VCells: Simple and Efficient Superpixels Using Edge-Weighted Centroidal Voronoi Tessellations

Jie Wang, *Student Member*, *IEEE*, and Xiaoqiang Wang

Abstract—VCells, the proposed Edge-Weighted Centroidal Voronoi Tessellations (EWCVTs)-based algorithm, is used to generate superpixels, i.e., an oversegmentation of an image. For a wide range of images, the new algorithm is capable of generating roughly uniform subregions and nicely preserving local image boundaries. The undersegmentation error is effectively limited in a controllable manner. Moreover, VCells is very efficient with core computational cost at $\mathcal{O}(K\sqrt{n_c \cdot N})$ in which K, n_c, and N are the number of iterations, superpixels, and pixels, respectively. Extensive qualitative discussions are provided, together with the high-quality segmentation results of VCells on a wide range of complex images. The simplicity and efficiency of our model are demonstrated by complexity analysis, time, and accuracy evaluations.

Index Terms—Superpixels, k-means, centroidal Voronoi tessellations, image segmentation, image labeling, clustering.

1 Introduction

SUPERPIXELS are an oversegmentation of an image. The superpixel concept was originally developed by Ren and Malik [20], and suggests that small image segments obtained by the pixel grouping process might be a more natural and perceptually meaningful representation of reality. Because the superpixel graph is obtained by contracting and grouping the pixel graph, computation costs can be greatly reduced, thereby achieving an important, practical advantage over other, more traditional methods of image segmentation.

Region segmentation algorithms used to generate superpixels generally fall into two categories—those that have no constraint on compactness and those that embed certain constraints on compactness. Algorithms that have no compactness constraint—such as Mean Shift [3], local variation [9], or watershed [26] algorithms—are fast. Due to the lack of a compactness constraint, however, these segmentation algorithms may lead to large undersegmentation errors when the image has poor contrast or shadows. An example of irregular boundaries and shapes is shown in Figs. 1d, 1e, and 1f.

The second category of superpixel algorithms embed certain compactness constraints. Normalized Cuts (N-Cuts) [20], [30], TurboPixels [15], and the VCells proposed in this paper are algorithms of this type. Based on the N-Cuts algorithm (e.g., [23], [22]), Ren and Malik proposed a superpixel algorithm [20] that partitions the image into a large number of small, homogeneous regions. As one of the most popular and widely used graph cut algorithms, N-Cuts is very powerful in feature extraction and visual perceptual grouping. The high computational cost, however, prohibits its use in high superpixel density or megapixel sized images [15]. In addition, the speedup strategies (e.g., [5], [21]) often sacrifice the flexibility of the algorithm and/or the quality of the resulting superpixels. It may be hard to control the number of superpixels and they may not be uniform in size, shape, and other qualities.

More recently, Levinshtein et al. proposed a very efficient algorithm called TurboPixels [15] to generate superpixels. The TurboPixels algorithm segments an image into a set of lattice-like structures. It is the first attempt and successful application to solving superpixel problems by using geometric flow ideas. When the TurboPixels algorithm is used, superpixel accuracy is comparable to N-Cuts, but because the computational cost increases linearly in proportion to the number of pixels, computational cost is significantly less.

Due to the increasing application of superpixels in the past few years [1], some improved algorithms for superpixels have been developed [24], [4], [19], [25], [1]. Quick shift [24], for example, is a variant of Mean Shift, but is more practical and efficient. Couprie et al. proposed combining graph-cuts, random walker, and optimal spanning forest under a unified framework [4]. Suggested ideas [19], [25] are based on graph-cuts, but lattice-cut algorithms pose restrictions on superpixel shapes [25]. Achanta et al. [1] proposed a linear iterative clustering algorithm leading to relatively lower computational cost.

In this paper, we propose a novel approach to solve superpixel problems based on a generalized concept of Centroidal Voronoi Tessellations (CVTs) (e.g., [27], [6], [7], [11], [13], [8], [14]). In simple cases, CVTs are the same as k-means clustering, an algorithm that is widely known and easily implemented [6]. However, because of the lack of compactness constraint, classic CVT-based algorithms are very sensitive to image "noise" (e.g., [8], [27]). To overcome this limitation, we developed Edge-Weighted Centroial Voronoi Tessellations (EWCVTs) [27], [28] by introducing a new energy term related to the clusters' boundary length. Essentially, limiting the boundary length is a compactness constraint. The EWCVT algorithm has been successfully used for general image segmentation purposes [27]. However, the segments produced by EWCVT may lack spatial connectivity. Therefore, we developed special mechanisms named "Looking-Nearest-Neighbors" (LNN) and "Detecting-Segment-Breaking" (DSB) to ensure the resulting segments are spatially connected.

We named our algorithm Voronoi-Cells, or VCells. Compared with existing superpixel algorithms, our algorithm has several desirable advantages. First, the boundaries of superpixels are naturally formed during the iteration process. This eliminates concerns about boundary crossing, collision, and other qualities of the region-grow methods. Second, VCells is very efficient. Third, the accuracy of our algorithm is comparable to N-Cuts and TurboPixels. Finally, the compactness constraint, i.e., the boundary energy term in our algorithm, is controllable, and thus leads to greater flexibility.

In Section 2, we give a detailed explanation and discussion of the VCells algorithm. Accuracy and performance evaluations are presented in Section 3, together with numerical examples, and in Section 4, we give some concluding remarks.

