ISPRS Journal of Photogrammetry and Remote Sensing 95 (2014) 93–108

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Tensor-Cuts: A simultaneous multi-type feature extractor and classifier and its application to road extraction from satellite images

Charalambos Poullis [1]

Immersive and Creative Technologies Lab, Cyprus University of Technology, Cyprus[2]

ARTICLE INFO

Article history:
Received 24 September 2013
Received in revised form 5 June 2014
Accepted 6 June 2014

Keywords:
Feature extraction
Feature classification
Graph-Cuts
Road extraction
Tensor

ABSTRACT

Many different algorithms have been proposed for the extraction of features with a range of applications. In this work, we present Tensor-Cuts: a novel framework for feature extraction and classification from images which results in the simultaneous extraction and classification of multiple feature types (surfaces, curves and joints). The proposed framework combines the strengths of tensor encoding, feature extraction using Gabor Jets, global optimization using Graph-Cuts, and is unsupervised and requires no thresholds. We present the application of the proposed framework in the context of road extraction from satellite images, since its characteristics makes it an ideal candidate for use in remote sensing applications where the input data varies widely. We have extensively tested the proposed framework and present the results of its application to road extraction from satellite images.

1. Introduction

For many years, there has been a plethora of research work in the area of automatic extraction of road networks from satellite images. Many different algorithms have been proposed so far each with its strengths and weaknesses. Although there have been considerable progress in this area there still exists a rather large gap between the current state-of-the-art and the desired goal. To make matters worse each proposed algorithm excels when used with a particular type of data and fails with another. Quite often, the primary reason for this is due to the fact that existing segmentation frameworks used in the processing pipeline are only able to deal with particular types of imagery, or even worse require an abundance of thresholds which have to be tuned for different types of data/images. Hence, it is no surprise that the segmentation is often identified as the *weakest link* in automatic road extraction pipelines since it dictates the accuracy and quality of the final results.

Motivated by the limitation of our earlier work in Poullis and You (2010) to only classify curve pixels, we present a novel framework for the simultaneous segmentation and classification of multiple image features, called Tensor-Cuts. The proposed framework contains *no thresholds* since *all* information is encoded as tensors, and the efficient and global optimization technique Graph-Cuts is used to refine this information. Our technical contributions are:

- A novel segmentation framework which relies on tensorial representation, Gabor filters and global optimization using Graph-Cuts; named Tensor-Cuts.
- The proposed segmentation has *no threshold* since *all* information is encoded as tensors and refined using Graph-Cuts.
- An incremental improvement on the road center-point extraction procedure for the automatic extraction of center-points.

In contrast to our previous work (Poullis and You, 2010), the proposed segmentation encodes labels as tensors which can *simultaneously* capture information about three geometric types: surface, curve and junction. This eliminates the need for a pre-classification step (to separate curve pixels) prior to the segmentation as was the case in Poullis and You (2010) since both, the initial and final labeling have the form of tensors. Moreover, the energy function penalizes for dissimilarities between *all* pixels/tensors (rather than differences between orientations of *only* the pixels classified as curves) resulting in better defined segmentation. Thus, the user interaction previously required to mark foreground and background areas in the image prior to the processing is no longer required making the proposed framework automatic. Furthermore, the newly introduced energy label term in the Graph-Cut optimization ensures that a *minimum* number of labels are used in the new labeling which produces smoother results. Finally, the

[1] More information: http://www.poullis.org.
[2] http://www.theictlab.org.
E-mail address: charalambos@poullis.org

http://dx.doi.org/10.1016/j.isprsjprs.2014.06.006
0924-2716/

extraction of roads is achieved using an improved and more efficient variant of the procedure described in Poullis and You (2010) which exploits prior-knowledge about the characteristics and the function of roads to efficiently apply a set of Gaussian kernels to extract road *entry points* in the image. Once the entry points have been identified a variety of road tracking algorithms can then be employed for extracting the complete network; in our case tracking using template matching.

The paper is organized as follows: Section 2 presents a brief overview of the state-of-the-art in the area and Section 3 a system overview of the proposed method. Section 4 explains the intrinsic details of how the encoding of the information into their tensorial representation is performed. This includes the application of Gabor filters (Section 4.1.1), and the conversion of the response images to tensors (Section 4.1), the creation of the tensor labels (Section 4.2). Section 5 presents the application of the global optimization Graph-Cuts using tensors and an explanation of the energy functions used. Section 6 presents the road extraction pipeline which incorporates the segmentation framework. Finally, Section 7 shows the results of the proposed technique.

2. Related work

There has been considerable work in the area of automatic road extraction. Below we present an overview of the state-of-the-art in the area.

Doucette et al. (2001) present a self-organizing road map algorithm. Their approach proceeds by performing spatial cluster analysis as a mid-level processing technique. This enables them to improve tolerance to road clutter in high-resolution images, and to minimize the effect on road extraction of common classification errors. Their approach is designed in consideration of the emerging trend towards high-resolution multispectral sensors. Indeed, their preliminary results demonstrate robust road extraction ability due to the non-local approach, when presented with noisy input.

Hinz and Baumgartner (2003), propose a system which integrates detailed knowledge about roads and their context using explicitly formulated scale-dependent models. The knowledge about how and when certain parts of the road and context model are optimally exploited is expressed by an extraction strategy.

Bacher and Mayer (2005), proposed an automatic road extraction technique from satellite images of rural and suburban areas. The first step is the extraction of Steger lines in all spectral channels which are then used as cues for roads to generate training areas for a subsequent automatic supervised classification. The resulting classification is finally used as an additional source for the extraction of road candidates.

Mena and Malpica (2005) propose a system which includes four different modules: data pre-processing; binary segmentation based on three levels of texture statistical evaluation; automatic vectorization by means of skeletal extraction; and finally a module for system evaluation. The proposed system is quite efficient in producing the results.

On a different note, Lacoste et al. (2005) propose an approach which models the target line network by an object process, where the objects correspond to interacting line segments. They design the prior model so that they will exploit as fully as possible the topological properties of the network under consideration, while the radiometric properties of the network are modeled using a data term based on statistical tests.

In a different approach Grote and Heipke (2008), proposed a region-based approach on high resolution aerial images working from small local regions to roads as groups of road parts. The first step is to segment the image using the normalized cuts algorithm and group small segments to form larger segments; from these grouped segments road parts are then extracted.

In Das et al. (2011), the authors propose a multi-stage automated system for extracting road networks from high-resolution satellite images which produces impressive results. During the first stage, information about edges is extracted using a proposed process called "dominant singular measure" and information about regions is extracted using a probabilistic SVM classifier. This information is then integrated together using a constraint satisfaction neural network. A set of post-processing steps is then applied at a second stage, in order to improve the accuracy the extracted roads by removing false positives as well as to recover missing small areas due to false negatives. The system does not vectorize the road network but rather provides a binary classification of each pixel into road or non-road.

By employing neural networks with millions of trainable weights, Mnih and Hinton (2010) proposed the detection of roads which looks at a much larger context than was used in previous attempts at learning the task. The network is trained on massive amounts of data using a consumer GPU. The results confirm that their method works reliably on challenging urban datasets that are an order of magnitude larger than what was used to evaluate previous approaches.

Using stereoscopic satellite aerial images, Poz et al. (2012) proposes a semiautomatic method for 3-D road extraction in rural areas. They employ a strategy based on the dynamic programming algorithm which provides a solution to the road extraction problem in the object space. In order to find road centerlines, the extraction process begins by first measuring a few seed points in one image of the stereoscopic pair and then transforming these into the object-space reference system. Experimental results show that the proposed method is efficient and provides relatively accurate road centerlines.

A higher-order CRF model is presented in Wegner et al. (2013) for road network extraction from dense urban scenes. A CRF formulation is proposed for road labeling, where the prior is represented by higher-order cliques which connect sets of superpixels along straight line segments. Although the parameters are manually tuned for the clique sampling the results seem very promising.

Although a plethora of techniques have been proposed for the automatic or semi-automatic road extraction, the gap between the state-of-the-art and the desired goal, of automatic road extraction from satellite images, still remains wide. In this work we introduce a framework which is fully automatic and does not require interaction with the user to mark road and/or non-road pixels nor to exactly separate various objects from the image.

3. System overview

Fig. 1 shows an overview of the proposed system consisting of two phases:

1. *Tensor encoding:* The original image is filtered with a set of Gabor Jets and the responses are encoded into tensors. Similarly, a Gaussian hemisphere's geometric properties are encoded as tensors as described in Section 4.
2. *Global optimization:* The encoded tensors are optimized using the efficient, global optimization technique, Graph-Cuts, which is explained in Section 5.
3. *Road extraction:* Road center-point candidates are extracted using a set of single and bi-modal Gaussian-based kernels. Based on the road center-points the roads are then extracted via tracking.

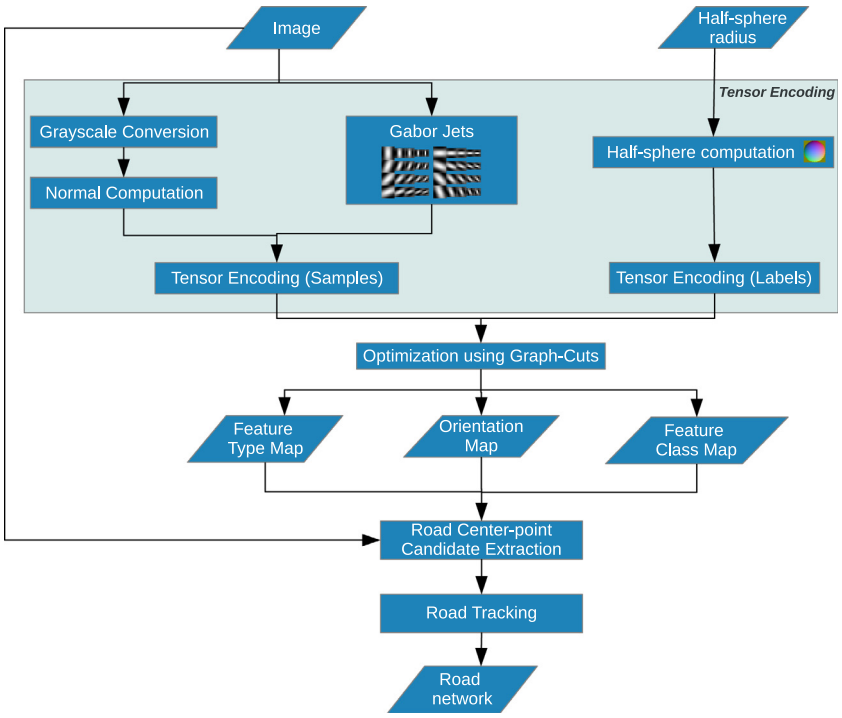


Fig. 1. Tensor-Cuts overview.

The input parameters to the system are (1) the satellite image and (2) the Gaussian hemisphere radius. The output of the system is a set of three maps each indicating, for each pixel in the image, (1) its feature type, (2) its orientation and (3) its class.

