatik.uni-ulm.de

Abstract. Tensor voting is an efficient algorithm for perceptual grouping and feature extraction, particularly for contour extraction. In this paper two studies on tensor voting are presented. First the use of iterations is investigated, and second, a new method for integrating curvature information is evaluated. In opposition to other grouping methods, tensor voting claims the advantage to be non-iterative. Although non-iterative tensor voting methods provide good results in many cases, the algorithm can be iterated to deal with more complex data configurations. The experiments conducted demonstrate that iterations substantially improve the process of feature extraction and help to overcome limitations of the original algorithm. As a further contribution we propose a curvature improvement for tensor voting. On the contrary to the curvature-augmented tensor voting proposed by Tang and Medioni, our method takes advantage of the curvature calculation already performed by the classical tensor voting and evaluates the full curvature, sign and amplitude. Some new curvature-modified voting fields are also proposed. Results show a lower degree of artifacts, smoother curves, a high tolerance to scale parameter changes and also more noise-robustness.

1 Introduction

Medioni and coworkers developed tensor voting as an efficient method for contour extraction and grouping. The method, supported by the Gestalt psychology, is based on tensor representation of image features and non-linear voting, as described in [2]. See also [9] for a comparison with other existing methods. Tensor voting is a non-iterative procedure, in the sense that the original scheme implements only 2 steps of voting, claiming that no more iterations are needed. In opposition, other methods for perceptual grouping [4,3,1] refine the results

* This research is supported in part by the German-Spanish Academic Research Collaboration Program HA 2001-0087 (DAAD, Acciones integradas Hispano-Alemanas 2002/2003), the projects TIC2001-3697-C03-02 from MCYT and IM3 from ISCIII and HGGM grants. S.F. and R.R. are supported by a MECD-FPU and a CSIC-I3P fellowships, respectively.

T. Pajdla and J. Matas (Eds.): ECCV 2004, LNCS 3023, pp. 158–169, 2004.

by iterative feedforward-feedback loops. Therefore, the aim of this study is to investigate how an incremented number of iterations can improve the results of tensor voting. Some basic examples are analyzed and an extraction quality measurement is proposed. The later allows to perform a statistical study on the influence of iterations in a simple case.

A curvature improvement has been proposed by Tang and Medioni [7]. They compute the sign of curvature and use it for modifying the voting fields. We propose a more sophisticated calculation of the curvature information with a low computational cost. Instead of the sign of curvature, the proposed method evaluates the full curvature using part of the calculations previously performed by the tensor voting. We adopt a curvature compatibility approach that was described by Parent and Zucker [6]. A statistical evaluation is presented and the methods are finally tested with more complex data in presence of noise.

Section 2 briefly introduces the tensor voting method. Section 3 presents a study on the usefulness of iterations for tensor voting and the section 4 describes some improvements that can be achieved when both curvature information and iterations are used. Some concluding remarks are drawn in section 5.

2 A Brief Introduction to Tensor Voting

The classical algorithm will not be fully described in detail here and only a brief description is presented in order to stress the new contributions of this paper. For a more in depth study the reader can refer to [2,7]. Also, it is necessary to remark that the present work is only restricted to still 2D images, but it could be extended to N-dimensional features, like volumetric data or motion [8,5].

A local description of the curves at each point of the image can be encoded by a symmetric positive 2x2 tensor. Tensors can be diagonalized, their eigenvalues are denoted λ_1, λ_2 with $\lambda_1 \geq \lambda_2 \geq 0$ and corresponding eigenvectors are denoted by e_1, e_2. Tensors can be decomposed as follows:

$$T = (\lambda_1 - \lambda_2)e_1 e_1^T + \lambda_2 I \tag{1}$$

where I is the identity matrix. The first term is called the *stick component*, where e_1 is an evaluation of the tangent to the curve. The *stick saliency* $\lambda_1 - \lambda_2$ gives a confidence measure for the presence of a curve. The second term is called the *ball component*, and its saliency λ_2 gives a confidence measure to have a junction.