2 The VCells SuperPixel Algorithm

The VCells superpixel algorithm has two stages. In the first stage, the image is divided into small segments of uniform size and shape; the second stage is used to apply EWCVT-LNN to these segments. We will first introduce some basic notations before moving into a more detailed discussion of these two stages.

2.1 Basic Notations

For a digital image, let $\mathcal{D}$ denotes the set of pixels $\mathbf{p}$ and $u(\mathbf{p})$ the gray levels or colors. For each pixel $\mathbf{p}$, we denote $\mathbb{N}_\omega(\mathbf{p})$ the collection of pixels inside the disk whose radius is ω and centered

• *J. Wang and X. Wang are with the Department of Scientific Computing, Florida State University, Tallahassee, FL 532306-4120. E-mail: jiewangustc@gmail.com, wwang3@fsu.edu.*

Manuscript received 14 Nov. 2011; accepted 18 Jan. 2012; published online 30 Jan. 2012.
Recommended for acceptance by M. Brown.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2011-11-0821.
Digital Object Identifier no. 10.1109/TPAMI.2012.47.

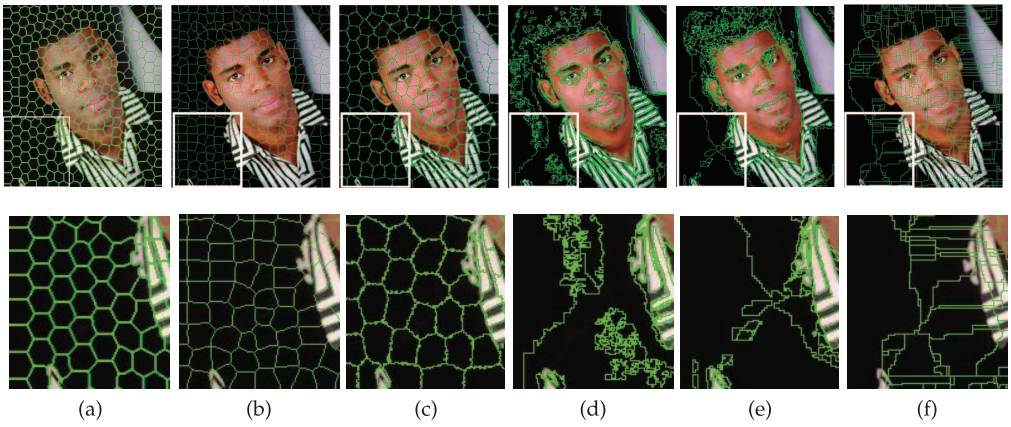

Fig. 1. Superpixels obtained by (a) VCells, (b) TurboPixels, (c) N-Cuts, (d) Local variation, (e) Mean shift, (f) WaterShed. Images (b) to (f) are from [15].

at $\mathbf{p}$. If pixel $\mathbf{q}$ is inside $\mathbb{N}_\omega(\mathbf{p})$, we call $\mathbf{q}$ a "*neighbor pixel*" of $\mathbf{p}$. Moreover, if the spatial distance between pixel $\mathbf{p}$ and its neighbor pixel $\mathbf{q}$ is 1, we call $\mathbf{q}$ a "*direct neighbor pixel*" of $\mathbf{p}$. Clearly, the direct neighbor pixels of $\mathbf{p}$ are the four adjacent pixels.

Suppose, we have a digital image $\mathbb{W} = \{u(\mathbf{p})\}_{\mathbf{p}\in\mathcal{D}}$, where a partition $\mathcal{P} = \{P_i\}_{i=1}^{n_c}$ of the image is a collection of pixel sets $\{P_i, i = 1, \ldots, n_c\}$ such that $P_i \bigcap P_j = \phi$ if $i \neq j$ and $\mathcal{D} = \bigcup_{i=1}^{n_c} P_i$. Each P_i is in fact a segment. n_c is the number of segments. Therefore, "*clusters/segments boundaries*" can be naturally formulated by the set of pixels $\mathcal{B} = \bigcup_{i=1,\ldots,n_c}\{\mathbf{p} \in P_i : (\mathbb{N}_1(\mathbf{p}) \setminus \mathbf{p}) \bigcap (\mathcal{D} \setminus P_i) \neq \emptyset\}$.

2.2 First Stage of VCells: Uniform Segmentation

The objective of the first stage of VCells is to generate small segments which are roughly the same shape and size. Our choice for stage one is Lloyd's algorithm [17], an effective tool for partitioning and grouping data points. For the VCells superpixel algorithm, we use *CfCVDT* [12], which is very efficient. *CfCVDT* is able to generate more than 10,000 small segments in 1 second and its runtime is independent of the image size. This means that *CfCVDT* requires roughly the same runtime for a very large image as it does for a small image.

In general, *CfCVDT* leads to small segments with a hexagonal shape, as shown in Fig. 4. As long as we can divide the image into segments with the same approximate shape and size, any method can be used. For example, we can simply divide the entire image into uniform squares or rectangles, as seen in Figs. 8 and 5.

Note that the first stage is independent of the image intensities and colors, and the segmentation results are reusable for images with the same number of superpixels. (For different image sizes, we can simply scale the image to obtain the desired results.) This feature is very useful in handling a large number of images.

In short, the first stage is actually an initialization process of VCells. The computed segments are used as the initial partition $\mathcal{P} = \{P_i\}_{i=1}^{n_c}$ for the next stage: EWCVT-LNN. Further discussion of the initialization is given in Section 3.1.

