Eikonal-Based region growing for efficient clustering

Pierre Buyssens, Isabelle Gardin, Su Ruan, Abderrahim Elmoataz

Eikonal-based Region Growing for Efficient Clustering

Pierre Buyssens[a], Isabelle Gardin[a,b], Su Ruan[a], Abderrahim Elmoataz[c]

[a]*LITIS EA 4108 - QuantIF, Université de Rouen,*
22 boulevard Gambetta, 76183 Rouen Cedex, France
[b]*Département de médecine nucléaire, centre Henri-Becquerel,*
1 rue d'Amiens, 76038 Rouen, France
[c]*GREYC (UMR 6072) - CNRS, Université de Caen Basse-Normandie,*
ENSICAEN - Image Team. 6, Bd. Maréchal Juin - 14000 Caen, FRANCE

Abstract

We describe an Eikonal-based algorithm for computing dense oversegmentation of an image, often called *superpixels*. This oversegmentation respect local image boundaries while limiting undersegmentation. The proposed algorithm relies on a region growing scheme, where the potential map used is not fixed and evolves during the diffusion. Refinement steps are also proposed to enhance at low cost the first oversegmentation. Quantitative comparisons on the Berkeley dataset show good performance on traditional metrics over current state-of-the art superpixel methods.

Keywords: Superpixels, segmentation, clustering, Eikonal equation

1. Introduction

With the increasing amount of available data, and the need for fast and accurate processing, the simplification of data becomes a crucial point for many applications. A convenient way to address this task is to consider that datas can be modeled with a graph $G = (V, E, w)$, where V is the set of vertices, E a set of edges, and $w > 0$ is the weight function that

models the interaction between vertices. Exhibiting clusters of this graph leads to a simplification of the data and decreases the size of the problem. Many techniques of graph clustering have been proposed such as cut-based, spectral or random walk methods (see [16] for a comprehensive review of these techniques).

Recent works [3] adapt the eikonal equation to graphs in order to perform over-clustering from an initial set of annotated vertices V_0. Let $f : V \to \mathbb{R}$ be a real-valued function that assigns a real value $f(u)$ to each vertex $u \in V$. The reformulation of the Eikonal equation in the graph domain leads to the equation :

$$\begin{cases} \|(\nabla_w^f)(u)\|_p = P(u) & \forall u \in V \\ f(u) = \phi(u) & \forall u \in V_0 \end{cases} \tag{1}$$

where $(\nabla_w^- f)(u)$ is the weighted morphological gradient at a vertex u (see [11] for details), P is a positive function, and ϕ is an initialization function.

In this paper, we focus on grid graphs for image processing with the aim of grouping perceptually and adjacent pixels into meaningful regions, the so-called *superpixels*. Superpixels have become an important step in many computer vision applications such as segmentation [8, 22], object localization [7], depth estimation [24], and scene labeling [5].

Some properties of an algorithm that generate superpixels are often desirable : (1) Superpixels should adhere well to object boundaries while limiting undersegmentation errors, (2) as superpixel methods are used as preprocessing, the algorithm should have a low complexity, (3) it has to be simple to use (i.e. few parameters). In addition, some other properties may be desired

: the control of the amount of superpixels, or the compactness of them.

Several superpixel algorithms exist in the literature, they can be roughly divided into two approaches : The first consists in gradually growing superpixels from an initial set of centers. This approach includes Watershed [21], Turbopixels [10], SLIC [1], Consistent Segmentation [25] and Quick Shift [19]. The second approach relies on a graph formulation of the problem and aims at finding an optimal cut according to an objective function that takes similarities of neighboring pixels into account. This approach includes Entropy–based energy function method [12], optimal cuts [14, 13], graph-cut [20], and agglomerative clustering of the nodes of the graph [6].