The classical tensor voting algorithm performs two voting steps in which each tensor propagates to its neighborhood. Stick tensors propagate mostly in the direction of e_1. The region of propagation is defined by the stick voting field which decay in function of the distance and curvature (see Eq. 3 and Fig. 2.h). Ball tensors propagate in all directions and decay with the distance. After all tensors are propagated, all contributions are summed up to define new tensors that will be used for the next step. That summation can be considered as an averaging or a *"voting"*. The first voting step is referred as *"sparse vote"* because the vote is performed only on points where tensors are not null. The last voting step is called *"dense vote"* because the vote is accomplished on every point. After

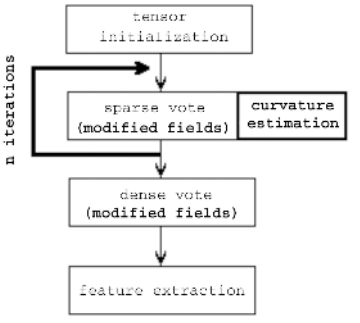


Fig. 1. Classical tensor voting consists of four steps. (1) Tensor initialization, (2) sparse voting, (3) dense voting, and (4) feature extraction. The new contributions are depicted with boldface characters, which describe iterations of the sparse voting process and curvature calculations during the sparse vote stage, modifying the voting fields by incorporating the calculated curvature.

all the voting steps are completed, curves are extracted as local maximum of stick saliency along the normal direction to stick components. Note that thresholds are necessary to eliminate low-saliency local maxima. These thresholds are held constant for each of the following experiments. Fig. 1 summarizes the different steps of the algorithm showing with boldface characters the new contributions proposed: an iterative sparse voting mechanism and a curvature calculation for modifying the voting fields.

3 Iterated Tensor Voting

3.1 Example

Tensor voting is a very efficient technique for grouping data-points that are separated by almost the same distance. A free parameter σ (the scale factor, see Eq. 3) has to be adjusted to the inter-distance between points. If σ is miss-adjusted, performance results strongly decrease: if σ is too small points will not be grouped, if σ is too big the grouping is less selective.

Fig. 2.a shows a simple example with two sets of points: first a three by three array of points separated by 11 pixels vertically and 13 pixels horizontally. Because the vertical distance is smaller, following Gestalt psychology rules, these points have to be grouped vertically. Secondly, a set of three points aligned in diagonal and separated by 42 pixel gaps. Because the gaps are different in both sets of points it is not possible to adjust σ for extracting both structures correctly. As it is shown in Fig. 2.c, if σ is small, i.e. around 5, only the vertical array is grouped. If σ is bigger than 15, only the diagonal line is correctly grouped (Fig. 2.i). Between these values (from σ =7 to 13, Fig. 2.e and g) none of these sets of points are accurately grouped.

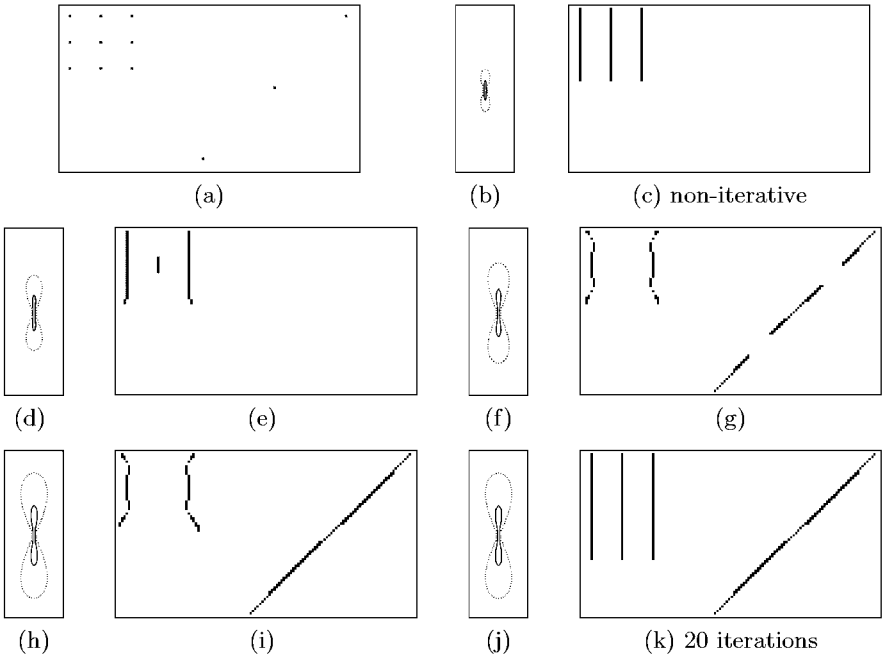
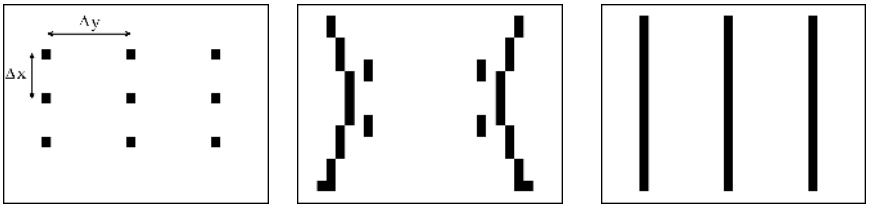