2.3 Second Stage of VCells: EWCVT-LNN

The EWCVT algorithms were carefully developed with the appropriately detailed definitions, theorems, proofs, and discussions [27]. EWCVT-LNN is a modified version of EWCVT. The key differences are the special mechanism Looking-Nearest-Neighbor and its companion Detecting-Segment-Breaking, which are used to ensure the resulting segments are spatially connected.

2.3.1 Algorithm for EWCVT-LNN

Given a partition $\mathcal{P} = \{P_i\}_{i=1}^{n_c}$ of image $\mathbb{W} = \{u(\mathbf{p})\}_{\mathbf{p}\in\mathcal{D}}$. The centroid of each P_i is defined as

$$\tilde{g}_i^e = \frac{1}{|P_i|}\sum_{\mathbf{p}\in P_i} u(\mathbf{p}), \tag{1}$$

where $|P_i|$ is the number of pixels belong to P_i. Since the centroids are calculated by using the colors of pixels, we highlight them as "*color centroids*."

Given a weight, λ, we can define the *Edge-Weighted Distance*.

Definition 1. *Suppose we have a digital image $\mathbb{W}$ and we are given a partition $\mathcal{P} = \{P_i\}_{i=1}^{n_c}$ and a set of colors $\mathcal{G} = \{g_i\}_{i=1}^{n_c}$. The edge-weighted distance from a pixel $\mathbf{p}$ to g_k is defined as*

$$\mathrm{dist}(\mathbf{p}, g_k) = \sqrt{|u(\mathbf{p}) - g_k|^2 + 2\lambda\tilde{n}_k(\mathbf{p})}, \tag{2}$$

where $\tilde{n}_k(\mathbf{p}) = |\mathbb{N}_\omega(\mathbf{p}) \setminus (P_k \cup \mathbf{p})|$, the number of pixels within $\mathbb{N}_\omega(\mathbf{p}) \setminus (P_k \cup \mathbf{p})$.

The EWCVT algorithm can be summarized as follows:

- Step 1: Partition the image into small segments.
- Step 2: Compute the color centroids of all segments.
- Step 3: For each pixel, evaluate the edge-weighted distance between the pixel and all color centroids and assign the pixel to the segment whose color centroid is nearest to it.
- Step 4: If there is no pixel going into a different segment in one iteration, return the current segments as the EWCVTs; otherwise, go to step 2 and repeat the iteration.

The EWCVT algorithm has been proven very successful for general image segmentation problems [27]. The segments obtained by EWCVT, however, may not be spatially connected. For some applications, this feature is useful and conceptually meaningful, as demonstrated in the European landmass in Fig. 2. The landmass in the image is not spatially connected, and although the islands are separated by the ocean, they should belong to the same cluster as

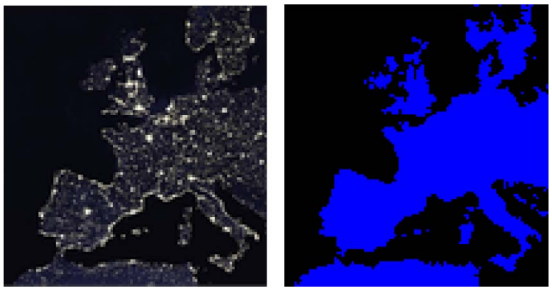

Fig. 2. Segmented the "Europe at night" image by EWCVT proposed in [27].

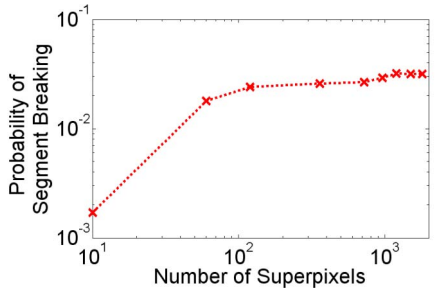

Fig. 3. Probability of segment breaking.

the mainland. For superpixel purposes, this *"good"* feature becomes unacceptable.

In Step 3 of the EWCVT algorithm, breakages may occur. According to the algorithm, if the edge-weighted distance $\mathrm{dist}(\mathbf{p}, g_k) \leq \mathrm{dist}(\mathbf{p}, g_i)$ for all $i = 1, \ldots, n_c$, $\mathbf{p}$ should be put into P_k whether $\mathbf{p}$ is physically connected to P_k or not. If $\mathbf{p}$ is an interior point of P_l in which $l \neq k$, then we will definitely have an unconnected segment.

A simple but effective strategy to significantly reduce the chance of segment breaking is LNN. By LNN, we actually mean that for each iteration in Step 3, we only evaluate $\mathrm{dist}(\mathbf{p}, g_k)$ in which $\mathbf{p}$ is a boundary pixel, i.e., $\mathbf{p} \in \mathcal{B}$; P_k contains at least one direct neighbor pixel of $\mathbf{p}$. Therefore, a pixel $\mathbf{p}$ can only be assigned to a segment which is physically connected to it and thus unconnected segments can be mostly avoided. We can see that the computational cost is significantly reduced by LNN since there is *no* need to evaluate the edge-weighted distance between a pixel and *all* color centroids.

Fig. 3 plots the probability of segment breaking versus the number of superpixels after we incorporate LNN to EWCVT. For a different preset number of superpixels, we apply EWCVT combined with LNN to the 300 images of the Berkeley data set [18] and record the number of broken segments. The probability of segment breaking is the averaged number of broken segments divided by the predefined number of superpixels. We can see the chance of segment breaking is very small, approximately 2 to 3 percent. In general, smaller superpixels are more likely to break because they are comparable to local regions which are highly irregular and inhomogeneous.