In this paper, we propose a new algorithm for superpixel generation : *Eikonal-based Region Growing Clustering*[1] (ERGC) that starts from an initial set of seeds and dilates them, and then refines the result oversegmentation by adding/moving cuts. It formulates the superpixel segmentation task as a solution of an Eikonal equation. Equation 1 becomes :

$$\begin{cases} \|\nabla U(x)\| = F(x) & \forall x \in \mathcal{I} \\ U(x) = 0 & \forall x \in \Gamma \end{cases} \tag{2}$$

where $\mathcal{I}$ is the image domain, F a positive function, Γ the set of initial seeds, and $U(x)$ the traveling time or geodesic distance of x from source Γ. Focusing on grid graphs, it can be solved efficiently with the *Fast-Marching* method. The major change proposed in this paper concerns the function F, which is not fixed and evolves during the front evolution. It is detailed at Section 2.2.

[1]Source code and executable of ERGC can be found at
`https://sites.google.com/site/pierrebuyssens/ergc`

ERGC is simple to use (by default, the only parameter is the desired number of superpixels), as fast as other superpixels methods, and outperforms them on two of the three traditional metrics.

The rest of the paper is organized as follows : Section 2 details the proposed potential function F, and the ERGC algorithm. Section 3 gives qualitative and quantitative comparisons of performances between ERGC and state-of-the-art methods. Some aspects and extensions of the proposed method are than discussed in Section 4, while Section 5 concludes the paper.

2. Superpixels method

2.1. Notations

In the following we adopt several notations to simplify the reading of the paper. A particular pixel of image $\mathcal{I}$ is noted $\boldsymbol{p}$ and consists of a coordinate couple $(x_{\boldsymbol{p}}, y_{\boldsymbol{p}})$. A region $\boldsymbol{R}_i$ consists of a seed pixel $\boldsymbol{s}_i$ and a size N_i in pixels. The color of a pixel $\boldsymbol{p}$ is noted $\boldsymbol{C}_{\boldsymbol{p}}$, and the mean color of a region R_i is noted $\boldsymbol{C}_i$.

Note that the color images are considered in the CIELAB colorspace, so the color vector of a pixel (or a region) $\boldsymbol{C}$ reduces to $[l, a, b]^T$.

2.2. Proposed potential function

Since a superpixels method aims at grouping perceptually and adjacent pixels into meaningful regions, we propose a potential function F that conveys this desirable property. The right term of equation 2 is computed according to the mean color of the adjoining region :

$$F_c(\boldsymbol{p}, R_i) = \|\boldsymbol{C}_{\boldsymbol{p}} - \boldsymbol{C}_i\|_2^2 \tag{3}$$

This potential function measures the perceptual color distance between the pixel $\boldsymbol{p}$ and the region R_i. For color images in the CIELAB colorspace, F reduces to :

$$F_c(\boldsymbol{p}, R_i) = (l_{\boldsymbol{p}} - l_i)^2 + (a_{\boldsymbol{p}} - a_i)^2 + (b_{\boldsymbol{p}} - b_i)^2 \tag{4}$$

where $[l_i, a_i, b_i]^T$ is the mean color vector of region R_i.

In comparison to traditional gradient–based approaches [3] where $F(\boldsymbol{p}) = \|\nabla \mathcal{I}\|$, the proposed formulation favors the grouping of similar pixels, even for pixels that are far from the initial seeds (Fig. 2).

As a numerical solver of the Eikonal equation 2, we adopt the Fast Marching method introduced by Sethian in [17]. It uses a priority queue to order the pixels as being the current estimate of the geodesic distance to the closest seed (see [15] for a detailed description of the fast marching algorithm).

Within the fast marching algorithm, each time a pixel $\boldsymbol{p}$ is inserted to a region R_i, the attributes of this region are easily updated :

$$\begin{cases} \boldsymbol{C}_i & \leftarrow & \frac{\boldsymbol{C}_i \times N_i + \boldsymbol{C}_{\boldsymbol{p}}}{N_i + 1} \\ N_i & \leftarrow & N_i + 1 \end{cases} \tag{5}$$

This formulation clearly makes the potential of a pixel dependant on the features of an adjoining region, which is updated during the process. It promotes the diffusion to pixels whose color is close to the color of the region, hence creating homogeneous superpixels. A region can also absorb smoothly localized noisy pixels since such a pixel contributes weakly to the mean color of the region.