4. Tensor encoding

The tensor encoding[3] consists of the following two phases:

1. Firstly, the input image is filtered with a bank of Gabor Jets and at the same time, it is converted into a three-dimensional structure and the normal orientations are extracted. The extracted information is then encoded into tensors which will form the *initial set of labels* of the sample data. This process is presented in detail in Section 4.1.
2. Secondly, the input sphere radius is used to create a Gaussian hemisphere. The normals of the Gaussian hemisphere are computed and are encoded into tensors. The resulting tensors will form the *new labeling* of the sample data. This process is presented in detail in Section 4.2.

4.1. Derivation of the initial labeling *f*

The first phase of the tensor encoding is the derivation of the initial labeling *f* and is performed in three steps:

1. The application of a bank of Gabor Jets on the original image described in Section 4.1.1.
2. The conversion of the image to a three-dimensional structure described in Section 4.1.2.
3. The encoding of the computed information (Gabor response images and normals) to tensors. The resulting tensors will form the *initial labeling f*, described in Section 4.1.3.

[3] An excellent overview of tensorial representation and the tensor voting framework can be found in Medioni et al. (2000).

4.1.1. Application of Gabor Jets

Gabor filters have long been used as the most appropriate mathematical model to represent the function of the simple cell receptors present in the primary visual cortex (V1) because of their ability to respond to bars of given frequency and orientation (Hubel, 1982; Hubel and Wiesel, 1962; Hubel and Wiesel, 1974; Jones and Palmer, 1987; Daugman et al., 1985). Motivated by this unique characteristic we employ a bank of Gabor Jets consisting of a set of finely-tuned 2D Gabor functions $g(x,y)$ to extract linear features from the input image of given frequency and orientations.

A 2D Gabor function $g(x,y)$ is defined as the product of a complex sinusoidal (known as the carrier) and a Gaussian function (known as the envelope) and is defined as,

$$g(x,y) = Ae^{j(2\pi(u_0x+v_0y)+\phi)}e^{(-\pi(s_x^2(x-x_0)_\vartheta^2+s_y^2(y-y_0)_\vartheta^2))} \quad (1)$$

where A is a scale of magnitude, $(u_0, v_0$ is the spatial frequency which can also be expressed as polar coordinates with magnitude F_0 and direction ω_0, ϕ is the phase of the sinusoidal, (s_x, s_y) are scale factors for the axes, $(x_0, y_0$ are the peak coordinates of the oscillation, and ϑ is the rotation angle.[4]

Similarly to the work in Poullis and You (2010), the bank of Gabor Jets consists of 40 Gabor functions (8 orientations $\times$ 5 frequencies), each one designed to respond to a different frequency and different orientation as shown in Fig. 2(a). In all reported experiments, the 8 orientations $\theta_i, 0 \leqslant i < 8$ are uniformly distributed in the range $[0, \pi)$. Similarly, the 5 frequencies $\phi_j, 0 \leqslant j < 5$ are uniformly distributed in the range $[\frac{\pi}{32}, \frac{\pi}{8})$.

The application of the Gabor Jets on the input image in Fig. 2(b) results in 40 response images R_{θ_i,ϕ_j} as shown in Table 1.

4.1.2. Conversion of original image to three-dimensional structure

The input color image I_{RGB} can be interpreted as a three dimensional structure in color space, where each pixel $p \in I_{RGB}$ is

[4] ϑ denotes a rotation operation, e.g. $(x-x_0)_\vartheta = (x-x_0)\cos\vartheta + (y-y_0)\sin\vartheta$

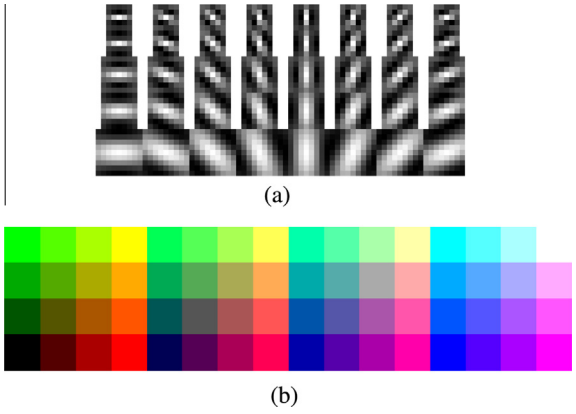


Fig. 2. (a) A bank of Gabor Jets consisting of 40 Gabor functions (8 orientations, 5 frequencies), each one designed to respond to a different frequency and orientation. (b) A simple image used for proof-of-concept, exhibiting linear features.

represented as a three-dimensional point $x_p \in \Re^3$ with Euclidean coordinates $x_p = \langle r_p, g_p, b_p \rangle$.

Instead of using the RGB color space, extensive empirical experiments have shown that by first converting the original input color image I_{RGB} to a *perceptually linear* color space such as CIELAB I_{Lab} significantly improves the subsequent processing. Perceptually linear means that a change of the same amount in a color value will produce a change of about the same visual importance. This is because the RGB color space is suitable for expressing the colors but it is not very good when it comes to color image segmentation and analysis due to the large correlation between the three primary colors red, green and blue. The CIELAB color, on the other hand, controls color and intensity information more independently and is especially efficient when measuring small color differences. The image I_{Lab} is then converted to grayscale I_g and finally, converted to image I_{XYG} where each pixel $p \in I_{XYG}$ is represented as a three-dimensional point $x_p \in \mathbb{R}^3$ with Euclidean coordinates $x_p = \langle X_p, Y_p, G_p \rangle$, where X_p is the horizontal axis position of the pixel p and has a value in the range of $0 \leqslant X_p < W$ (where W is the width of the image I_{XYG}), Y_p is the vertical axis position of the pixel p and has a value in the range of $0 \leqslant Y_p < H$ (where H is the height of the image I_{XYG}) and G_p is the grayscale value of the pixel p and has a value in the range of $0 \leqslant G_p < 255$. Fig. 3 shows an example of the color space conversion. Fig. 3(a) shows the converted I_{XYG} being overlaid on its three-dimensional representation. Similarly, Fig. 3(b) shows the input color image overlaid on its three-dimensional representation. Another example is shown in Fig. 4 where the three-dimensional representation of the satellite image in Fig. 12(a) is shown.

Lastly, a three-dimensional mesh representing the image I_{XYG} is created using a nearest neighbor triangulation algorithm and the normal at each point is computed using local neighborhood information at each point in the mesh. For every point p_i of the mesh M corresponding to the image I_{XYG} we define the normal N_{p_i} of that point as,

$$N_{p_i} = \frac{1}{8} * \sum_{j=1}^{8} N_{p_j} \tag{2}$$

where N_{p_j} is the normal computed with the neighboring point p_j within the 8-neighborhood system. Each of the 8 normals N_{p_j}, is computed as the cross product of the vectors connecting the point p_i and two consecutive(in clockwise order) neighboring points p_j, p_{j+1}, as indicated by the vectors $\vec{a}, \vec{b}, \vec{c}, \vec{d}, \vec{e}, \vec{f}, \vec{g}, \vec{h}$ in Fig. 5.

4.1.3. Encoding of information to tensors

At this point, the information captured by the Gabor filters could be used for classifying each point in the image as a curve or non-curve. However, this would introduce the need for hard thresholds. Moreover, it would not take into account normal information computed from the three-dimensional representation of the image. To avoid the use of hard thresholds we integrate the tangent and normal information together using a tensorial representation. The reason for choosing this particular tensorial representation is the unique characteristic that a tensor can simultaneously capture the geometric information for multiple feature types (junction, curve, surface) and a saliency, or likelihood, associated with each feature type passing through the pixel being encoded. Thus, no decision has to be made about the feature type of each pixel, which eliminates the need for thresholds.

The information of the Gabor response images and the computed normals capture the local tangent and normal information at each point in the image. This information is encoded into second-order symmetric tensors which provides a mechanism for combining this information together. A second-order symmetric tensor T is defined as,

$$T = \begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \vec{e}_1^T \\ \vec{e}_2^T \\ \vec{e}_3^T \end{bmatrix} \tag{3}$$

$$T = \lambda_1 \vec{e}_1 \vec{e}_1^T + \lambda_2 \vec{e}_2 \vec{e}_2^T + \lambda_3 \vec{e}_3 \vec{e}_3^T \tag{4}$$

where $\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3 \geqslant 0$ are eigenvalues, and $\vec{e}_1, \ \vec{e}_2, \ \vec{e}_3$ are the eigenvectors corresponding to $\lambda_1, \lambda_2, \lambda_3$ respectively. By applying the spectrum theorem, the tensor T in Eq. (4) can be expressed as a linear combination of three basis tensors(ball, plate and stick) as in Eq. (5).

$$T = (\lambda_1 - \lambda_2)\vec{e}_1 \vec{e}_1^T + (\lambda_2 - \lambda_3)(\vec{e}_1 \vec{e}_1^T + \vec{e}_2 \vec{e}_2^T) + \lambda_3(\vec{e}_1 \vec{e}_1^T + \vec{e}_2 \vec{e}_2^T + \vec{e}_3 \vec{e}_3^T) \tag{5}$$

In Eq. (5), $(\vec{e}_1 \vec{e}_1^T)$ describes a stick (surface) with associated saliency $(\lambda_1 - \lambda_2)$ and normal orientation $\vec{e}_1$, $(\vec{e}_1 \vec{e}_1^T + \vec{e}_2 \vec{e}_2^T)$ describes a plate (curve) with associated saliency $(\lambda_2 - \lambda_3)$ and tangent orientation $\vec{e}_3$, and $(\vec{e}_1 \vec{e}_1^T + \vec{e}_2 \vec{e}_2^T + \vec{e}_3 \vec{e}_3^T)$ describes a ball (junction) with associated saliency λ_3 and no orientation preference.

For each pixel p_{valid} with a non-zero response (i.e. >0.001) in a response image I_{θ_i,ϕ_j}, a plate tensor T_p is created, having the minimum eigenvector $\vec{e_3}$ aligned to the orientation θ_i and the maximum eigenvector $\vec{e_1}$ aligned to the computed normal at the point, and associated eigenvalues $\lambda_1 \simeq \lambda_2 \simeq I^p_{\theta_i,\phi_j} and \lambda_3 = 0$, where $I^p_{\theta_i,\phi_j}$ is the response of pixel p in the response image $I_{\theta_i,\phi j}$.[5] Similarly, for each pixel $p_{invalid}$ with a zero response (i.e. $\leqslant 0.001$) in the response image $I_{\theta i,\phi j}$, a stick tensor T_p is created, having the minimum eigenvector $\vec{e_3}$ aligned to the orientation θ_i and the maximum eigenvector $\vec{e_1}$ aligned to the computed normal at the point, and associated eigenvalues $\lambda_1 = I^p_{(XYG)} and \lambda_2 \simeq \lambda_3 = 0$, where $I^p_{(XYG)}$ is the grayscale value of pixel p in the image I_{XYG}.[6] This is repeated for each response images; and for each pixel, 40 tensors are added together to form one resulting tensor as described in detail in Algorithm 1.