In order to guarantee the connectivity of the resulting segments, we can simply extract all of the "*connect components*" [10] and relabel them. This process is called *"Detecting-Segment-Breaking."* The key to DSB is the "*Flood Fill*" [2] algorithm, which can be written as a function: $n_{cc} = \mathrm{FloodFill}(\mathbf{p}, tc, rc)$. The inputs of FloodFill are pixel $\mathbf{p}$, the index of target cluster tc, and the replacement cluster rc. FloodFill searches all pixels which are connected to $\mathbf{p}$ by a path inside cluster tc and relabels them as a new cluster rc. The output n_{cc} is the size of the connected component containing $\mathbf{p}$. DSB is summarized by Algorithm 1.

Algorithm 1. DSB
Given: an image $\mathbb{W}$ and an arbitrary partition $\{P_i\}_{i=1}^{n_c}$;
mark all pixels $\mathbf{p}$ and segments P_i as unvisited;
for all pixel $\mathbf{p} \in \mathcal{D}$ **do**
  if $\mathbf{p}$ is unvisited **then**
    mark $\mathbf{p}$ as visited;
    say $\mathbf{p} \in P_k$;
    if P_k is unvisited **then**
      mark P_k as visited;
      $n_{cc} = \mathrm{FloodFill}(\mathbf{p}, k, k)$;
      if $n_{cc} \neq |P_k|$ **then**
        update color centroid g_k; {P_k is broken}
        end if
      else
        $n_c = n_c + 1$;
        $n_{cc} = \mathrm{FloodFill}(\mathbf{p}, k, n_c)$;
        compute color centroid g_{n_c};
      end if
  end if
end for
Return: $\{g_i\}_{i=1}^{n_c}$ and $\{P_i\}_{i=1}^{n_c}$

After we apply DSB, since the groups of pixels that have the same label are all connect components, the resulting segments are guaranteed to be spatially connected. Algorithm 2 is a full version of EWCVT-LNN incorporated with DSB. LNN is implemented in Steps 5 and 6 of Algorithm 2. Note that in Step 7, if $l \neq m$, we need to transfer $\mathbf{p}$ to a different cluster, which is equivalent to an update of P_l and P_m. We apply DSB at the last step of EWCVT-LNN to extract the connect component and do the relabeling, and thus the resulting superpixels are all spatially connected.

Algorithm 2. EWCVT-LNN
Given: an image $\mathbb{W}$ and an arbitrary partition $\{P_i\}_{i=1}^{n_c}$;
calculate the color centroids $\{g_i\}_{i=1}^{n_c}$ of $\{P_i\}_{i=1}^{n_c}$;
bPixelMoved = **true**;
while bPixelMoved **do**
  bPixelMoved = **false**
  for all pixel $\mathbf{p} \in \mathcal{B}$ **do**
    say $\mathbf{p} \in P_l$, evaluate $\mathrm{dist}(\mathbf{p}, g_k)$ defined in (2) for all $\{k : \mathbb{N}_\omega(\mathbf{p}) \bigcap P_k \neq \emptyset\}$;
    find $m = \arg\min_{k=1,\ldots,n_c} \mathrm{dist}(\mathbf{p}, g_k)$, say, from P_l to P_m;
    if $l \neq m$ **then**
      bPixelMoved = **true**;
      transfer $\mathbf{p}$ to P_m;
      replace g_l and g_m with the color centroids of the modified clusters P_l and P_m respectively;
    end if
  end for
end while
apply Algorithm 1 {DSB};
Return: $\{g_i\}_{i=1}^{n_c}$ and $\{P_i\}_{i=1}^{n_c}$

2.3.2 Another Perspective of Algorithm EWCVT

Algorithm 2 is actually a generalization of the classic k-means or Lloyd's algorithm [17]. As we pointed out in an earlier model, this algorithm is equivalent to "*minimize*" the *Edge-Weighted CVT Clustering Energy*, which is defined by

$$\begin{aligned}\widehat{E}(\mathcal{G}; \mathcal{P}) &= E^{CVT}(\mathcal{G}; \mathcal{P}) + E^{\mathcal{L}}(\mathcal{P}) \\ &= \sum_{l=1}^{L} \sum_{\mathbf{p} \in P_l} |u(\mathbf{p}) - g_l|^2 + \lambda \sum_{\mathbf{p} \in \mathcal{D}} n_{\mathcal{L}}(\mathbf{p}).\end{aligned} \tag{3}$$

$n_{\mathcal{L}}(\mathbf{p})$ is the number of neighbor pixels of $\mathbf{p}$ which do not belong to the same cluster of $\mathbf{p}$. The first term $E^{CVT}(\mathcal{G}; \mathcal{P})$ on the right-hand side of (3) is the classic CVT clustering energy. The second part $E^{\mathcal{L}}(\mathcal{P})$ is the so called "*edge energy*," which is proven to be an approximation of the cluster's edge length [27].

2.4 Summary of VCells Superpixel Algorithm and Its Complexity

Based on the former preparation, we can now give a clear description of VCells.