Fig. 2. Example showing the tensor voting results for different values of the scale factor σ and of the number of iterations. **a.** Data points belong to two sets: three points aligned in diagonal and an array which has to be grouped in vertically lines. **b.** Contours of the voting field for $\sigma = 5$ are drawn at 50% (solid line) and 5% (dash-dot line) of the maximum value (see Eq. (3) for voting fields description). **c.** Extraction results with the classical tensor voting algorithm (two voting steps) and $\sigma = 5$: array structure is accurately extracted, but the scale factor is too small to group diagonal points. **d. and f.** Voting fields with $\sigma = 9$ and $\sigma = 12$, respectively. **e. and g.** Contours extracted respectively with $\sigma = 9$ and $\sigma = 12$, by the non-iterative tensor voting. In both cases algorithm fails to find both array and diagonal points structures. The scale factor σ is too big for the array and too small for the diagonal points. **h. and j.** Voting field with $\sigma = 15$. **i.** With non-iterative tensor voting and $\sigma = 15$, diagonal points are correctly grouped, but not array points. **k.** With $\sigma = 15$ and 20 iterations the structure is accurately extracted, both array and diagonal line are correctly grouped.

Iterations are implemented on the sparse voting stage. For n iterations, $n-1$ sparse votes and one dense vote are required, as shown Fig. 1. An increased number of iterations can refine the results until the correct structure is extracted. Fig. 2.k shows the results with $\sigma = 15$ and 20 iterations. Both array and diagonal line structures are now simultaneously extracted what non-iterative algorithm was not able to do. Note that a normalization stage is applied after each iteration to keep the sum of tensor eigenvalues constant.



(a) (b) (c)

Fig. 3. Systematic evaluation of the influence of iterations using a three by three array of points. **a.** Original array of points separated by Δx, Δy (parameters are the same in all insets: $\Delta x = 4$, $\Delta y = 9$ and $\sigma = 10$). **b.** Contour extraction with the non-iterative tensor voting fails: central points are not grouped, lateral points are grouped but not in strictly vertical lines, moreover there are some artifacts (Q=0.40). **c.** The structure is well extracted after 10 iterations of voting: points are grouped in vertical lines (Q=2.38).

3.2 Statistics on the Influence of Iterations

A 3x3 array of points, shown in Fig. 3.a, is used to evaluate the effect of iterations on tensor voting. Vertical and horizontal distances between points are denoted Δx and Δy respectively. In the following, Δx will be chosen smaller than Δy. In such case points have to be grouped vertically (on the contrary if $\Delta x > \Delta y$ points would have to be grouped horizontally). Taking into account that points have to be grouped vertically, a measure of how good tensor orientation is can be represented by:

$$Q = -log_{10}\left[\frac{1}{9}\sum_{i=1,\ldots,9}\left(1 - \frac{T_i(1,1)}{S_i}\right)\right] \tag{2}$$

where i indexes the 9 points of the array. T_i is the tensor of the point i, S_i is the sum of eigenvalues of T_i and $T_i(1,1)$ the vertical component of the tensor T_i.

As vertical lines have to be extracted, tensors are correctly oriented if they have a form close to $T_i = S_i \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$. In such case $\sum(1-\frac{T_i(1,1)}{S_i})$ is close to zero, providing a high value for Q. Thus, Q can be considered as an extraction quality measurement for the described experiment. When $Q < 1$ tensors are miss-oriented and extraction can be considered as failed. On the contrary $Q > 2$ indicates tensors are well orientated and the structure is correctly extracted.

3.3 Results

Fig. 4 presents results for different parameters Δx, Δy and n (number of iterations). For all cases the scale factor σ is fixed to 10. Again, please note that in this study we are only considering cases where $\Delta x < \Delta y$ (which should yield vertical grouping following Gestalt rules of proximity and good continuation).