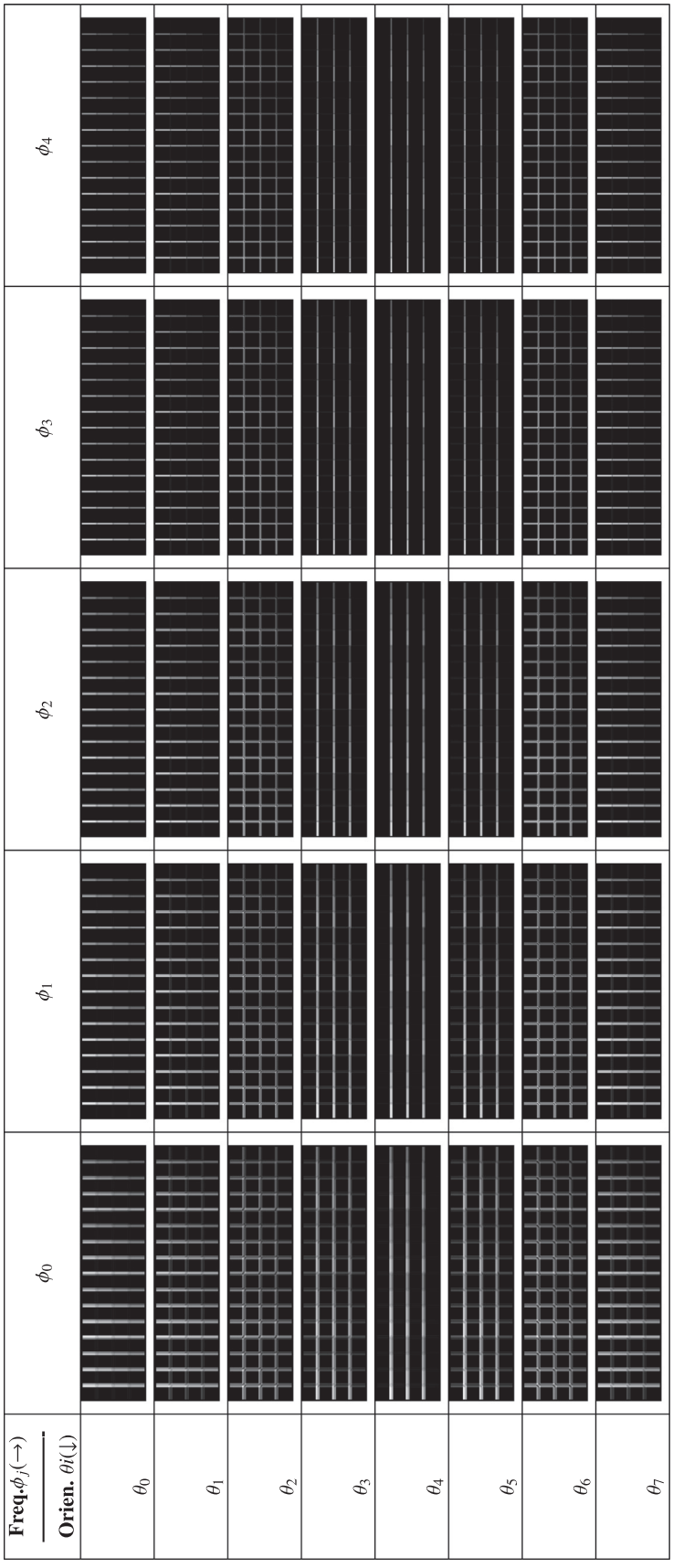
Using the tensor decomposition Eq. (5), all pixels for which $(\lambda_2 - \lambda_3) > \lambda_3$ are classified as part of curves with tangent orientation $\vec{e}_3$. Similarly all pixels for which $\lambda_3 > (\lambda_2 - \lambda_3)$ are classified as junction points with no orientation preference.

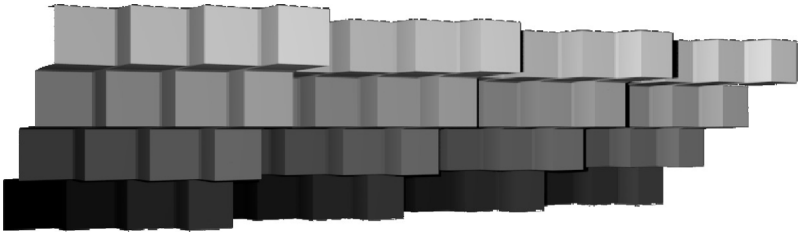
Fig. 6(a) shows the saliencies of each point for the image in Fig. 2(b). Fig. 6(b) shows the corresponding orientations

[5] The intensity responses $I^p_{\theta_i,\phi_j}$ for each pixel p are normalized and are in the range of [0,1].

[6] The grayscale values $I^p_{(XYG)}$ for each pixel p are normalized and are in the range of [0,1].

Table 1
The 40 Gabor response images corresponding to the image in Fig. 2(b). Each response image corresponds to a particular frequency and orientation. The horizontal axis represents the five different frequencies used. The vertical axis represents the eight different orientations used.

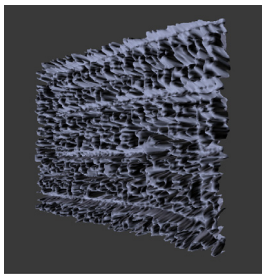




(a) The converted I_{XYG} image represented as a three-dimensional structure in color space.

(b) The original image in Figure 2(b) overlaid on the three-dimensional structure.

Fig. 3. Conversion of the input image to a three-dimensional structure.



(a) The converted I_{XYG} image represented as a three-dimensional structure in color space.

(b) Orthographic view of (a)

Fig. 4. Conversion of the input image in Fig. 12(a) to a three-dimensional structure.

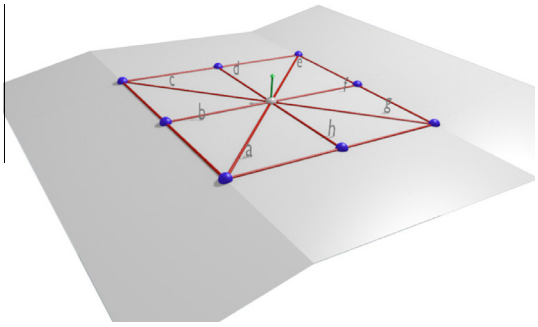
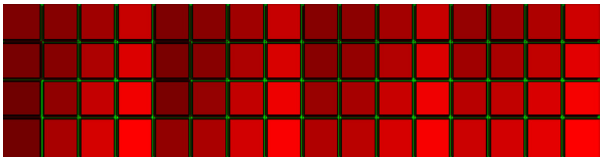
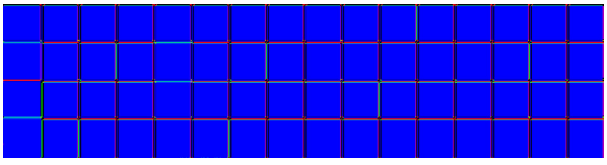


Fig. 5. Normal computation. The normal vector (blue) for each point P_i is computed as the normalized sum of the 8 cross products of the vectors $|\vec{a} \times \vec{b}|, |\vec{b} \times \vec{c}|, |\vec{c} \times \vec{d}|$, etc., as given by Eq. (2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Saliency map computed by decomposing the tensors calculated from the responses of Gabor Jets. Classified surface points are shown in red, curve points in green and joins in blue.



(b) Orientation map corresponding to the saliency map in Figure 6(a). The orientation is expressed as a three-dimensional vector. The axes X, Y, Z correspond to the colors R, G, B respectively. Negative values are not shown.

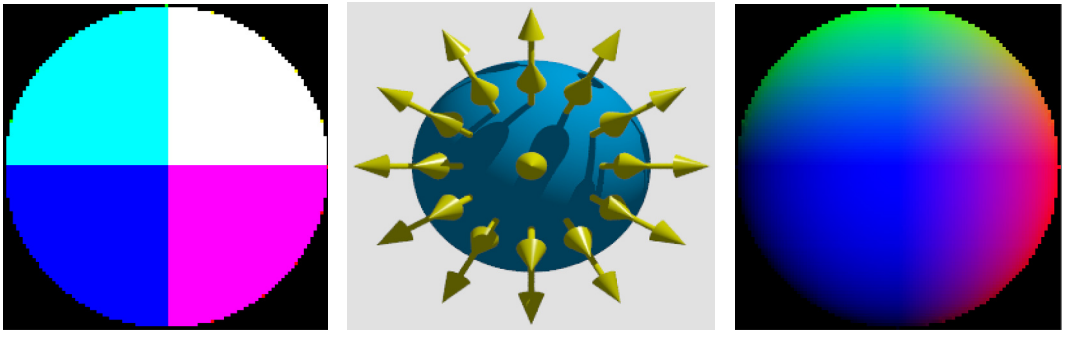
Fig. 6. Tensor decomposition. The saliency and orientation maps are the results of tensor decomposition. The meaning of the orientation varies according to the feature type passing through each point.

corresponding to the saliency map in Fig. 6(a). The meaning of the orientation varies according to the feature type passing through each point. Intuitively, if a pixel is classified as a curve i.e. $(\lambda_2 - \lambda_3) > (\lambda_1 - \lambda_2), (\lambda_2 - \lambda_3) > \lambda_3$ then the orientation represents the tangent $\vec{e}_3$ of the curve passing through that point. Similarly, if a pixel is classified as a surface then the orientation represents the normal $\vec{e}_1$ of the surface passing through that point; if a pixel is classified as a joint then the orientation is set to zero.

The result is the *initial labeling f* of the sample data which consists f a set of tensors, each one corresponding to the sum of the tensor-encoded responses of each pixel in the image.

Algorithm 1. Encode Gabor response images and Normals to tensors.

$T_f = \emptyset$ ▷ set of tensors $\Longleftrightarrow$ initial labeling f
for $p = 0 \rightarrow width \times height$ ▷ number of pixels
  $T_{p_{sum}} \leftarrow \{0\}$ ▷ $T_{p_{sum}}$: sum of plate tensors; initialize $T_{p_{sum}}$ corresponding to pixel p
  for $i = 0 \rightarrow O$ **do** ▷ O = number of orientations
    $\theta \leftarrow \frac{i}{8}\pi$
    for $j = 0 \rightarrow F$ **do** ▷ F = number of frequencies
      $\phi \leftarrow \frac{j}{5}(\frac{\pi}{32} - \frac{\pi}{8})$
      $\vec{e}_3 \leftarrow \langle \cos(\theta), \sin(\theta), 0 \rangle$ ▷ align $\vec{e}_3$ with the Gabor function's orientation
      $\vec{e}_1 \leftarrow \vec{N_p}$ ▷ align $\vec{e}_1$ with the point's normal
      $\vec{e}_2 \leftarrow \vec{e}_1 \times \vec{e}_3$ ▷ compute $\vec{e}_2$ from $\vec{e}_1$, $\vec{e}_3$ (ortho-normal basis)
      $v \leftarrow I^p_{i,j}$ ▷ get the response value for this pixel
      if $v \neq 0$ **then**
        $e_{val} \leftarrow v * \langle 1, 1, 0 \rangle$ ▷ scale the eigenvalues
        $T_p \leftarrow e_{val} * (\lambda_1 \vec{e}_1 \vec{e}_1^T + \lambda_2 \vec{e}_2 \vec{e}_2^T + \lambda_3 \vec{e}_3 \vec{e}_3^T)$ ▷ plate tensor for p (Eq. (3))
      else
        $v \leftarrow I^p_{XYG}$ ▷ get the grayscale value for this pixel
        $e_{val} \leftarrow v * \langle 1, 0, 0 \rangle$ ▷ scale the eigenvalues
        $T_p \leftarrow e_{val} * (\lambda_1 \vec{e}_1 \vec{e}_1^T + \lambda_2 \vec{e}_2 \vec{e}_2^T + \lambda_3 \vec{e}_3 \vec{e}_3^T)$ ▷stick tensor for p (Eq. (3))
      end if
      $T_{p_{sum}} \leftarrow T_{p_{sum}} + T_p$ ▷ Matrix addition
    end for
  end for
  $T_f \leftarrow T_f \bigcup T_{p_{sum}}$ ▷ Add to set of tensors
end for



(a) Depth map generated from the input radius r. The R,G,B channels correspond to the X, Y, Z axes. The center of the image is the origin. Negative values cannot be shown i.e. only the top-right quadrant contains all positive values for X,Y,Z.