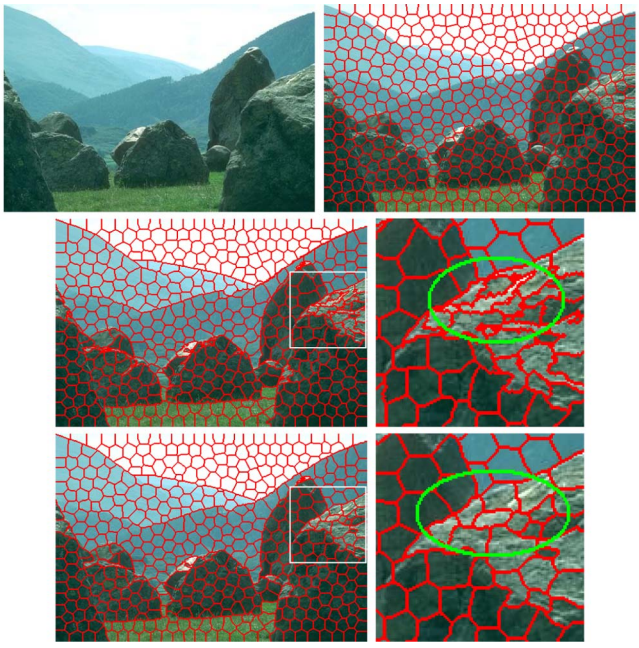

Fig. 4. First row: Original image and initial configuration by *CfCVDT*. Second and third rows: VCell results with $\lambda = 100, 300$, respectively, and the corresponding zoomed portions.

Algorithm 3. VCells

Given: an image $\mathbb{W}$ and an integer n_c;

1: apply *CfCVDT* to compute segments $\mathcal{P} = \{P_i\}_{i=1}^{n_c}$ uniformly in shape and size;

2: take $\mathbb{W}$, n_c and $\mathcal{P}$ as input, then apply EWCVT-LNN;

Return: $\{P_i\}_{i=1}^{n_c}$ updated by EWCVT-LNN as superpixels.

Because *CfCVDT* is independent of the image size, the computational cost in the first stage of VCells is negligible. In the second stage, because EWCVT only considers the boundary pixels, the computational cost in each iteration is $\mathcal{O}(n_B)$, in which n_B is the number of boundary pixels. Suppose the image contains N pixels and the number of superpixels is n_c. Each superpixel should contain approximately $\frac{N}{n_c}$ pixels and thus $\sqrt{\frac{N}{n_c}}$ boundary pixels. Therefore,

$$n_B \sim \mathcal{O}\left(n_c\sqrt{\frac{N}{n_c}}) = \mathcal{O}(\sqrt{n_c \cdot N}\right).$$

For K iterations, the total computational cost is $\mathcal{O}(K\sqrt{n_c \cdot N})$. In addition, the complexity of DSB is mainly due to the flood fill algorithm which is $\mathcal{O}(N)$. Excluding the cost of DSB and the initialization of EWCVT, the main computational cost of VCells algorithm is $\mathcal{O}(K\sqrt{n_c \cdot N})$.

3 Experimental Results and Discussions

In this section, we first discuss some important issues of VCells, including initialization and parameters's effects. Then the evaluation of the performance of VCells is presented together with other five algorithms: Turbopixels, Normalized Cuts, FH, Mean shift, and Watershed. The images used in this paper are all from the Berkeley database [18], which contains 300 (481×321 or 321×481) images. We use the entire data set to evaluate each algorithm. All free parameters were left at their default values for each of the six algorithms. The numerical experiments were performed on a laptop with an Intel Duo Core processor at 2.4 GHz.

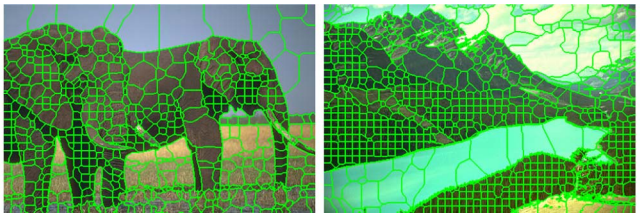

Fig. 5. Superpixel results obtained by VCells with Square Box as initializer.

3.1 Initial Clusters

As discussed in Section 2.2, we use *CfCVDT* to compute uniform hexagon segments for initialization. For most experiments in this paper, *CfCVDT* takes only several milliseconds. The first row of Fig. 4 shows the segmentation result by *CfCVDT*. As we mentioned in Section 2.2, the first stage of VCells is to compute a uniform segmentation of the image. Therefore, we can divide the image into small square boxes of approximately equal side length, size, and shape. Fig. 5 shows the results obtained by VCells using Square Box as initializer. Clearly, the shape deformation of superpixels is more severe than that of VCells with *CfCVDT*, see Fig. 8. Therefore, in the sense of deformation, VCells with *CfCVDT* results in more stable results.

3.2 Illustrations of Parameters' Effects

As we hypothesized, the mechanism in VCells, i.e., limiting the length of the boundaries, is in fact a compactness constraint. Thus, the weight factor λ of the edge energy can be used to control this constraint. Generally speaking, larger λ results in shorter and smoother boundaries and vice versa. The second and third rows of Fig. 4 show the superpixels obtained by VCells by using $\lambda = 100, 300$, respectively.

Consider the region highlighted by the green ellipse; clearly smaller λ results in more irregular superpixels with zigzag boundaries. However, compared with the original image, the superpixels fit the local image structure better. Let us take the "*spear*"-shaped superpixel inside the ellipse as an example. The spear's boundary is not smooth and its shape greatly differs from other superpixels. The expected brightness of the spear, however, is obviously much higher than its surroundings, which implies it is a good representation of the local image structure. For values of λ below 300, the spear disappear, as shown in the third row of Fig. 4, and the resulting superpixels are more uniform in size and shape. In the sense of the whole rock, the superpixels depict the boundaries with good accuracy.