Algorithm 1 summarized the *Fast-Marching* algorithm with our proposed potential function. Within the algorithm, a state Σ is given to each pixel and

changes during the processing : *COMPUTED* states that the solution for a pixel has been computed (i.e. its solution will not change anymore), *ALIVE* states that the solution of a pixel is being computed, and *FAR AWAY* states that a pixel has not yet been visited. The algorithm involves a heap structure of *ALIVE* points, noted L, and each time a pixel p with coordinates (x, y) is visited, its local solution is computed w.r.t. its neighbors $Neigh(p)$:

$$U(p) \;=\; \begin{cases} \frac{1}{2}(U(q) + U(r) + \sqrt{\Delta}) & if \qquad \Delta \geq 0 \\ \min(U(q), U(r)) + F & otherwise \end{cases} \tag{6}$$

where F is computed with equation 3, q and r are the neighbors of p such that $U(q) \;=\; \min(U(x-1,y), U(x+1,y))$, $U(r) \;=\; \min(U(x,y-1), U(x,y+1))$, and Δ corresponds to solving the equation $(u - U(r))^2 + (u - U(q))^2 = F^2$ (see [15] for details).

Figure 1 compares the geodesic distance map computed on a synthetic image (left) with the gradient-based potential function (middle) and the proposed one (right), with an initial seed depicted by the white dot. This example exhibits the main feature of the proposed potential function. With the gradient-based potential function (middle), the front propagates on the white square before having recovered all the black area. Some pixels belonging to the square then have a lower geodesic distance than pixels belonging to the black area. A good segmentation of the square based on these distances is then impossible. With the proposed potential function (right), the front propagates first entirely on the black area before entering onto the white square. A good segmentation of the square can then easily be achieved.

Such a result can be interpreted in the following way : For the gradient-based potential function, when a front arrives on a contour, its speed heavily

Algorithm 1 Fast-Marching algorithm with the proposed potential function

for all $p \in \mathcal{I}$ **do**
  if p is a seed **then**
    $\Sigma_p \leftarrow$ ALIVE, $U(p) = 0$, add p to L
  else
    $\Sigma_p \leftarrow$ FAR AWAY, $U(p) = \infty$
  end if
end for
while $L \neq \varnothing$ **do**
  $p \leftarrow \arg\min_q(U(q) \mid q \in L)$
  $\Sigma_p \leftarrow$ COMPUTED
  Update adjoining superpixel R_i (equation 5)
  for $q \in Neigh(p)$ **do**
    compute $F(q, R_i)$ (equation 3)
    compute local solution $t(q)$ (equation 6)
    if $\Sigma_q =$ ALIVE **and** $t(q) < U(q)$ **then**
      $U(q) = t(q)$
    end if
    if $\Sigma_q =$ FAR AWAY **then**
      $\Sigma_q \leftarrow$ ALIVE
      $U(q) = t(q)$
      Add q to L
    end if
  end for
end while

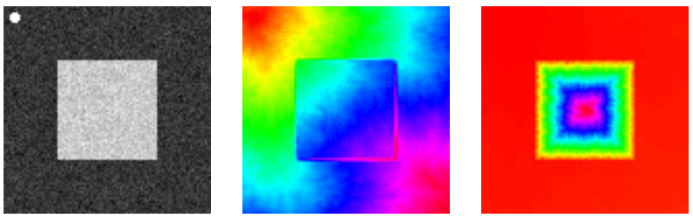


Figure 1: Left : Initial image with the given seed depicted as the white dot. Middle : geodesic distance map U in false color obtained with the gradient-based potential function. Right : geodesic distance map obtained with the proposed potential function. A segmentation of the square based on the geodesic distances can not be obtained with the gradient-based potential function.

decreases as the potential is high. Nevertheless, after a given time, the front ends up passing through the contour, and the potential becomes low again, so the front can evolve with a high speed. This behavior is not a surprise and can be seen on the *red-blue* image of Figure 2. The proposed potential function adds to the diffusion a sort of memory of the initial color of the seeds. When a front arrives on a region with a different color, the potential becomes very high. Even if the front passes through the interface separating the two regions, the potential remains very high, hence prohibiting the diffusion too much.