(b) The 3D geometry corresponding to the XYZ map in (a). The yellow vectors indicate the normals at each point. Not all normals are shown for clarity.

(c) Based on the geometry in (b) the normal orientation at each point on the sphere's surface is computed. In our experiments, the radius is set to $r = 11$ which produces a constant $N = 81$ of usable normals.

Fig. 7. Computation of the normals based on the geometry of a Gaussian hemisphere.

A comparison with the method introduced in Poullis and You (2010) shows that a two step classification process was required because the tensor encoding was performed using only orientation information while completely ignoring color information: first a refinement and classification using the computationally expensive Tensor Voting and secondly, a segmentation using Graph-Cuts whose sole purpose was to incorporate back the ignored color information, along with the orientation information. In the proposed framework, these difficulties are overcome by incorporating all available information while encoding to tensors right from the beginning.

4.2. Derivation of the new labeling f'

As mentioned earlier, the extraction and simultaneous classification of image features is reformulated as a labeling problem, e.g. given an initial *large* set of labels, re-label them using a *small* set of new labels, subject to a cost function. In our case, the initial labels are usually equal to the number of pixels in the image and may contain *near* duplicates due to noise. In order to reduce the number of the initial labels and avoid having *near* duplicates, we perform the second phase of the tensor encoding and derive a new labeling f' in two steps:

1. The creation of a depth map based on a Gaussian hemisphere created with the input radius r as shown in Fig. 7(a). Based on the geometry of the Gaussian hemisphere shown in Fig. 7(b), the normal orientation at each point is computed similarly to the process explained in Section 4.1.3.
2. This information is used to compute a set of tensors at each point, as described below.

The new labeling is computed as a set of stick and plate tensors, representing surface and curve features respectively, which result

from encoding the points lying on the Gaussian hemisphere. For each point p with local normal orientation $\vec{n_p}$ lying on the surface of the Gaussian hemisphere, Θ plate tensors and $\Theta \times C$ stick tensors are created, representing possible surfaces and curves passing through that point. Θ is a predefined number of orientations which in our experiments is $\Theta = O$,[7] $0 \leqslant \Theta < \pi$ i.e. equals the number of orientations used in the computation of the Gabor Jets. C is a predefined number of gray levels which in our experiments is $C = 64, 0 \leqslant C \leqslant 255$. C is the quantization factor used to scale the eigenvalues of the stick tensors. The scaling is imperative since it allows the differentiation of two surface tensors with same orientation but different "height". Consider the example in Fig. 3(a). Almost all plateaus have the same orientation, hence a unit stick tensor can correspond to any one of the surfaces. Scaling the eigenvalues of the stick tensor allows us to differentiate between the several surfaces.

The total number of normals N resulting from the geometry of the Gaussian hemisphere depends on the radius r. In all reported results, the radius is set to $r = 11$ which results in $|T| = \Theta \times N + \Theta \times C \times N = \Theta \times N \times (1 + C)$ tensors. Thus, the new labeling f' consists of a set of scaled stick and plate tensors which are computed as explained in detail in Algorithm 2.

cut $C \subset E$ is a partition of the vertices V of the graph G into two disjoint sets S, T where $V_s \in S$ and $V_t \in T$. The cost of each cut C is the sum of the weighted edges $e \in C$ and is given by

$$|C| = \sum_{\forall e \in C} w_e \tag{6}$$

The minimum cut problem can then be defined as finding the cut with the minimum cost.

The binary case explained above can easily be extended to a case of multiple terminal vertices. We create a terminal vertex for each label tensor $T_i \in f'$ corresponding to each tensor in the new labeling tensor set. Thus the set of labels L has size $|L| = |T| = \Theta \times N \times (1 + C)$ and is defined to be $L = \{t_1, t_2, \ldots, t_{|T|}\}$.

Finding the minimum cut of a graph is equivalent to finding an optimal labeling $\phi : I_{XYG}^{P_i} \longrightarrow L$ which assigns a label $l \in L$ to each pixel $P_i \in I_{XYG}$ where ϕ is piecewise smooth and consistent with the original data. Thus, our energy function for the graph-cut minimization is given by

$$E(\phi) = E_{data}(\phi) + \kappa_1 * E_{smooth}(\phi) + \kappa_2 * E_{label}(\phi) \tag{7}$$

where κ_1 is the weight of the smoothness term and κ_2 the weight of the label term.

Algorithm 2. Encode Gaussian hemisphere normals to stick and plate tensors.

$T_{f'} = \emptyset$ ▷ set of tensors $\Longleftrightarrow$ new labeling f'
for $i = 0 \rightarrow N$ **do** ▷ N = normals on Gaussian hemi-sphere
  $\vec{e}_1 \leftarrow \vec{N}_i$ ▷ align $\vec{e}_1$ with the Gabor functions orientation
  for $j = 0 \rightarrow \Theta$ **do** ▷ Θ = number of orientations
    $\vec{e}_3 \leftarrow \langle \cos(\Theta_j), \sin(\Theta_j), 0 \rangle$
    $\vec{e}_2 \leftarrow \vec{e}_1 \times \vec{e}_3$ ▷ compute $\vec{e}_2$ from $\vec{e}_1, \vec{e}_3$ (ortho-normal basis)
    $e_{val} \leftarrow \langle 1, 1, 0 \rangle$ ▷ eigenvalues for unit plate tensor
    $P_{p_{\theta,\phi}} \leftarrow e_{val} * (\lambda_1 \vec{e}_1 \vec{e}_1^T + \lambda_2 \vec{e}_2 \vec{e}_2^T + \lambda_3 \vec{e}_3 \vec{e}_3^T)$ ▷ plate tensor (Eq. (3))
    $T_{f'} \leftarrow T_{f'} \bigcup P_{p_{\theta,\phi}}$ ▷ Add to set of tensors
    for $k = 0 \rightarrow C$ **do** ▷ C = number of gray levels
      $v \leftarrow \frac{C_k}{255}$
      $e_{val} \leftarrow v * \langle 1, 0, 0 \rangle$ ▷ scale the eigenvalues
      $S_{p_{\theta,\phi,C_k}} \leftarrow e_{val} * (\lambda_1 \vec{e}_1 \vec{e}_1^T + \lambda_2 \vec{e}_2 \vec{e}_2^T + \lambda_3 \vec{e}_3 \vec{e}_3^T)$ ▷ stick tensor (Eq. (3))
      $T_{f'} \leftarrow T_{f'} \bigcup S_{p_{\theta,\phi,C_k}}$ ▷ Add to set of tensors
    end for
  end for
end for

5. Optimization using Graph-Cuts

The extraction and simultaneous classification of image features is reformulated as a labeling problem. The initial labeling f consisting of the set of tensors encoded as described in Section 4.1 is to be relabeled using the new labeling f' consisting of a set of tensors encoded as described in Section 4.2. This problem can be efficiently solved using Graph-Cuts as explained below in Section 5.1.

5.1. Graph-Cuts

Given the input image I_{XYG} containing the pixels of the input image, an undirected graph $G = \langle V, E \rangle$ is created where each vertex $v_i \in V$ corresponds to a pixel $P_i \in I_{XYG}$ and each undirected edge $e_{i,j} \in E$ represents a link between neighboring pixels $P_i, P_j \in I_{XYG}$. In addition, two distinguished vertices called *terminals* V_s, V_t, are added to the graph G. An additional edge is also created connecting every pixel $P_i \in I_{XYG}$ and the two *terminal* vertices, e_{i,V_s} and e_{i,V_t}. For weighted graphs, every edge $e \in E$ has an associated weight w_e. A

In order for the energy function to be defined we need to first define a cost function $\Delta(t_i, t_j) \longrightarrow \alpha$ which will determine the cost (or similarity/dissimilarity) of two tensors t_i, t_j.

5.1.1. Design of cost function $\Delta(t_i, t_j)$

The cost function $\Delta(t_i, t_j)$ must be designed in such a way that it would possess the following desirable properties:

- In the case of two stick tensors t_i, t_j, the orientation difference $e_{max}^{(i,j)}$ i.e. the difference of the local surface normals $(e_{max}^{t_i}, e_{max}^{t_j})$ of the two tensors, and the eigenvalue differences $\lambda_{(1,2)}^{(i,j)}$ indicating the likelihood of the tensor being a surface, are compared, where,

$$\begin{aligned} e_{max}^{(i,j)} &= \| e_{max}^{t_i} - e_{max}^{t_j} \| \\ \lambda_{(1,2)}^{(i,j)} &= \| (\lambda_1^{t_i} - \lambda_2^{t_i}) - (\lambda_1^{t_j} - \lambda_2^{t_j}) \| \end{aligned} \tag{8}$$

- In the case of two plate tensors t_i, t_j, the orientation difference $e_{min}^{(i,j)}$ i.e. the difference of the local tangents $(e_{min}^{t_i}, e_{min}^{t_j})$ of the two tensors,and the eigenvalue differences $\lambda_{(2,3)}^{(i,j)}$ indicating the likelihood of the tensor being a curve, are compared where,

[7] O = number of orientations as first presented in Algorithm 1.

$$e^{(i,j)}_{min} = \|e^{t_i}_{min} - e^{t_j}_{min}\|$$
$$\lambda^{(i,j)}_{(2,3)} = \|(\lambda^{t_i}_2 - \lambda^{t_i}_3) - (\lambda^{t_j}_2 - \lambda^{t_j}_3)\| \quad (9)$$

- In the case of junctions, only the eigenvalue differences $\lambda^{(i,j)}_{(3)}$ are compared since there is no orientation preference, where,

$$\lambda^{(i,j)}_{(3)} = \|(\lambda^{t_i}_3 - \lambda^{t_j}_3)\| \quad (10)$$

- Large differences in the orientations of two neighboring (stick or plate) tensors are heavily penalized, except in the cases where the eigenvalues are close to being identical. This exception is necessary to overcome the problem occurring when the initial orientation and the new orientation vary slightly. This occurs whenever the new label orientations are fewer than the initial label orientations: in the case of the curves, there can be an arbitrary number of tangents whereas the Gabor functions are only tuned on eight orientations; similarly in the case of surfaces, there can be an arbitrary number of surface normals whereas the Gaussian hemisphere possibly does not (depending on the radius).
- Small differences in the orientations of two neighboring (stick or plate) tensors are favored, even in cases where the eigenvalues are not close to being identical. This property ensures that the alignment of the orientations has a higher precedence over the equality of the eigenvalues. For example if two neighboring plate tensors have the same orientation but different eigenvalues due to different values in the response images, we would still want to classify them as curves.