Recall that the edge weighted CVT clustering energy consists of two terms: the classic CVT clustering energy and the edge energy. Therefore, we need to choose λ to balance these two energies [27], [29]. We can calculate the edge energy and classic CVT clustering energy as long as we have a partition. After the initial segmentation is computed, we choose λ such that

$$\frac{E^{\mathcal{L}}(\mathcal{P})}{E^{CVT}(\mathcal{G};\mathcal{P})} = M, \tag{4}$$

where M is a predetermined positive number between 1 and 10. We then fix λ and continue the iterations till convergence. Intuitively, M should be small for short initial boundaries since the boundaries may expand to fit the "*true*" boundaries and vice versa (notice that λ is proportional to M by (4)). As we computed in Section 2.4, the total boundary pixels are roughly $\mathcal{O}(\sqrt{n_c \cdot N})$ and thus the normalized boundary length is of

$$\frac{\mathcal{O}(\sqrt{n_c \cdot N})}{N} = \mathcal{O}\left(\sqrt{\frac{n_c}{N}}\right).$$

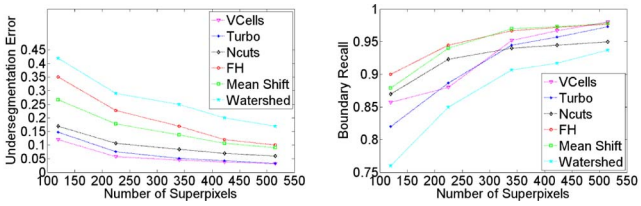

Fig. 6. Left: Undersegmentation error. Right: Boundary recall.

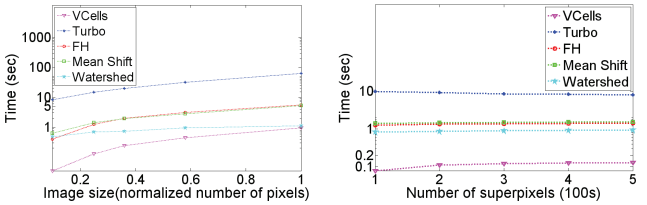

Fig. 7. Comparison of running time of VCells, TurboPixels, FH, Mean shift, and Watershed.

For simplicity, we can set $M = \max(\min(25\sqrt{\frac{n_c}{N}}, 10), 1)$. Because of this formula, M is predetermined and is always between 1 and 10. Unless stated otherwise, λ is calculated automatically by using the above process in all of our examples and experiments.

As we discussed in [27], the effect of ω is in fact very similar to λ. The computational cost, however, increases in a square manner with ω. Therefore, we can choose a relatively small value of ω and then hold it constant for all experiments. We set $\omega = 3$ for all examples and experiments in this paper.

3.3 Undersegmentation Error and Boundary Recall

The undersegmentation error is calculated by using the formula defined in [15], i.e.,

$$\frac{\left[\sum_{\{s_j | s_j \bigcap g_i \neq \phi\}} Area(s_j)\right] - Area(g_i)}{Area(g_i)}, \tag{5}$$

in which $\{g_i\}_{i=1}^{K}$ represents the segmentation of the ground truth image and $\{s_j\}_{j=1}^{L}$ denotes the superpixels produced by the algorithms. Equation (5) refers to a single ground truth image segment. The undersegmentation error of an image is calculated by averaging this quantity over all ground truth segments.

The boundary recall is calculated by using the standard measure, i.e., the fraction of the ground truth boundaries which fall within a small disk shaped neighborhood of the superpixels' boundaries in which the radius is set to 2 pixels.

Fig. 6 represents the undersegmentation error and boundary recall of VCells, Turbopixels, N-cuts, FH, Meanshift, and Watershed, all of which are obtained by averaging the 300 images from the Berkeley data set.

The plot of the undersegmentation error shows the overall performance of the algorithms that incorporate compact constraints is clearly much better than those that do not. The reason is simply because lack of compact constraints makes it impossible to prevent FH, Mean Shift, and Watershed from growing highly irregular boundaries. Accordingly, superpixels generated by FH, Mean Shift, and Watershed vary greatly in size and shape.

However, lack of a compact constraint would enable FH and Mean Shift to better capture the boundaries of the irregular regions at lower superpixel densities [15]. As shown by the plot of boundary recall, FH and Mean Shift offer the best overall recall of the algorithms for low superpixel densities. Although Watershed does not encode a compact constraint, it is very sensitive to variations in local intensities. To extract useful features, intense prior image enhancement is needed. Therefore, for a broad range of images from real scenes, Watershed varies too much and, on average, causes poor behavior. As seen in Fig. 6 (right), the performance of VCells is better than TurboPixels for low superpixel density among the compact constraint encoded algorithms. For high superpixel density, VCells offers the highest boundary recall.

3.4 Timing Evaluation

In this section, we compare the runtime of VCells, Turbopixels, FH, Meanshift, and Watershed. We disregard N-cuts since it is computationally expensive and memory consuming compared to the other five algorithms.

The left image of Fig. 7 shows the plot of the running time as a function of image size. We roughly set the expected size of superpixels as 10×10. Compared to the other four algorithms, VCells is even faster than those which do not encode compact constraint. Despite the low computational complexity of VCells, as discussed in Section 2.4, extensive numerical experiments indicate that only a small portion of superpixels near the boundaries deform to fit the local image structure during each iteration. A larger amount of superpixels which are far away from the local image boundaries will remain roughly the same. Therefore, it will save us a lot of time to reevaluate the color centroids of the deformed segments.