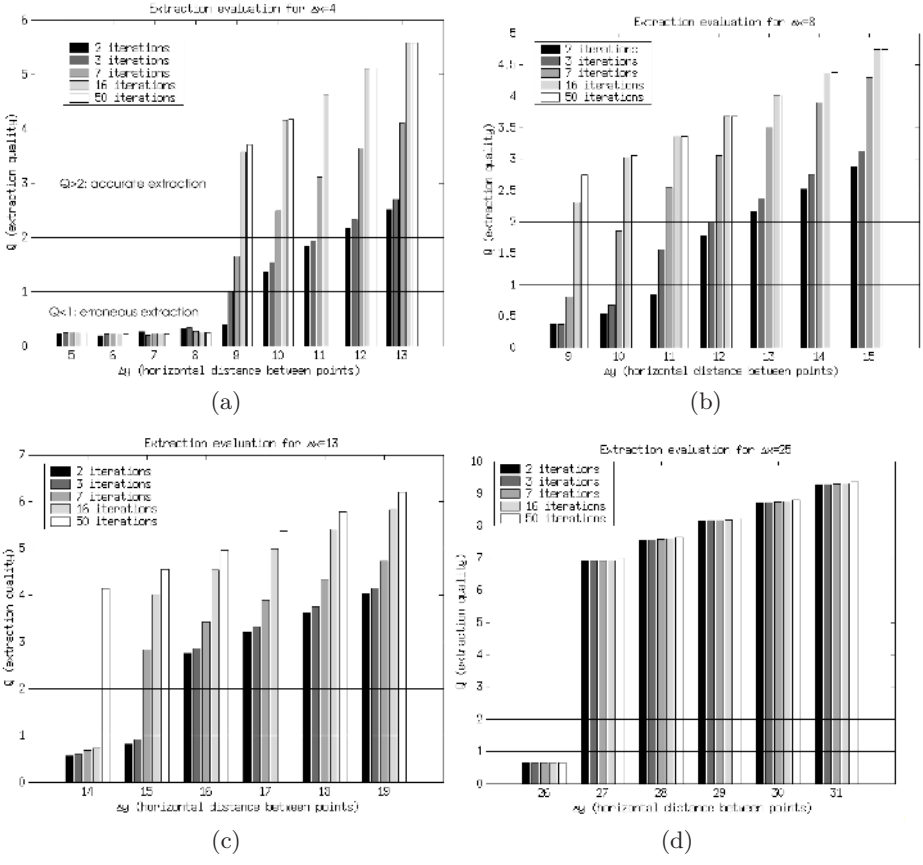


Fig. 4. Extraction quality as a function of array parameters Δx and Δy for the grid example of Fig. 3. The number of iterations n is indicated by different gray shades in the bars (two iterations bar corresponds to the classical algorithm with two voting steps). $\sigma = 10$ is held constant for the entire experiment. Only cases with $\Delta x < \Delta y$ are shown here. **a.** With a fixed $\Delta x = 4$ and $5 \leq \Delta y \leq 13$. If $\Delta y \ll \sigma$, σ is too large in comparison to the features and the extraction fails even if more iterations are deployed. If $9 \leq \Delta y \leq 11$ the structure is extracted using several iterations (results start from failed ($Q < 1$) when using the non-iterative algorithm up to accurate ($Q > 2$) when more iterations are deployed). Only if $\Delta y \geq 12$ the non-iterative algorithm is able to extract the desired information. **b.** $\Delta x = 8$ and $9 \leq \Delta y \leq 15$. **c.** $\Delta x = 13$ and $14 \leq \Delta y \leq 19$. In difficult cases like when $\Delta x \simeq \Delta y$ or $\Delta y \simeq \sigma$ several iterations are needed for extracting accurately the structure. **d.** $\Delta x = 25$ and $26 \leq \Delta y \leq 31$. Although σ is too small in comparison to the features, an accurate extraction is obtained due to the infinite Gaussian extension of the propagation fields.

Extraction is accurate for any number of iterations if σ corresponds to the optimal scale for the investigated stimuli and if there is no competition between vertical and horizontal grouping, that is, if $\sigma \ll \Delta y$ and $\Delta x \ll \Delta y$ (see Fig. 4.a,b,c,d at their rightmost parts).

If $\Delta y \ll \sigma$ it is impossible to extract the structure even if more iterations are deployed (see Fig. 4.a left part), the scale factor is indeed too large to be selective enough.