Taking into account the above desirable properties we define the comparison cost function $\Delta(t_i, t_j)$ as,

$$\Delta(t_i,t_j) = 3 - \begin{cases} e^{\left(\lambda^{(i,j)}_{(1,2)}\right)^{\|e^{(i,j)}_{max}\|^2}}, & \text{iff } t_i, t_j \in T_{stick} \\ e^{\left(\lambda^{(i,j)}_{(2,3)}\right)^{\|e^{(i,j)}_{min}\|^2}}, & \text{iff } t_i, t_j \in T_{plate} \\ e^{\left(\lambda^{(i,j)}_{(3)}\right)}, & \text{iff } t_i, t_j \in T_{ball} \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

where $T_{surface}, T_{curve}, T_{ball}$ are sets of surface, curve and ball tensors respectively. The eigenvectors and eigenvalues in Eq. (11) are the result of decomposing the two tensors t_i, t_j (Eq. (12)) based on the spectrum theorem in Eq. (5).

$$t_i \propto \langle \vec{e_{max}}^{t_i}, \vec{e_{mid}}^{t_i}, \vec{e_{min}}^{t_i} \rangle, \langle \lambda^{t_i}_1, \lambda^{t_i}_2, \lambda^{t_i}_3 \rangle$$
$$t_j \propto \langle \vec{e_{max}}^{t_j}, \vec{e_{mid}}^{t_j}, \vec{e_{min}}^{t_j} \rangle, \langle \lambda^{t_j}_1, \lambda^{t_j}_2, \lambda^{t_j}_3 \rangle \quad (12)$$

The three conditional cases encoded in the comparison cost function $\Delta(t_i, t_j)$ are shown in Fig. 8.

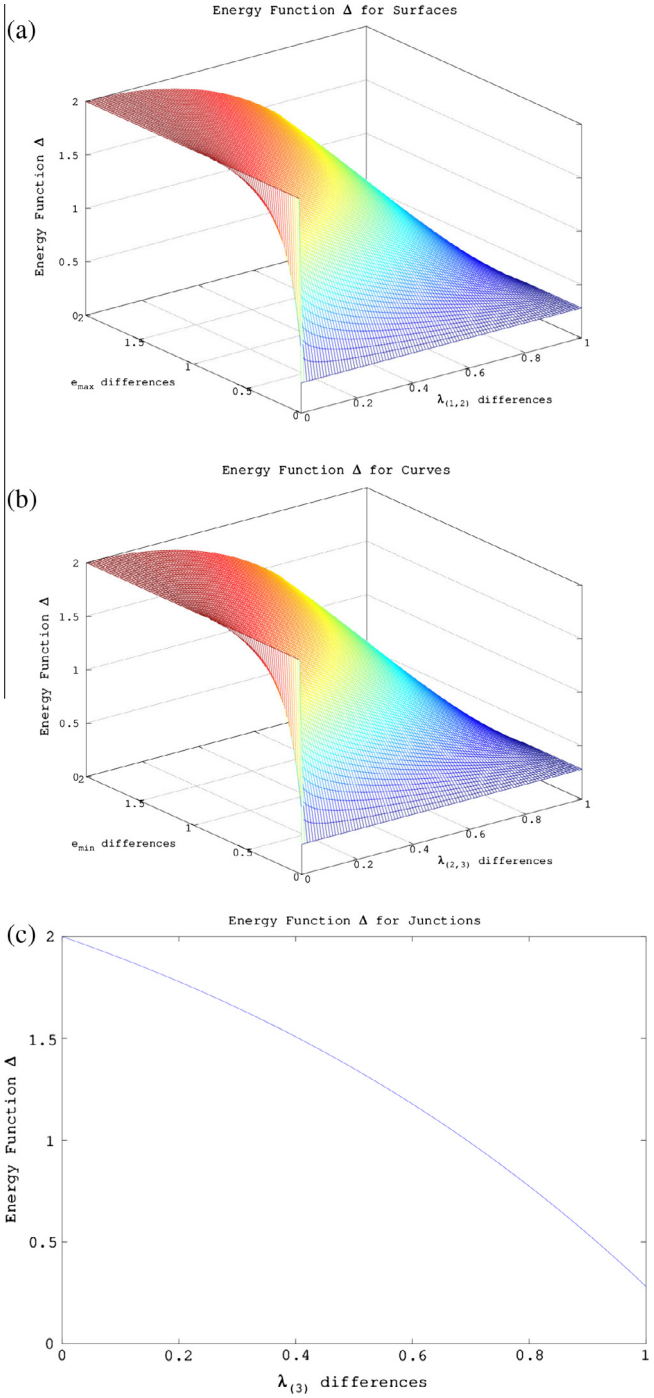


Fig. 8. Energy function Δ.

5.1.2. Implementation issues

The minimum and maximum eigenvalue and eigenvector differences, between two tensors t_i, t_j of the same feature type are computed as follows:

- The eigenvectors and eigenvalues are computed using Householder-QL algorithm (Press et al., 1986). The sign ambiguity is resolved by checking the sign of the sum of the signed inner products and determining the eigenvectors consistently i.e. if the sum of the signed inner product is negative then the eigenvectors are negated.
- The eigenvalues are normalized during the encoding to the tensorial representation and are in the range of $[0, 1]$. In the first case, during the encoding of the Gabor responses to tensors the computed tensors are scaled by the normalized intensity of the response v as shown in Algorithm 1. In the second case, during the encoding of the Gaussian hemisphere normals to tensors, the computed tensors are scaled by the grayscale value $v = \frac{C_k}{255}$ as shown in Algorithm 2. Thus, the minimum eigenvalue difference between two tensors t_i, t_j is $\lambda^{(i,j)} = 0$. Similarly, the maximum eigenvalue difference between two tensors t_i, t_j is $\lambda^{(i,j)} = 1$.
- The eigenvectors correspond to the Gabor filters' orientations and to the normals of the Gaussian hemisphere. The minimum occurs in the trivial case where the orientations are identical, hence the square of the magnitude of the minimum eigenvector difference between two tensors t_i, t_j is $e^{(i,j)} = 0$. Similarly, the square of the magnitude of the maximum eigenvector difference occurs when two orientation vectors are opposing, e.g.

$\langle 1,0\rangle$ and $\langle -1,0\rangle$. In this case, the square of the magnitude of the maximum eigenvector difference between two tensors t_i, t_j is $\|e^{(i,j)}\|^2 = 4$.

As previously mentioned the cost function $\Delta(t_i, t_j)$ was designed in a way such that the properties described in Section 5.1.1 would be encompassed. In addition to these properties, an exponential form was chosen for the cost function to address instability issues during the optimization by Graph-Cuts:

- Firstly, the exponential form ensures that the cost associated with each feature type will have the same geometrical representation. Not using the exponential form would implicitly introduce a bias towards a particular feature type, i.e. in this case the cost associated with surfaces $\left(\left(\lambda_{(1,2)}^{(i,j)}\right)^{\|e_{max}^{(i,j)}\|^2}\right)$ and curves $\left(\left(\lambda_{(2,3)}^{(i,j)}\right)^{\|e_{min}^{(i,j)}\|^2}\right)$ changes non-linearly whereas the cost associated with junctions changes linearly $\left(\lambda_{(3)}^{(i,j)}\right)$.
- Similarly, the exponential form ensures that the cost associated with each feature type have the same range of values. Not using the exponential form would again implicitly introduce a bias towards the feature type with the smallest values.

Hence, the use of this particular exponential form is necessary to overcome biases towards a particular feature type during optimization.

To demonstrate the use of the cost function, consider the following example where two tensors t_i, t_j are identical therefore, both the eigenvalue and eigenvector differences are zero, e.g. $\lambda^{(i,j)} = 0, e^{(i,j)} = 0$. In this case, the cost function $\Delta(t_i, t_j)$ reduces to the following,

$$\Delta(t_i,t_j) = 3 - \begin{cases} e^{(0)^0} = 3 - e^1 = 0.28, & \text{iff } t_i, t_j \in T_{stick} \\ e^{(0)^0} = 3 - e^1 = 0.28, & \text{iff } t_i, t_j \in T_{plate} \\ e^0 = 3 - e^0 = 2, & \text{iff } t_i, t_j \in T_{ball} \\ 0 = 3 - 0 = 3, & \text{otherwise} \end{cases} \tag{13}$$

If in the above example, the two tensors t_i, t_j had the same eigenvalue but maximally different eigenvector differences, e.g. $\lambda^{(i,j)} = 0, e^{(i,j)} = 4$ the cost function $\Delta(t_i, t_j)$ would instead become the following,

$$\Delta(t_i,t_j) = 3 - \begin{cases} e^{(0)^4} = 3 - e^0 = 2, & \text{iff } t_i, t_j \in T_{stick} \\ e^{(0)^4} = 3 - e^0 = 2, & \text{iff } t_i, t_j \in T_{plate} \\ e^0 = 3 - e^0 = 2, & \text{iff } t_i, t_j \in T_{ball} \\ 0 = 3 - 0 = 3, & \text{otherwise} \end{cases} \tag{14}$$

As it is evident, the function returns a large cost value if two tensors are similar and a small cost value otherwise. This is so, because if two tensors are similar we would want to penalize heavily for relabeling them differently, hence the large cost value. Similarly, if two tensors are not similar then we would not want to penalize a different relabeling, hence the small cost value. The cost values returned by the cost function are in the range $[0.28, 3]$. Table 2 shows how the comparison function varies according to the feature type passing through the pixel in question. The cost function $\Delta(t_i, t_j)$ is then used to define the energy functions $E_{data}(\phi), E_{smooth}(\phi), E_{label}(\phi)$ as follows.

Table 2
The comparison function varies according to the feature type passing through the pixel in question.

$\Delta(t_i,t_j)$	T_{stick}	T_{plate}	T_{joint}
T_{stick}	$3 - e^{\left(\lambda_{(1,2)}^{(i,j)}\right)^{\|e_{\bar{max}}^{t_i} - e_{\bar{max}}^{t_j}\|^2}}$	3	3
T_{plate}	3	$3 - e^{\left(\lambda_{(2,3)}^{(i,j)}\right)^{\|e_{\bar{min}}^{t_i} - e_{\bar{min}}^{t_j}\|^2}}$	3
T_{joint}	3	3	$3 - e^{\left(\lambda_{(3)}^{(i,j)}\right)}$

5.1.3. Energy data term $E_{data}(\phi)$

The data term provides a per-pixel measure of how appropriate a label $l \in L$ is, for a pixel $P_i \in I_{XYG}$ in the observed data and is given by,

$$E_{data}(\phi) = \sum_{P_i \in I} \Delta(P_i, \phi(P_i)) \tag{15}$$

where $\Delta(P_i, \phi(P_i)$ measures the difference between the existing label t_{P_i}, i.e. the tensor produced by the local Gabor response and normal orientation at the pixel P_i, and a new label $\phi(P_i) \in L$, as defined in Eq. (11).