The right image of Fig. 7 shows the plot runtime versus superpixel density. The image size is kept at 240×160. Since the runtime of FH, Meanshift, and Watershed mainly depend upon the image size, the plots for these three algorithms are roughly constant. Similarly, since the image size is fixed, the running time of VCells increases roughly as $\mathcal{O}(\sqrt{n_c})$, as shown in Fig. 7b.

3.5 More Experimental Results of VCells

We provide more examples in Fig. 8 to give a qualitative feel for superpixels by VCells. The images in the Berkeley data set are from a broad range of real scenes, which is very useful in evaluating VCells thoroughly. In all of the examples shown in Fig. 8, the expected size of the superpixels is kept as 10×10. Therefore, for the images from Berkeley benchmark, the number of superpixels is roughly 48×32. Clearly most of the superpixels are almost uniform hexagon cells. As we explained in Section 3.4, severe shape deformation only happens among the superpixels near the boundaries. For most of the superpixels which are far away from

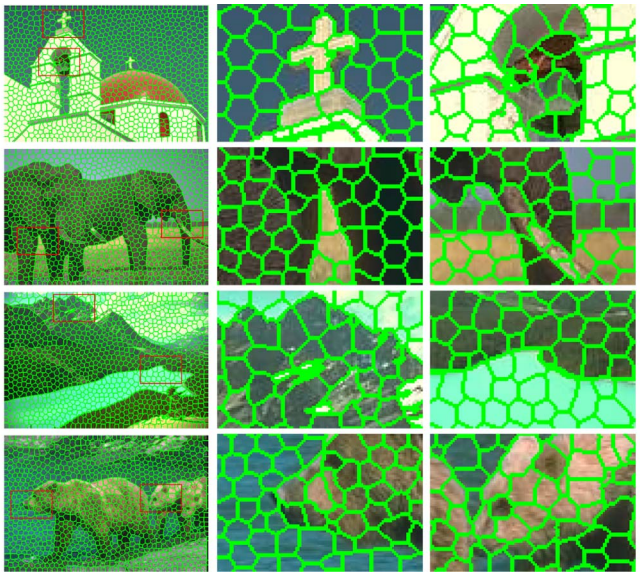

Fig. 8. Superpixel results obtained by applying the VCells algorithm on various real scene images.

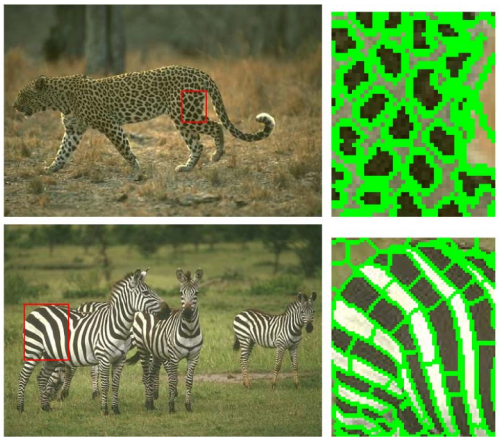

Fig. 9. VCells for texture images.

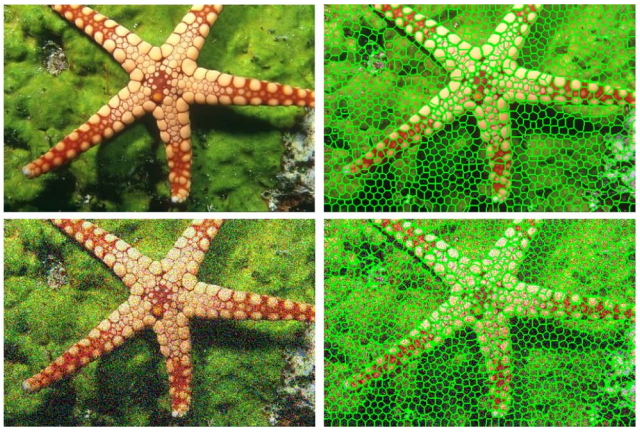

Fig. 10. Demonstration of the robustness of VCells. The second row is the noised version of the first row.

the boundaries, they roughly remain the same. However, unlike Turbopixels (Fig. 7 in [15]), when we apply VCells to texture images, the superpixels inside the texture region deform to fit the textures. Fig. 9 provides two examples by applying VCells to images with texture. In the first example, superpixels inside the cougar coincide with the spots, instead of hexagons as shown in Fig. 8. In the second example, superpixels covering the zebra deform into quadrilaterals and thus nicely depict the striped pattern. We expect self-adaptivity of superpixels to texture would be helpful for postprocessing images, like merging, etc.

Fig. 10 shows the superpixels by VCells of image with noise. The first picture of the second row is obtained by adding Gaussian noise with 0 mean and 0.1 variance to the original picture. The two pictures in the right column are the superpixels by VCells of the corresponding left pictures. Clearly, the superpixels results are very close to each other, which implies that VCells is very robust with noise.

4 Concluding Remarks

In conclusion, we offer the following comments regarding the superpixels by VCells according to the five basic principles proposed in [15]:

1. **Uniform size and coverage**. With hexagons by *CfCVDT* as initializer, only the superpixels near objects' boundaries deform sharply while the others remain roughly the same during each iteration. In the homogenous region, most resulting superpixels remain roughly like hexagons.
2. **Connectivity**. LNN encourages connectedness while DSB ensures the safety of VCells.
3. **Compactness**. The compactness constraint is encoded by introducing the edge energy in VCells.
4. **Smooth, edge-preserving flow**. Our algorithm is not a geometric-flow based formulation, so difficulties that occur in the edge evolution process (such as boundary crossing and collision) are nonexistent.
5. **No superpixel overlap**. This is automatically ensured in VCells algorithm.