If $\Delta y \simeq \sigma$ the application of the classical algorithm fails to extract the curves. On the contrary, iterations allow tensor voting obtaining the correct structure as it can be observed in Fig. 4.a center part and Fig. 4.b left part. A similar situation is observed if $\Delta x \simeq \Delta y$ and $\Delta x, \Delta y$ are not much bigger than σ. Iterated tensor voting allow to extract the structure where the classical algorithm fails (see Fig. 4.c left part).

In conclusion, only if the features to be extracted are simple and they do not appear in competition, the non-iterative algorithm would suffice for correctly extracting image features. For more complicated cases, when some competition between orientations is present or when the scale factor σ is not precisely adjusted, more than two iterations are required. Moreover, it has been seen that in almost all cases iterations do not impair the quality of the results and on the contrary they allow to refine the final structures. In all, the use of iterations can help to overcome the limitations of the non-iterative method, improving the feature extraction results.

4 Curvature Improvement

4.1 Method

The proposed curvature improvement introduces a curvature calculation and modified stick voting fields. The curvature is evaluated in each voter point by averaging over all receiver points the curvature calculation ρ already computed in the classical tensor voting. In the classical tensor voting, a voter A votes on a receiver B with an amplitude described by the stick voting field equation:

$$V(A,B) = exp\left(-\frac{s(A,B)^2 + c\,\rho(A,B)^2}{\sigma^2}\right) \tag{3}$$

with

$$\rho(A,B) = \frac{2sin\theta}{d} \tag{4}$$

where $s(A,B)$ and $\rho(A,B)$ are respectively the length and the curvature of the circular arc which is tangent to $\vec{e_1}\ (A)$ in point A and goes through point B (see Fig. 5.a). d is the Euclidean distance between both points A and B, θ is the angle between vectors $\vec{e_1}\ (A)$ and $\overrightarrow{AB}$. σ -the scale factor- and c are constants. Fig. 2.b,d,f,h shows the contours of such voting fields for different values of σ.

The curvature will be evaluated in each voter point A. To permit inflexion points and changes of curvature, the curvature is calculated separately in both half planes P_+ and P_- defined respectively by $P_+ = \{B, (\vec{e_1}\ (A), \overrightarrow{AB}) > 0\}$ and $P_- = \{B, (\vec{e_1}\ (A), \overrightarrow{AB}) < 0\}$. The weighted average over each half plane gives $\gamma_i(A)$ (where $i = +$ or $-$), which is a curvature evaluation at the point A:

$$\gamma_i(A) = \frac{\sum_{B \in P_i} (\lambda_1(B) - \lambda_2(B))\, V(A,B)\, \rho(A,B)}{\sum\limits_{B \in P_i} (\lambda_1(B) - \lambda_2(B))\, V(A,B)} \tag{5}$$

where $\lambda_1(B)$, $\lambda_2(B)$ are the eigenvalues of the tensor B. The weighted average is very similar to the "voting" used in tensor voting: the same weighting functions composed by the voting fields V and the stick saliency $\lambda_1 - \lambda_2$ are used.

The γ_i determined at one iteration, can then be used in the next iteration for modifying the stick voting fields. The following equation extends Eq. 3:

$$V(A,B) = exp\left(-\frac{s(A,B)^2 + c\,(\rho(A,B) - \gamma_i(A))^2}{\sigma^2}\right) \quad for\ any\ B \in P_i \tag{6}$$