5.1.4. Energy smoothness term $E_{smooth}(\phi)$

The smoothness term provides a measure of the difference between two neighboring pixels $P_i, P_j \in I_{XYG}$ with labels $l_i, l_j \in L$ respectively. Let t_{P_i} and t_{P_j} be the initial tensors of the neighboring pixels in the observed data $P_i, P_j \in I$ respectively. We define a prior measure of the *observed* smoothness between pixels P_i and P_j as $\Delta(P_i, P_j)$.

In addition, we define a measure of smoothness for the global minimization. Let $l_i = \phi(P_i)$ and $l_j = \phi(P_j)$ be the orientations under a labeling ϕ. We define a measure of the smoothness between neighboring pixels P_i, P_j under a labeling ϕ as $\widetilde{\Delta}(P_i, P_j) = \Delta(l_i, l_j)$

Using the smoothness prior defined for the observed data and the smoothness measure defined for any labeling ϕ we can finally define the energy smoothness term as follows,

$$E_{smooth}(\phi) = \sum_{(P_i,P_j)\in\aleph} V_{P_i,P_j}(\phi(P_i), \phi(P_j)) \tag{16}$$

$$E_{smooth}(\phi) = \sum_{(P_i,P_j)\in\aleph} \left[\sqrt{2} - e^{-\frac{(\Delta(P_i,P_j))^2}{2*\sigma^2}}\right] \widetilde{\Delta}(P_i, P_j) \tag{17}$$

$$E_{smooth}(\phi) = \sum_{(P_i,P_j)\in\aleph} K_{(P_i,P_j)} \Delta(l_i, l_j) \tag{18}$$

where $\aleph$ is the set of neighboring pixels, $K_{(P_i,P_j)} = \sqrt{2} - \left[e^{-\frac{\Delta(P_i,P_j)^2}{2\sigma^2}}\right]$ is the smoothness prior and gives an estimate of the smoothness between two neighbors, and σ controls the smoothness uncertainty. Since the cost function $\Delta(P_i, P_j)$ is strictly positive i.e. never zero or negative for any choice of P_i, P_j, it is convenient to express the smoothness prior $K_{(P_i,P_j)}$ as exponential. The exponential representation of $K_{(P_i,P_j)}$ is called the Boltzmann distribution and the smoothness uncertainty σ (corresponding to the temperature T term of the Boltzmann distribution) controls the variation in the smoothness between the two neighbors in the observed data. When dealing with Markov Random Fields, it is very common that the clique potentials are modeled in the form of $\phi_i(x_i) = e^{(}\Delta(xi))$, where $\Delta(x_i)$ is an energy function over values of x_i. This particular form is preferred whenever the opposite effect of the energy function is required i.e. a lower probability for high energy configurations and vice-versa.

Intuitively the meaning of the smoothness term is the following: if two neighboring pixels P_i and P_j have similar tensors in the observed data, then $\Delta(P_i, P_j)$ will be small, therefore $K_{(P_i,P_j)}$ will be high which means there will be a higher probability of $\widetilde{\Delta}(P_i, P_j)$ being small. In all our experiments the smoothness uncertainty is set to $\sigma = 0.25$.

5.1.5. Energy label term $E_{label}(\phi)$

The label term penalizes each unique label which appears under a labeling ϕ (Delong et al., 2012). We define the label term as follows,

$$E_{label}(\phi) = \sum_{l \in L} h_l . \zeta_l(\phi) \tag{19}$$

where h_l is a non-negative label cost of label l and set to $h_l = 1$ for all labels and, $\zeta_l(\phi)$ is a function which indicates whether a label is unique under a labeling ϕ and is defined as,

$$\zeta_l(f) = \begin{cases} 1, & \exists P : \phi = l \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Intuitively, the label term penalizes each unique label that exists under the labeling ϕ i.e. it favors a minimal set of labels.

Finally, the energy function $E(\phi)$ in Eq. (7) penalizes heavily for variations between neighboring pixels of similar geometric type (and orientation in the case of surfaces and curves), and vice versa, which results in a better defined segmentation of the image. Moreover, it penalizes heavily any unique labels which have not been assigned to any pixels thus eliminating unnecessary tensor labels as we demonstrate in Section 7.

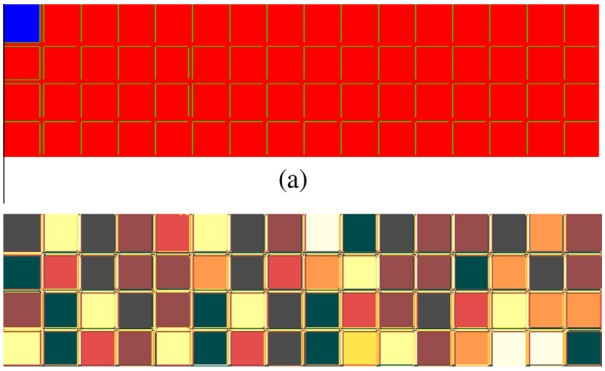
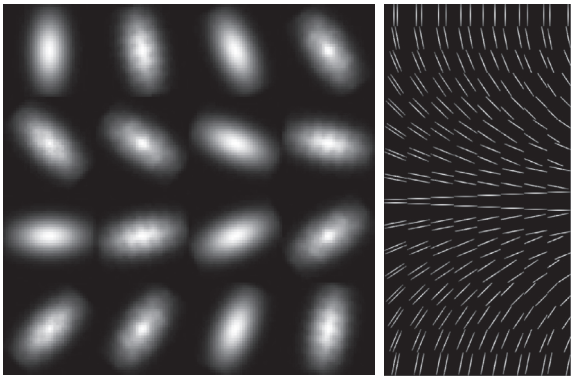


Fig. 9. (a) The feature types are color-coded using three colors. Red indicates a surface, green indicates a curve and blue indicates neither i.e. a junction. (b) The resulting, color-coded clusters. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Single mode Gaussian-based kernels.

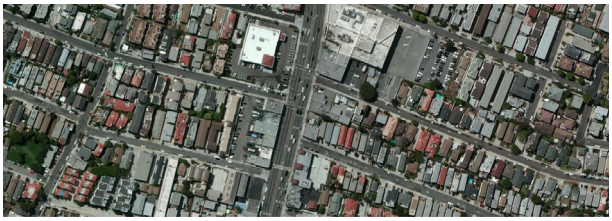
(b) Bi-modal Gaussian-based kernels.

Fig. 10. The single mode and bi-modal Gaussian-based kernels.

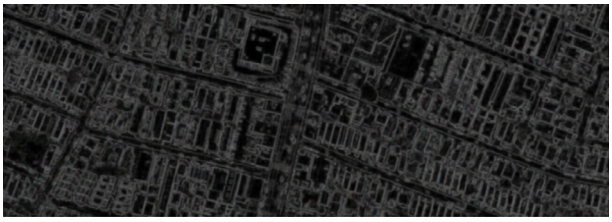
5.1.6. Energy minimization using Graph-Cuts

The energy minimization is performed using the α-expansion algorithm. For each iteration, the algorithm selects a tensor label $\alpha \in L$, and then finds the best configuration within this α-expansion move. If the new configuration reduces the overall energy $E(\phi)$ in Eq. (7), the process is repeated. The algorithm stops if there is no α that decreases the energy. The α expansion algorithm requires that the user-defined energy is regular and thus graph-representable (Kolmogorov, 2004). Kolmogorov also proves that any class F^2 functions of one variable are regular, hence the label term $E_{label}(f)$ in Eq. (19) is regular.

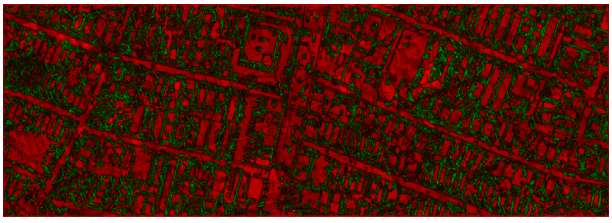
Fig. 9(a) shows the result of the application of the optimization on the simple case shown in Fig. 2(b). As mentioned earlier, each pixel is classified as a surface (shown in red), curve (shown in



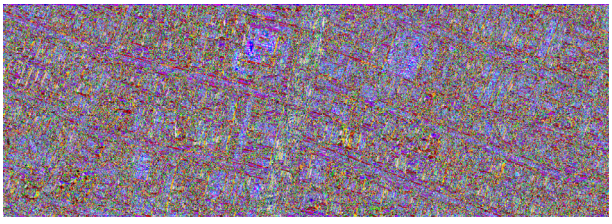
(a) Input image.



(b) The sum of Gabor responses for the input image.



(c) The resulting saliency map.



(d) The orientation map corresponding to the saliency map in (c) according to their feature type.

(e)

Fig. 11. Test case 1: An example of a complex urban satellite image with roads of varying width.

green) or neither (shown in blue) i.e a junction. Fig. 9(b) shows the resulting clusters. In this example, 57 different clusters were identified. It should be noted that although the number of new labels used is sufficient for this type of data, the time required for the segmentation and classification is directly dependent on this number. Therefore, considerably increasing the number of new labels will result in considerable computational time required. The time required for the sample image was 36 s.

6. Road extraction

The result of the Tensor-Cuts segmentation is a set of features of multiple types and their associated orientation preference, in the cases of surfaces and curves.