Figure 2 compares the behavior of the proposed potential function and the gradient–based approach [3] on a color synthetic image with two seeds, and on a natural image with three seeds. In both cases, the proposed formulation gives a better segmentation.

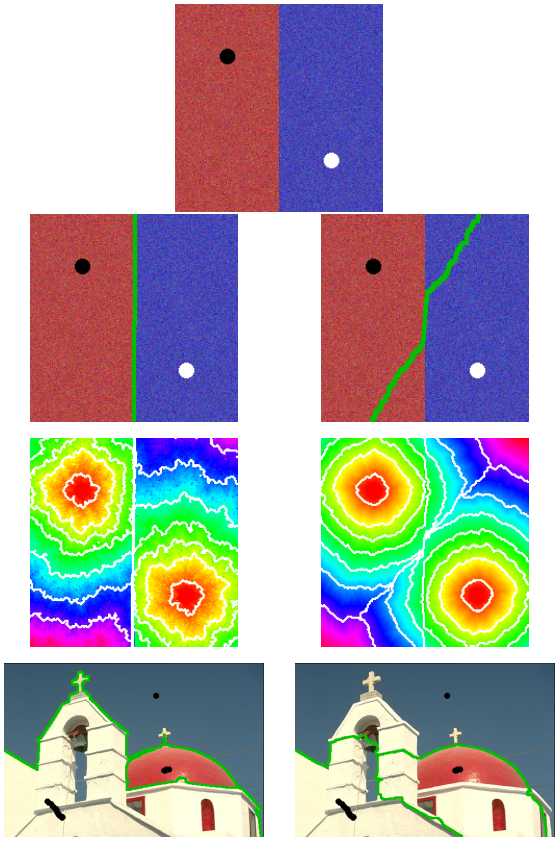


Figure 2: First row : initial image. The back and white circles depict the initial seeds. Second row : the result segmentation with the proposed F (left) and the gradient–based potential function (right). Third row : geodesic distances map U in fake color. Some isocontours are shown in white. Last row : segmentation of a natural scene. Seeds are depicted in black.

2.3. Algorithm initialization

The initialization consists of sampling K seed pixels on a regular grid with an interval S equal to $S = \sqrt{N/K}$ with N the number of pixels in the image (initialization similar to [1] and [10]).

The mean and variance color of the seed pixel and its 4-connexity neighboring pixels is then computed. The same computation is performed for pixels that lie in a 3×3 neighborhood, and the seed is moved to the pixel that lowers the variance color. Such a perturbation of the initial seeds avoids potential outlier pixels as seeds, favorably initializes the diffusion, and gives a more robust initial mean color for each superpixel.

2.4. Complexity and segmentation speed

After the initialization, a front propagation is performed with an online update of the superpixels. Complexity of the diffusion relies essentially on the complexity of the fast marching algorithm, which is roughly in $\mathcal{O}(n \log(n))$ with an appropriate heap for sorting the pixels according to their geodesic distance. Despite this theoretical complexity, the proposed algorithm is very fast in practice, and is nearly linear in time. Note that using different data structures (and additional storage), different $\mathcal{O}(n)$ implementations have been proposed in [23, 9].