In the VCells algorithm, EWCVT-LNN inherits the simplicity of the k-means algorithm and encodes the compact constraint in a controllable manner by introducing edge energy. We expect VCells to be a standard tool for preprocessing the images. We believe superpixels, together with some *"merging"* schemes, could be a powerful tool to analyze inhomogeneous images which appear so frequently in the real world [16]. One of our future research projects is to apply VCells to the field of medical image processing.

Acknowledgments

The authors would like to thank Alex Levinshtein and Kiriakos N. Kutulakos for contributing the images, data, and TurboPixels [15] used in the comparisons. The authors extend their sincere gratitude to Lili Ju for his valuable discussions and suggestions and for making the CfCVDT package available. This research is supported in part by US National Science Foundation Grant NSF-DMS 0913491.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels," Technical Report 149300, EPFL, June 2010.
[2] W. Burger and M. Burge, *Principles of Digital Image Processing*. Springer, 2009.
[3] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, May 2002.
[4] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power Watersheds: A New Image Segmentation Framework Extending Graph Cuts, Random Walker and Optimal Spanning Forest," *Proc. 12th IEEE Int'l Conf. Computer Vision*, pp. 731-738, 2009.
[5] T. Cour, F. Benezit, and J. Shi, "Spectral Segmentation with Multiscale Graph Decomposition," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, pp. 1124-1131, 2005.
[6] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Rev.*, vol. 41, pp. 637-676, 1999.
[7] Q. Du, M. Gunzburger, and L. Ju, "Constrained Centroidal Voronoi Tessellations on General Surfaces," *SIAM J. Scientific Computing*, vol. 24, pp. 1488-1506, 2003.
[8] Q. Du, M. Gunzburger, L. Ju, and X. Wang, "Voronoi Tessellation Algorithms for Image Compression and Segmentation," *J. Math. Imaging and Vision*, vol. 24, pp. 177-194, 2006.
[9] P. Felzenszwalb and D. Huttenlocher, "Efficient Graph-Based Image Segmentation," *Int'l J. Computer Vision*, vol. 59, pp. 167-181, 2004.
[10] R. Gonzalez and R. Woods, *Digital Image Processing*. Prentice Hall, 2007.
[11] A. Hausner, "Simulating Decorative Mosaics," *Proc. 28th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 573-580, 2001.
[12] L. Ju, "Conforming Centroidal Voronoi Delaunay Triangulation For Quality Mesh Generation," *Int'l J. Numerical Analysis and Modeling*, vol. 4, pp. 531-547, 2007.
[13] L. Ju, Q. Du, and M. Gunzburger, "Probabilistic Methods for Centroidal Voronoi Tessellations and Their Parallel Implementations," *Parallel Computing*, vol. 28, pp. 1477-1500, 2002.
[14] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881-892, July 2002.
[15] A. Levinshtein, A. Stere, K.N. Kutulakos, D.J. Fleet, S.J. Dickinson, and K. Siddiqi, "TurboPixels: Fast Superpixels Using Geometric Flows," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2290-2297, Dec. 2009.
[16] C. Li, C. Kao, J. Gore, and Z. Ding, "Minimization of Region-Scalable Fitting Energy for Image Segmentation," *IEEE Trans. Image Processing*, vol. 17, no. 10, pp. 1940-1949, Oct. 2008.
[17] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Information Theory*, vol. 28, no. 2, pp. 129-137, Mar. 1982.

[18] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A Database of Human Segmented Natural Images and Its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 416-423, 2001.
[19] A. Moore, S. Prince, and J. Warrel, "Lattice Cut—Constructing Superpixels Using Layer Constraints," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2117-2124, 2010.
[20] X. Ren and J. Malik, "Learning a Classification Model for Segmentation," *Proc. IEEE Ninth Int'l Conf. Computer Vision,* vol. 1, pp. 10-17, 2003.
[21] E. Sharon, A. Brandt, and R. Basri, "Fast Multiscale Image Segmentation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 70-77, 2000.
[22] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 731-737, 1997.
[23] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.
[24] A. Vedaldi and S. Soatto, "Quick Shift and Kernel Methods for Mode Seeking," *Proc. European Conf. Computer Vision,* pp. 705-718, 2008.
[25] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and Supervoxels in an Energy Optimization Framework," *Proc. European Conf. Computer Vision,* pp. 211-224, 2010.
[26] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 6, pp. 583-598, June 1991.
[27] J. Wang, L. Ju, and X. Wang, "An Edge-Weighted Centroidal Voronoi Tessellation Model For Image Segmentation," *IEEE Trans. Image Processing,* vol. 18, no. 8, pp. 1844-1858, Aug. 2009.
[28] J. Wang, L. Ju, and X. Wang, "Edge-Weighted Centroidal Voronoi Tessellations," *Numerical Math.: Theory, Methods and Applications,* vol. 3, pp. 223-244, 2010.
[29] J. Wang, L. Ju, and X. Wang, "Image Segmentation Using Local Variation and Edge-Weighted Centroidal Voronoi Tessellations," *IEEE Trans. Image Processing,* vol. 20, no. 11, pp. 3242-3256, Nov. 2011.
[30] S. Yu and J. Shi, "Multiclass Spectral Clustering," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 313-319, 2003.