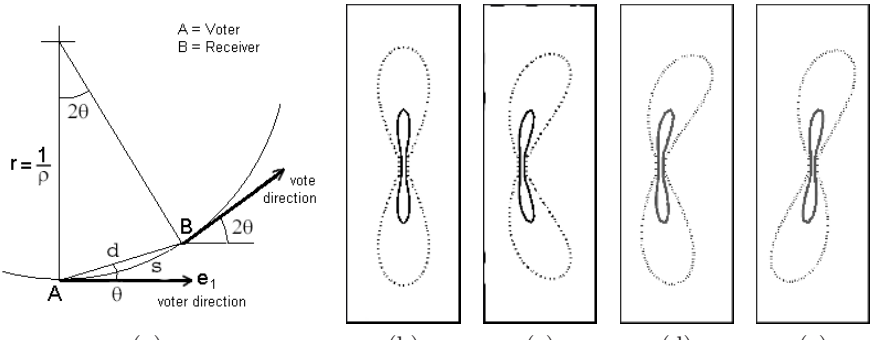


Fig. 5. **a.** Tensor voting fields are build calculating the distance d, the angle θ, the arc longitude s and the curvature ρ between the voter A oriented by its first eigenvector e_1 and the receiver B. In the curvature improvement the curvature is evaluated in the voter A by averaging ρ over all receivers. **b.** Classical voting field without curvature. Contours are drawn at 50% and 5% of the maximum value, $\sigma = 15$ for all voting fields of the figure. **c.** Symmetric curved voting field with curvatures $\gamma^+ = \gamma^- = .06$. **d.** Curved voting field with different curvature in both half planes, $\gamma^+ = .09$ and $\gamma^- = .03$. **e.** Curved voting field with inflexion, $\gamma^+ = .06$ and $\gamma^- = -.06$.

Some examples of such curvature-modified voting fields are shown Fig. 5.c,d,e. See in comparison the former contours Fig. 5.b. In points where the ball component has a significant level in comparison to the stick component, curvatures have to be considered as zero because no reliable curvature calculation is possible if curve orientation is itself not reliable. Therefore curved voting fields are employed only where tensor orientation has high confidence (the curved voting fields are only used under the condition $\frac{\lambda_1}{\lambda_2} > 10$).

Remarkably the method follows the "voting" methodology. Curvature is found by averaging. Moreover it uses the same voting fields V as tensor voting. It can then be hoped to conserve the good properties of the tensor voting, like the robustness to noise. The curvature improvement does not entail an important additional computational cost in comparison to the classical method,

while it uses the same kind of operations as the tensor voting and reuses calculations already done, i.e. in the curvature calculation of Eq. 5 all variables λ_1, λ_2, V and ρ are already computed by the classical tensor voting.

Note also that an increased number of iterations is necessary to refine the results. The number of iterations can be considered as an additional parameter of the algorithm. A procedure could also be implemented for stopping the iterations when the results do not change much from one iteration to the following one. For all examples presented here a fixed number of iterations is used. 10 iterations have be seen to be sufficient unless data structure presents some special ambiguity.

In the following, the curvature improvement will be compared with the non-iterative tensor voting and iterative tensor voting without curvature improvement. Results need to be compared with Tang and Medioni's method taking into account the sign of curvature [7], although this was out of the scope of the present study.

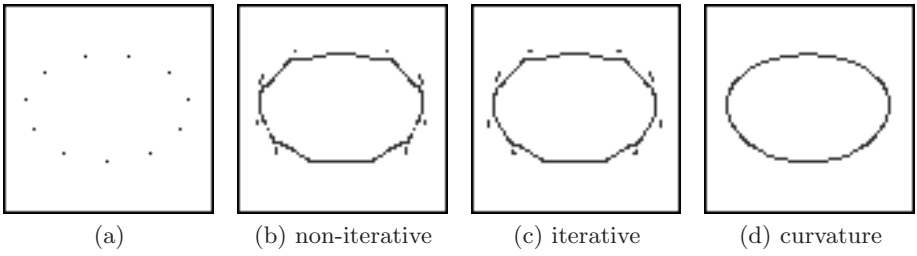
4.2 Statistical Study

Fig. 6.a shows an image composed by sparse points located on the edges of an ellipse. The distance between points vary between 6 to 12 pixels. This example is used for comparing the three versions of the algorithm. For different values of the scale factor σ, we count the number of points erroneously extracted outside the ellipse contour, tolerating a deviation of two pixels around the ideal ellipse. Results are presented in Fig. 6.b-e.

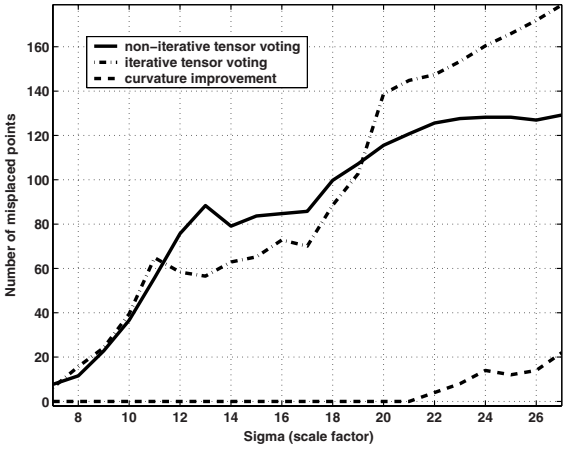
All versions of the algorithm require a σ value to be higher than a minimum value ($\sigma \geq 7$ in the present case) for extracting the contour of the ellipse. With a smaller value of σ, points are not grouped together. On the other hand, σ needs to be small for avoiding artifacts, i.e. the number of misplaced points increases strongly for tensor voting without curvature information for $\sigma > 10$, and for $\sigma > 34$ if the curvature improvement is considered. Classical tensor voting adequately extracts the contours, although with artifacts, for σ between 7 and 10. Iterations have few influence on the results. In comparison curvature improvement extracts adequately the ellipse over a large range of σ values, i.e. between 7 to 34. Moreover it does not produce any artifacts for σ between 7 and 21 and yields smoother slopes.