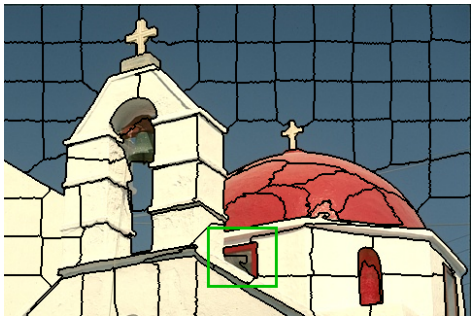
2.5. Refinement by adding new superpixels

In this section, we propose a refinement of the clustering that iteratively adds new seeds after a full pass of the algorithm. Since initial seeds are placed on a grid, some objects of the image may not contain an appropriate seed, and these objects may not be finally well segmented (Fig. 3(c)). In such a

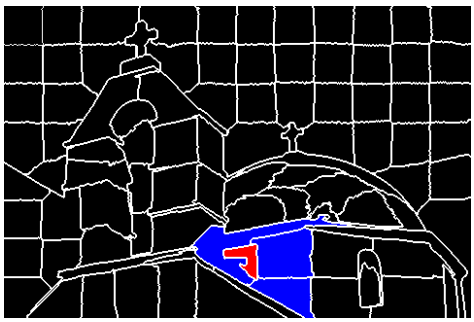
case the resulting geodesic distance map U exhibits high values in this area (Fig. 3(d)). The refinement consists of adding a new seed to the location of the maximum geodesic distance, and to recompute the solution of equation 2. Since it is unnecessary to apply the algorithm to the whole image, only a small part of the image around the new seed is considered (Fig. 3(b)). The refinement is summarized as follows :

1. Perform a full pass of ERGC with the seeds placed on a grid,
2. add a new seed to the location of the maximum geodesic distance found at the previous step,
3. let R_i be the superpixel in which there is the new seed (red superpixel of Fig. 3(b)), perform ERGC for pixels belonging to R_i and its adjacent superpixels (blue superpixels of Fig. 3(b). At this step, the seeds of the refined superpixels are left unchanged,
4. iterate steps 2 and 3 until the number of desired new seeds is reached.

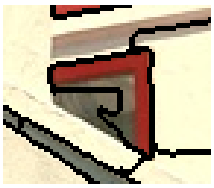
A resulting refinement iteration, and the associated geodesic distance map are shown at Figures 3(e) and 3(f). The cost of a refinement iteration depends on the number of pixels considered. Nevertheless, this refinement is considerably less costly than a full ERGC pass, since it only deals with a small part of the image. Given superpixels of size $S = \sqrt{N/K}$, and let be b the number of adjacent superpixels of R_i, the complexity of one refinement iteration is roughly $\mathcal{O}\left(bS\log(bS)\right)$, which can be approximated with $\mathcal{O}\left(\sqrt{\frac{N}{K}}\log(\frac{N}{K})\right)$. For information, one refinement iteration costs approximatively 3 ms for 500 initial seeds, and 8 ms for 100 initial seeds on a Berkeley image. These time calculations have been obtained with a standard laptop equipped with an Intel mono core 1.30GHz processor and 4 GB RAM.



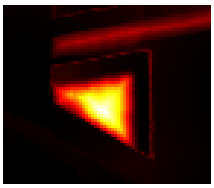
(a)



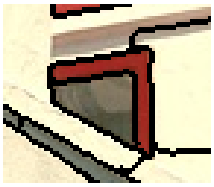
(b)



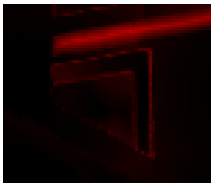
(c)



(d)



(e)



(f)

Figure 3: (a) ERGC segmentation with 100 superpixels (approximatively). (b) Part of the image to be refined (red and blue superpixels). (c) Detail of the image (green box of (a)) and corresponding geodesic distance map (d). Adding a new seed to the location of the maximum distance results in a better segmentation (e). (f) New geodesic distance map coresonding to (e).

2.6. Refinement by moving superpixels

In this section, we propose a simple procedure to increase the quality of the oversegmentation without increasing the total number of superpixels. It consists in adding a new superpixel while removing a "weak" one, such that the global number of superpixels remains constant.