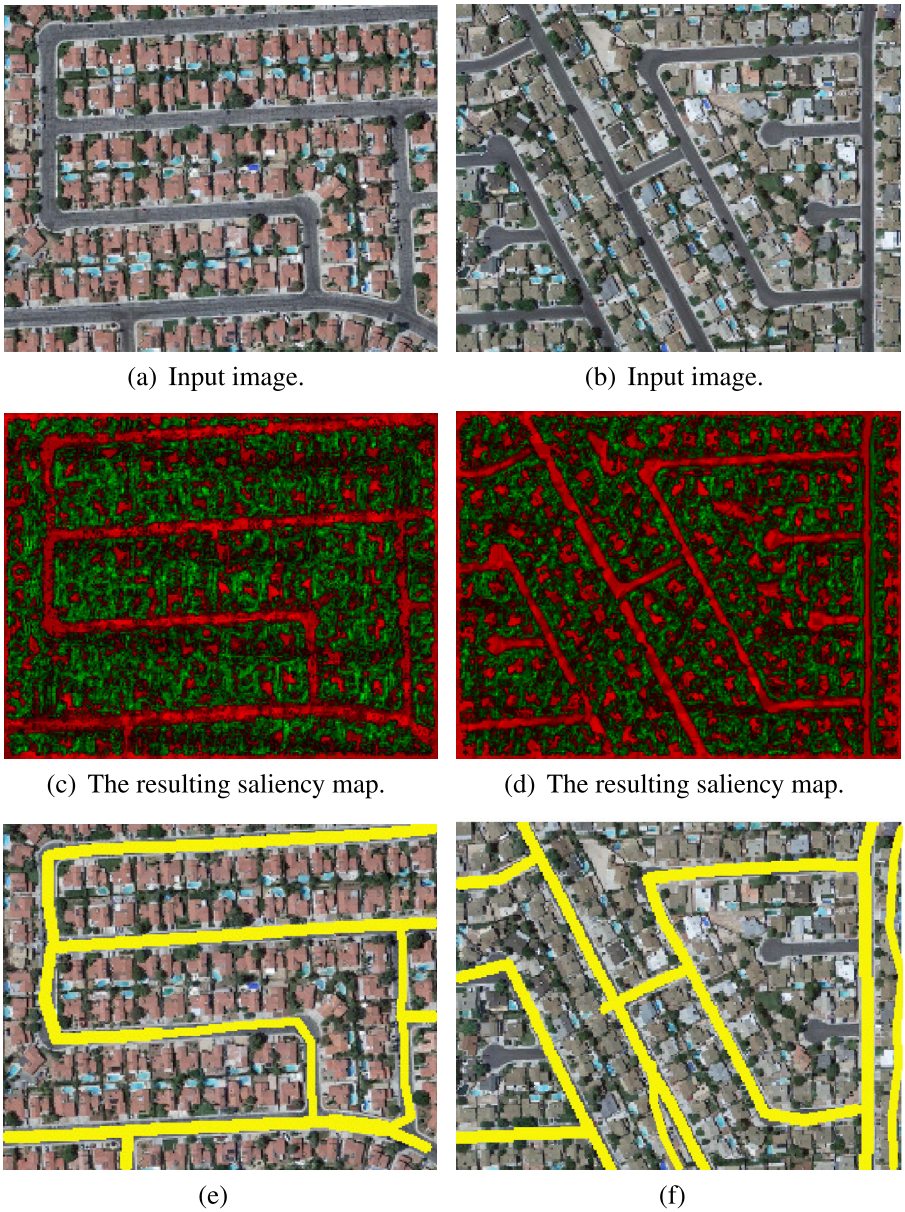
6.1. Road model

In order to proceed with the processing we first have to define the road model. The road model is defined in terms of various geometric characteristics and its function as follows:

1. A road is a curvilinear structure.
2. A road has width and orientation which may locally vary smoothly.
3. The area of the road is of uniform color.
4. The length is always larger than the width.
   This assumption is true even in local small patches as long as the size of the patch is larger than the width.
5. All roads must lead to somewhere.
   This means that when considering a satellite image there has to exist at least one road intersecting the boundaries of the image.

6.2. Detection of candidate road center-points

A similar road model to the above was used by Poullis and You (2010). In their work the authors developed a set of single-mode and bi-modal Gaussian-based kernels in order to measure the likelihood of an image pixel being a road center-point. We improve the efficiency of their approach by selectively applying the set of single-mode and bi-modal Gaussian-based kernels of different orientations and widths. The variable widths and orientations account



(a) Input image. (b) Input image.

(c) The resulting saliency map. (d) The resulting saliency map.

(e) (f)

Fig. 12. Urban area test case 2 (left column) and test case 3 (right column).

for the characteristic that roads can have locally varying width and orientation.

Fig. 10 shows the set of single and bi-modal Gaussian-based kernels. In our experiments, the number of orientations is set to sixteen and the number of widths to eight in the range of $[0, 2\pi)$ and $[10px, 60px)$, respectively. The bi-modal kernel is designed in such a way that pixels classified as road candidates will respond to it if they are aligned with the center line, having two parallel lines off their axis; corresponding to the side-walks. Similarly, the single-mode kernel is designed in such a way that pixels classified as road candidates will respond to it if they are aligned with the center line, having neighboring candidate road pixels *between* the parallel lines.

We exploit the characteristic that roads must lead to somewhere and that at least one road will intersect the boundaries of the image. Hence, instead of applying the kernels to all road candidates (Poullis and You, 2010), we selectively apply the kernels only on the image boundaries. Intuitively, the kernels are used to find an *entry* point to the image from which any traditional template fitting technique can be applied for road tracking. This considerably reduces the computational time needed. In the experiments shown, we use a brute-force template matching variant of the approach presented in Zhao and You (2012) and Poullis et al. (2008).

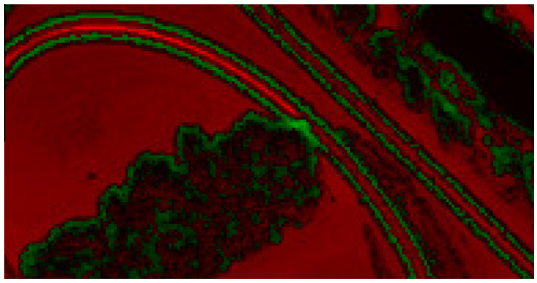
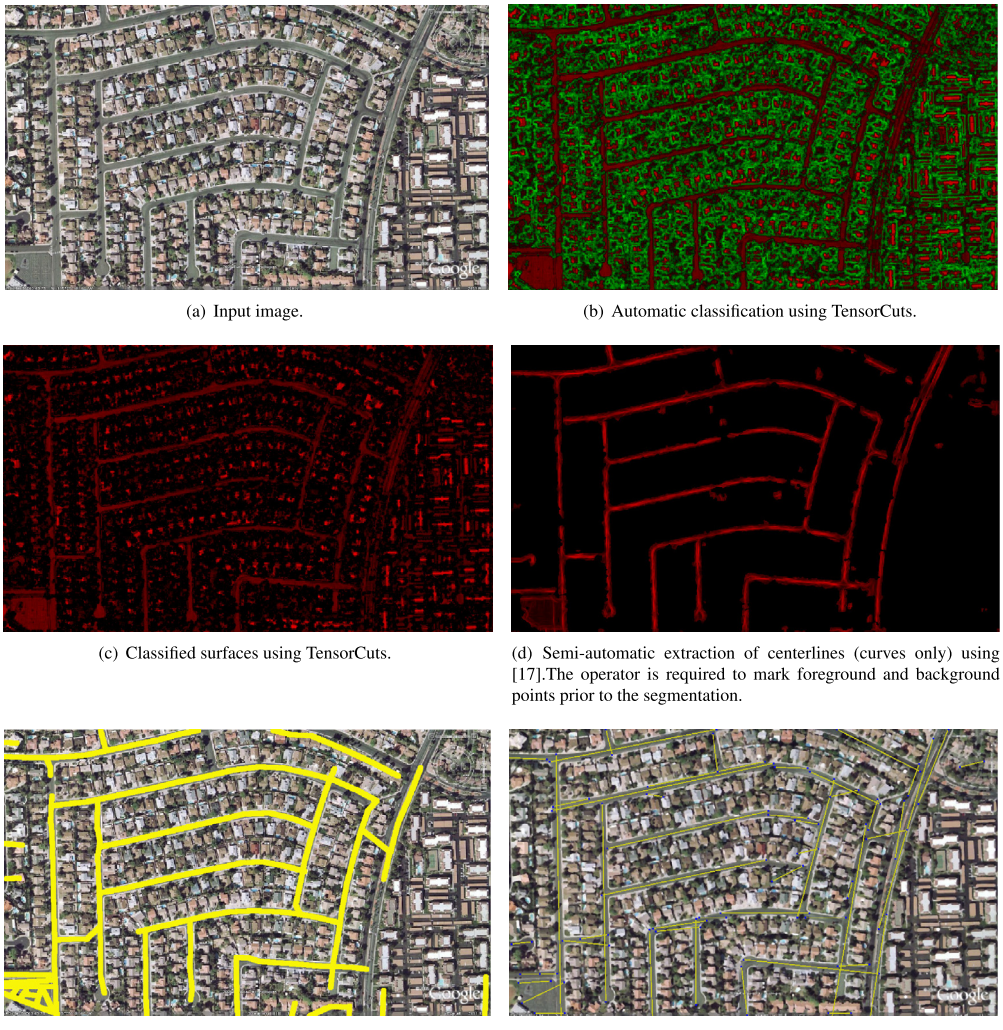


Fig. 14. Close up of Fig. 15(a). The Gaussian-based kernels respond only on pixels that are in the middle of parallel classified curves (i.e. green) and whose in-between area is classified as a surface (i.e. red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Experimental results

We have extensively tested the proposed method and report the results. For all the experiments shown the parameters controlling the smoothness and label cost terms in Eq. (7) are $\kappa_1 = 60$ and $\kappa_2 = 40$.



(a) Input image.

(b) Automatic classification using TensorCuts.

(c) Classified surfaces using TensorCuts.

(d) Semi-automatic extraction of centerlines (curves only) using [17].The operator is required to mark foreground and background points prior to the segmentation.

(e) Result using TensorCuts.

(f) Result using [17].

Fig. 13. Urban area test case 4. Comparison with Poullis and You (2010).

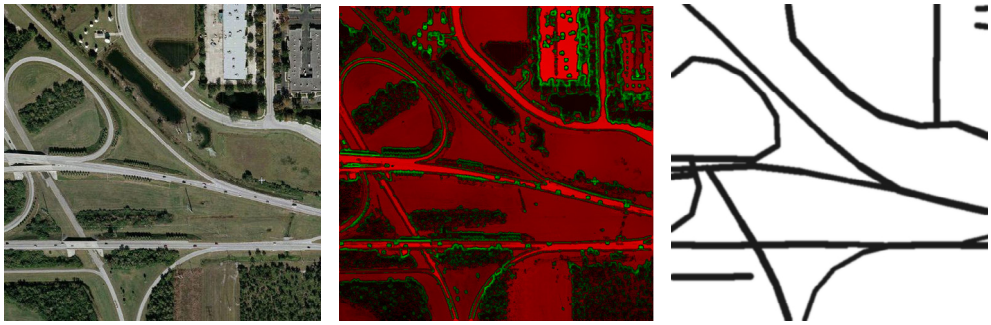
Fig. 11(a) shows an example satellite image. The sum of the response images resulting from the application of the bank of Gabor jets is shown in Fig. 11(b). The resulting saliency map resulting is shown in Fig. 11(c). The orientation map corresponding to the saliency map in Fig. 11(c) according to their feature type. For example, if a pixel is classified as part of a surface then the orientation represents the normal to that surface. Similarly, if a pixel is classified as part of a curve then the orientation represents the tangent to that curve. Pixels classified as junctions do not have any orientation preference, hence the orientation map contains no value.

The saliency map is further used in the extraction of the center-points. The single-mode Gaussian-based kernel is applied on the surface pixels of the image, i.e. red channel and the bi-modal Gaussian-based kernel is applied on the curve pixels of the image, i.e. green channel. The advantage of the approach is that even in the case where the road is not clustered in its entirety it can still be tracked if the road part closest to the image boundaries is detected. Once a candidate center point is identified on the boundaries, an iterative template fitting process such as Zhao and You (2012) is used to track the road network. In the case where no other roads can be tracked, the process is repeated and another candidate boundary center-point is identified using the aforementioned method.