4.3 Hand-Written Text Example

Fig. 7 shows another example of contour extraction with the three versions of the algorithm: non-iterative, iterative with 10 iterations and iterative with the curvature improvement (with also 10 iterations). The first image "Cyan" (Fig. 7.a) is composed of sparse points along handwritten characters. The second one (Fig. 7.b) is the same image "Cyan" with 20% of noise (i.e. every fifth data point is noise). Same parameters are used for each method. After tensor voting is applied the contours of the letters are extracted. Results show tensor voting with 10 iterations (Fig. 7.e) reduces the artifacts and closes the curves better



(a) (b) non-iterative (c) iterative (d) curvature



(e)

Fig. 6. Comparison between the three methods. **a.** The input image is composed by a sparse set of dots dispersed along the edges of an ellipse. In insets a., b. and c. all parameters are the same and $\sigma = 8$. **b. and c.** Extraction results with, respectively, the non-iterative algorithm and 10 iterations of tensor voting. The ellipse is adequately extracted but artifacts can be observed, moreover slopes are not smooth. Both methods provide similar results. **d.** With the curvature improvement and 10 iterations, the ellipse is extracted without artifacts and with smooth curves. **e.** Results for σ varying between 7 and 27 are presented. The number of points erroneously extracted, that is extracted out of the ellipse are plotted for each method. Tensor voting without curvature information extract the ellipse, although always with artifacts, for σ between 7 and 10. Curvature improvement extracts it without artifacts and tolerates a larger range of σ (from 7 to 21).

than non-iterative tensor voting (Fig. 7.c). With the curvature improvement (Fig. 7.g) extracted contours of the curves have even less artifacts and are much smoother. Comparison of the results with the noisy image (Fig. 7.d,f,h) shows that curvature improvement does not impair the quality but even improves it, e.g. contour continuity is better preserved.

For regions with straight segments and junctions both curvature improvement and iterative tensor voting behaves similarly. Therefore, curvature improvement does not impair the results for such situations. As a consequence the curvature