The selection of the superpixel to remove consists of 3 steps :

1. Given an oversegmentation of the image, compute the underlying Region Adjacency Graph (RAG),
2. For each vertex v of the graph, compute its normalized volume vol:

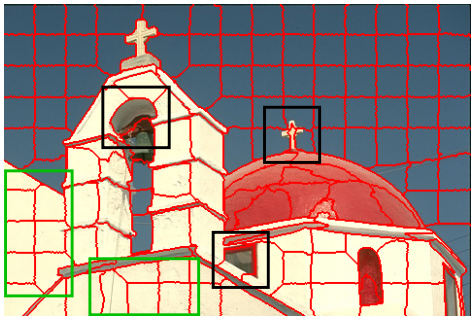
$$vol(v) = \frac{\sum_{u \sim v} w(u,v)}{Card(\mathcal{N}_v)} \tag{7}$$

   where $u \sim v$ denotes two adjacent vertices, $w(u,v)$ the weight of the edge connecting u and v, and $\mathcal{N}_v$ the set of neighbors of v.
3. Select the superpixel corresponding to the vertex with the minimal normalized volume as the superpixel to remove.

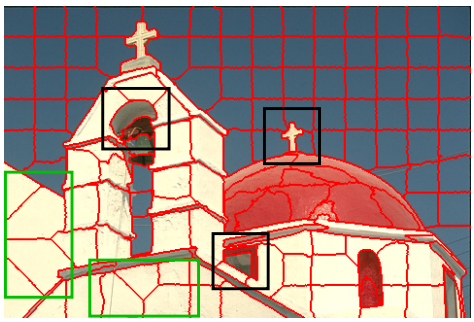
This procedure selects the superpixel that is the closest to its neighbors. The weight function is the $\mathcal{L}_2$ norm and reflects the difference between two adjacent superpixels.

The initial oversegmentation is then refined by adding a new superpixel with the procedure 2.5 detailed above, while the superpixel to remove is simply discarded.

The whole procedure is iterated until a criterion is reached. In the following, it is stopped when $\int_I U$ no longer decreases. We also add a limit of 10 iterations, which in practice, is rarely reached.



(a)



(b)

Figure 4: Example of refinement by moving superpixels. 3 superpixels have been removed (from green boxes), and added at other locations (black boxes).

Figure 4 shows a refinement of an initial oversegmentation by moving 3 superpixels. Both oversegmentations contain 150 superpixels. The complexity of this refinement is low since it is similar to the refinement by adding new seeds.

2.7. Summary of the algorithm

Although a spatial constraint can easily be added to F (see figure 3), by default the only parameter of ERGC is the number of desired superpixels. The whole algorithm consists of 3 steps :

1. Proceed to the initial diffusion with 90% of the desired seeds placed on a grid,
2. Refine the oversegmentation by adding the remaining 10% of seeds (Section 2.5),
3. Iterate the moving seeds procedure (Section 2.6) until the stopping criterion is reached.

For example, the algorithm produces 200 superpixels by first placing 180 seeds on the initial grid, then by adding 20 more seeds with the procedure described in Section 2.5.

These ratios ensure that the whole algorithm runs in a reasonable amount of time (about half a second on a Berkeley image).

3. Comparison with State–of–the–Art

We compare ERGC to state–of–the–art methods SLIC[2] [1], Entropy Rate Superpixels[3] [12] (ERS), SEEDS[4] [18], TurboPixels[5] [10] (TP), and Gradient–based diffusion [3] (GrB). Examples of superpixel segmentations produced by each method appear in Figure 7.