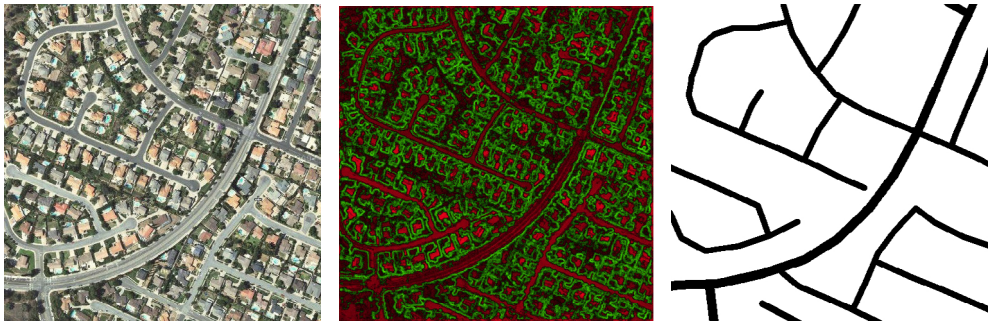
Fig. 11(e) shows the resulting road network extracted with the aforementioned technique. All roads are successfully extracted except from the main road. This is due to the range of the widths of the single and bi-modal Gaussian-based kernels. Adjusting the width to include all the various road sizes occurring in an image can resolve this problem, however it will increase the computational time. The time required for the segmentation and classification step is 00:01:26 s. The image size is 1297×454.

Fig. 12(a) shows an urban area. The resulting saliency map is shown in Fig. 12(c) and the result after the application of the road extraction is shown in Fig. 12(e). There were 31 clusters returned. As it is evident in this case, all roads have been successfully extracted.

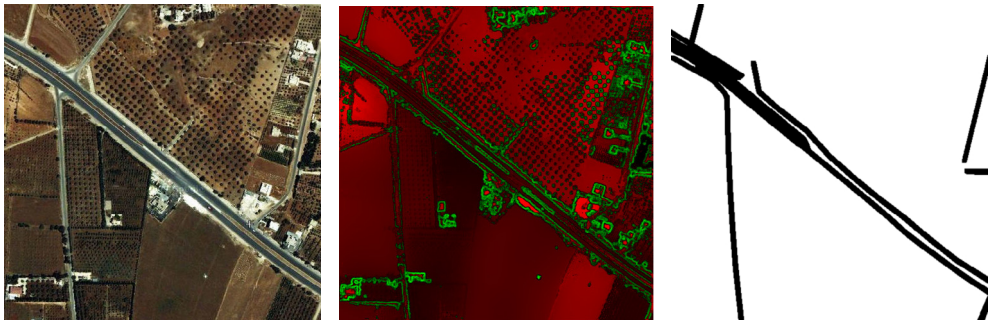
Similarly, Fig. 12(b) shows another urban area. The resulting saliency map is shown in Fig. 12(d) and the result after the application of the road extraction is shown in Fig. 12(f). In this case, 87 clusters



(a) Input image (Developed Sub-urban) (b) Feature types classification (c) Road network



(d) Input image (Developed Urban) (e) Feature types classification (f) Road network



(g) Input image (Emerging Suburban) (h) Feature types classification (i) Road network

Fig. 15. Images from the dataset in Das et al. (2011). (a) Input image. (b) Classification into the three feature types. (c) Extracted road network.

were created. In this example, it can be seen that in addition to the actual roads, houses are also erroneously extracted. This is due to the fact that the houses in this case have very close proximity to each other and also have the similar appearance as the roads.

Fig. 13(a) shows a satellite image of an urban area in Las Vegas, USA. The result of the automatic classification using TensorCuts is shown in Fig. 13(b) and the classified surfaces are shown in Fig. 13(c). The semi-automatic extraction of the centerlines i.e. curves using the method introduced in Poullis and You (2010) is shown in Fig. 13(d). The final results of the two methods are shown in Fig. 13(e) and (f), respectively. As expected, the semi-automatic method slightly outperforms the automatic method presented in this paper. However, user interaction is required at the beginning: the user is required to mark sample areas of foreground and background "objects" in order for the segmentation to succeed. In our case, there is no user interaction whatsoever and the resulting saliency map includes not just curves but also surfaces and junctions. Both the final results, seem to verify that the success of any algorithm in this context is strongly coupled with the successful result of the segmentation algorithm.

Fig. 15 shows the results for images of the dataset in Das et al. (2011). The images are of low resolution and depict road networks in different settings, i.e. of a developed suburban, developed urban and emerging suburban area. Fig. 15(b), (e), and (h) show the results of the application of TensorCuts. As it is evident, the classifications for the images in Fig. 15(a) and (g) contain many surfaces since there are large areas of smooth intensity variations. However, the Gaussian-based kernels respond only on points which lie between parallel classified curves whose in-between area is classified as a surface as shown in Fig. 14.

Despite the low resolution of the satellite images, the proposed method performs very well with a more than 95% completeness in all test cases. Although admittedly the method proposed in Das et al. (2011) slightly outperforms the proposed method in the test case of Fig. 15(d) and of Fig. 15(g), it should be noted that it uses at least *14* user-defined parameters in order to achieve these results; as reported in Das et al. (2011) – 3 these parameters are empirically obtained and any changes in the input data will require the parameters to be empirically tuned again for optimal parameter selection. As previously mentioned in Section 1, one of the primary motivations of the proposed method is the elimination (or at least reduction to a minimum) of the data-dependent thresholds.

7.1. Evaluation

The proposed method was evaluated in terms of the following well-established metrics as introduced by Wiedemann and Hinz (1999):

- *Completeness:* the ratio of the true positives over the sum of the true positives and false negatives.
- *Correctness:* the ratio of the true positives over the sum of the true and false positives.
- *Quality:* the ratio of the true positives over the sum of the true and false positives and false negatives.

The above parameters, i.e. true positives, false positives and false negatives are determined based on existing geographical databases. In cases where no such information is available, we manually indicate the ground truth. The results are shown in Tables 3 and 4. As it can be seen, the algorithm performs quite well in non-very complex cases such as the images from the dataset in Das et al. (2011) shown in Fig. 15 however, in complex cases where the buildings and the road network exhibit the same reflectance properties, it seems not to perform as well.

Table 3
Quantitative evaluation results for the test cases shown in this paper, in terms of the three evaluation metrics: completeness, correctness, and quality.

Image	Completeness	Correctness	Quality
Test case 1	72.2	62.1	51.6
Test case 2	94.3	79.2	81.5
Test case 3	82.9	74.3	76.8
Test case 4	68.1	64.3	61.9

Table 4
Quantitative evaluation results for the test cases shown Fig. 15, in terms of the three evaluation metrics: completeness, correctness, and quality.

Image	Completeness	Correctness	Quality
Developed suburban	99.6	96.4	95.3
Developed urban 2	96.2	98.3	94.6
Emerging suburban 3	95.7	96.4	92.4

8. Conclusion

We have presented a framework for simultaneous segmentation and classification of image features, called Tensor-Cuts. The unique characteristic of the proposed framework is that it requires *no thresholds*. This is due to the fact that *all* information is encoded using a tensorial representation and optimized using Graph-Cuts.

Moreover, we have presented how this framework is particularly suitable for applications in remote sensing and in particular for pre-processing satellite images for road extraction, since they contain mostly linear features. The proposed framework has been extensively tested on images of various contexts and the results are very promising.

References

Bacher, U., Mayer, H., 2005. Automatic road extraction from multispectral high resolution satellite images. In: Proceedings of CMRT05.

Das, S., Mirnalinee, T., Varghese, K., 2011. Use of salient features for the design of a multistage framework to extract roads from high-resolution multispectral satellite images. IEEE Trans. Geosci. Rem. Sens. 49, 3906–3931.

Daugman, J.G. et al., 1985. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. Opt. Soc. Am., J., A: Opt. Image Sci. 2, 1160–1169.

Delong, A., Osokin, A., Isack, H.N., Boykov, Y., 2012. Fast approximate energy minimization with label costs. Int. J. Comput. Vis. 96, 1–27, http://dx.doi.org/10.1007/s11263-011-0437-z, doi:http://dx.doi.org/10.1007/s11263-011-0437-z.

Doucette, P., Agouris, P., Stefanidis, A., Musavi, M., 2001. Self-organised clustering for road extraction in classified imagery. ISPRS J. Photogram. Rem. Sens. 55, 347–358.

Grote, A., Heipke, C., 2008. Road extraction for the update of road databases in suburban areas. Arch. Photogram., Rem. Sens. Spat. Inform. Sci., 563–568.

Hinz, S., Baumgartner, A., 2003. Automatic extraction of urban road networks from multi-view aerial imagery. ISPRS J. Photogram. Rem. Sens. 58, 83–98.

Hubel, D.H., 1982. Exploration of the primary visual cortex, 1955–78. Nature 299, 515–524.

Hubel, D.H., Wiesel, T.N., 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J. Physiol. 160, 106.

Hubel, D.H., Wiesel, T.N., 1974. Sequence regularity and geometry of orientation columns in the monkey striate cortex. J. Compar. Neurol. 158, 267–293.

Jones, J.P., Palmer, L.A., 1987. An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. J. Neurophysiol. 58, 1233–1258.

Kolmogorov, Zabih, 2004. What energy functions can be minimized via graph cuts. IEEETPAMI: IEEE Trans. Patt. Anal. Mach. Intell. 26.

Lacoste, C., Descombes, X., Zerubia, J., 2005. Point processes for unsupervised line network extraction in remote sensing. IEEE Trans. Patt. Anal. Mach. Intell. 27, 1568–1579.

Medioni, G., Tang, C.K., Lee, M.S., 2000. Tensor voting: theory and applications. In: Proceedings of RFIA. Paris, France.

Mena, J.B., Malpica, J.A., 2005. An automatic method for road extraction in rural and semi-urban areas starting from high resolution satellite imagery. Patt. Recog. Lett. 26, 1201–1220.

Mnih, V., Hinton, G.E., 2010. Learning to detect roads in high-resolution aerial images. In: Computer Vision–ECCV 2010. Springer, 2010, pp. 210–223.

Poullis, C., You, S., 2010. Delineation and geometric modeling of road networks. ISPRS J. Photogram. Rem. Sens. 65, 165–181.

Poullis, C., You, S., Neumann, U., 2008. A vision-based system for automatic detection and extraction of road networks. In: IEEE Workshop on Applications of Computer Vision, 2008, WACV 2008. IEEE, pp. 1–8.

Poz, A., Gallis, R.A., da Silva, J.F., Martins, E.F., 2012. Object-space road extraction in rural areas using stereoscopic aerial images. IEEE Geosci. Rem. Sens. Lett. 9, 654–658.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., 1986. Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, Cambridge, UK.

Wegner, J.D., Montoya-Zegarra, J.A., Schindler, K., 2013. A higher-order CRF model for road network extraction. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013. IEEE, pp. 1698–1705.

Wiedemann, C., Hinz, S., 1999. Automatic extraction and evaluation of road networks from satellite imagery. In: ISPRS Congress, pp. 95–100.

Zhao, J., You, S., 2012. Road network extraction from airborne lidar data using scene context. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2012. IEEE, pp. 9–16.