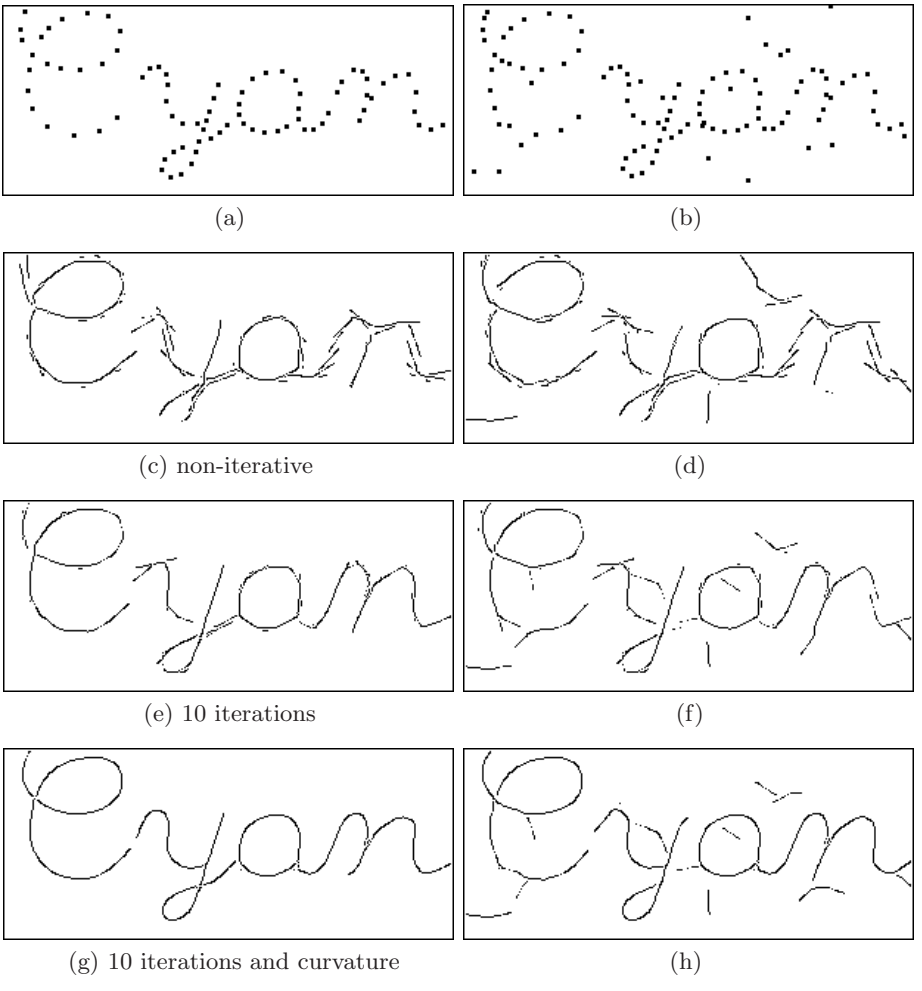


Fig. 7. A hand written example. **a.** The test image "Cyan" is a 128x304 pixel image composed by points dispersed along hand-written letters. For better visualization points are magnified. **b.** The second test image is the same image "Cyan" with 20% noise. Parameters are the same for all experiments ($\sigma = 15$). **c and d.** Extraction results of respectively the image "Cyan" and the noisy image version with non-iterative tensor voting. In both cases the algorithm fails to close the curves and yields high level of artifacts. **e and f.** Extraction results of "Cyan" images with 10 iterations. Curves are better closed and the level of artifacts is lower than with non-iterative tensor voting. **g and h.** Extraction results with the curvature improvement and 10 iterations. The text is accurately extracted, with less artifacts and smoother slopes. Results resist slightly better to noise than without curvature improvement. It is remarkable that the curve continuity of the letters C, Y and N is preserved.

improvement can be used for any kind of images. Remarkably, curvature improvement accurately extracts the structure of the example Fig. 2.a using the same parameters ($\sigma = 15$ and 20 iterations).

5 Conclusion

This paper demonstrated that iterations are useful for tensor voting, particularly for extracting correct contours in difficult situations like feature competition or scale parameter misadjustment. In almost all cases iterations do not impair the quality of the results and on the contrary they allow refining and improving the final structures. The curvature improvement provides better results for curved features as it reduces the level of artifacts and smoothes curves, besides the fact that it also increases the robustness of the method to scale parameter misadjustment and noise.

References

1. Hansen, T., Sepp, W., Neumann, H.: Recurrent long-range interactions in early vision. S. Wermter et al. (Eds.): Emergent neural computational architectures, LNAI 2036 (2001) 127–138
2. Medioni, G., Lee, M.-S., Tang, C.-K.: A computational framework for feature extraction and segmentation. Elsevier Science (mar. 2000)
3. Mingolla, E., Ross, W., Grossberg, S.: A neural network for enhancing boundaries and surfaces in synthetic aperture radar images. Neural Networks **12** (1999) 499–511
4. Neumann, H. and Mingolla, E.: Computational neural models of spatial integration in perceptual grouping. T.F. Shipley and P.J. Kellman, editors, From Fragments to Objects - Segmentation and Grouping in Vision. Elsevier Science (2001) 353–400
5. Nicolescu, M. and Medioni G., Layered 4D Representation and Voting for Grouping from Motion. IEEE Trans. P.A.M.I. **25(4)** (2003) 492–501
6. Parent, P. and Zucker, S.: Trace inference, curvature consistency, and curve detection. IEEE Trans, P.A.M.I. **11** (1989) 823–839
7. Tang, C.-K. and Medioni, G.: Curvature-Augmented Tensor Voting for Shape Inference from Noisy 3D Data. IEEE Trans. P.A.M.I. **24(6)** (June 2002) 868–864
8. Tang, C.-K., Medioni, G., Lee, M.: N-dimensional tensor voting and application to epipolar geometry estimation. IEEE Trans P.A.M.I. **23(8)** (2001) 829–844
9. Williams, L.R. and Thornber, K.K.: A comparison of measures for detecting natural shapes in cluttered backgrounds. Int. Jour. of Computer Vision **34(2/3)**(2000)81–96