The popular SLIC method proposed in [1] adapts k–means in the image plane to iteratively exhibit superpixels. The addition of a spatial constraint term produces regular regions that adhere quite well to image. The ERS algorithm proposed in [12] oversegments an image via an obective function

[2] http://ivrg.epfl.ch/supplementary_material/RK_SLICSuperpixels

[3] http://www.umiacs.umd.edu/~mingyliu/research

[4] http://www.vision.ee.ethz.ch/software

[5] http://www.cs.toronto.edu/~babalex/research.html

composed of two terms : entropy rate of a random walk on a graph and a balancing term. By default, it produces irregular superpixels that adapt to local image structure, but is quite slow in practice. Starting from an initial superpixel partitioning (a grid), SEEDS [18] continuously refines the superpixels by modifying the boundaries. Based on a simple hill-climbing optimization, it minimizes an energy function based on enforcing color similarity between the boundaries and the superpixel color histogram. This fast algorithm produces the most irregular superpixels of the literature. The TurboPixels algorithm [10] consists in iterating three steps : (1) evolve the boundaries of the superpixels for a given number of time steps, (2) compute the squeletton of these boundaries, and (3) update velocities of the boundaries. TP produces regular superpixels that often fail to adapt to local image structure, especially when the desired number of superpixels is low. Moreover, it is the slowest algorithm among the top performers. Finally, the gradient-based diffusion method GrB proposed in [3] essentially solves the Eikonal equation with a gradient-based potential function. It suffers from leaks inherent of this potential function as outlined in Section 2.2 (examples shown in Figures 1 and 2).

The Berkeley dataset [2] is used as a benchmark. It consists of 500 images of size 481×321 (or 321×481) and several ground truth manual segmentations for each image.

Figure 6 shows quantitative results on standard metrics, including *Boundary Recall*, *Undersegmentation Error* and *Achievable Segmentation Accuracy*. We also add a compactness metric that reflects the compacity of the superpixels. As a baseline for *Boundary Recall*, *Undersegmentation Error* and

Achievable Segmentation Accuracy metrics, we also show the performances of a grid of square superpixels (GRID).

All results are computed from scratch using these evaluation metrics and the same hardware. Default parameters are used for the state–of–the–art methods.

3.1. Boundary recall

Boundary recall (BR) measures the fraction of ground truth edges that is also present in superpixel segmentation within a distance threshold t. In our experiments, t is fixed to 2 as in [1, 12, 18]. Fig. 6(a) shows boundary recall measures for each method according to the number of superpixels. As the number of superpixels increases, the boundary recall is naturally higher. SEEDS outperforms all other algorithms on this metric. Nevertheless, this result has to be appreciated in the light of the superpixels compactness. As shown at Fig. 6(d) (second and third rows), ERS and SEEDS superpixels have the lowests mean compactness values. For a fixed number of superpixels, there are then much more ERS or SEEDS superpixels boundaries in the segmentation, which naturally increases the boundary recall value.

3.2. Undersegmentation error

Undersegmentation error (UE) is shown in Fig. 6(b). Given a ground truth and superpixel segmentation, this error measures the fraction of "bleeding" caused by superpixels that overlap a given ground truth segment. The standard formulation is

$$UE(s) = \frac{\sum_i \sum_{k:s_k \cap g_i \neq \emptyset} |s_k - g_i|}{\sum_i |g_i|} \tag{8}$$

where s_k are the outputs of the superpixel algorithm, g_i the ground truth segments, and $|.|$ denotes the size of an element.

There are significant changes in this evaluation according to the authors, because it is not clear how to treat pixels that lie on a boundary between two labels. In [1], the authors report a 5% tolerance margin, while authors in [12] remove the boundaries of s_k for the undersegmentation computation.

In this paper, we use the corrected undersegmentation error (CUE) proposed by the authors of SEEDS[6] defined as :

$$CUE(s) = \frac{\sum_k |s_k - g_{max}(s_k)|}{\sum_i |g_i|} \tag{9}$$

where $g_{max}(s_k)$ indicates the matching ground truth segment of s_k with the largest overlap. This corrected undersegmentation error measure seems more accurate and does not relie on any ad-hoc solution.

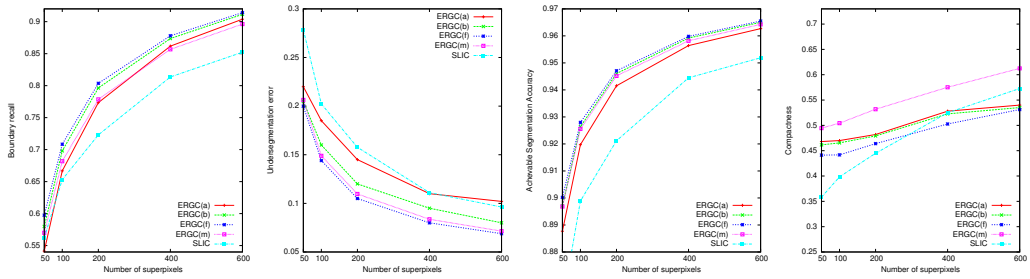
Finally this fraction is simply averaged across all ground truth segments and all images. Fig. 6(b) shows that the undersegmentation error of ERGC is the lowest upon the considered state–of–the–art methods.

3.3. Achievable Segmentation Accuracy

Achievable Segmentation Accuracy (ASA) is a performance upperbound measure. It gives the highest performance when taking superpixels as units for object segmentation. Each superpixel is labeled with the label of the ground truth segment with the largest overlap. The fraction of correctly labeled pixels is the achievable accuracy :

$$ASA(s) = \frac{\sum_k \max_i |s_k \cap g_i|}{\sum_i g_i} \tag{10}$$

[6] `http://arxiv.org/pdf/1309.3848v1.pdf`



(a) BR evolution (b) UE evolution (c) ASA evolution (d) COMP evolution

Figure 5: Performances evolution with the refinement steps : ERGC(a) stands for the initial pass of the algorithm, ERGC(b) after adding superpixels (Section 2.5), and ERGC(f) the final results after adding and moving superpixels (Section 2.6). ERGC(m) shows performances of the algorithm with the spatial constraint $m = 10$ (equation 12).

Fig. 6(c) shows that ERGC gives the best Achievable Segmentation Accuracy compared to the other methods.

3.4. Compactness

We introduce this metric in addition to traditional ones to measure the compactness of the superpixels. Compactness represents the degree to which the superpixel shape is compact. It is defined as the ratio of the area of a region to the area of a circle with the same perimeter. It is calculated as follows:

$$COMP(s_k) = \frac{4\pi|s_k|}{p_k^2} \tag{11}$$

where p_k is the perimeter of the superpixel s_k. The compactness is equal to 1 for a disc, $\pi/4$ for a square. Fig. 6(d) plots the mean compactness for each algorithm and for a different number of superpixels. Compactness for the grid of square superpixels (GRID) is the highest (around 0.85), and is not shown on Fig. 6(d).

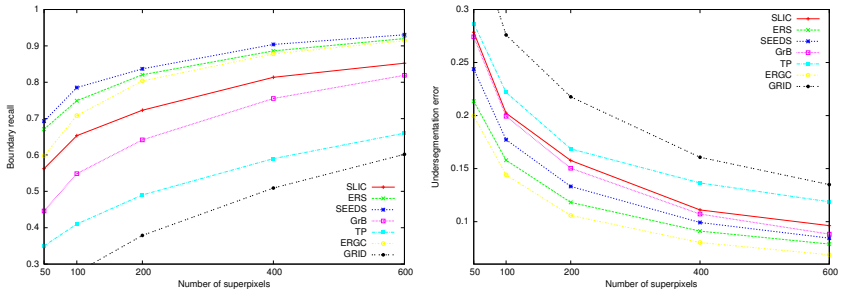
Figure 5 plots the evaluation of the proposed algorithm and its evolution within the two proposed refinement : the curves ERGC(a), ERGC(b) and ERGC(f) stand for the algorithm performances without any refinement, with only the adding superpixels refinement, and with the two proposed refinement respectively. One can particularly appreciate the improvements of the performances while refining iteratively the superpixels. Figure 5 also plots the performances of the proposed algorithm with a spatial constraint ($m = 10$ in equation 12) and SLIC (for comparison purposes). Adding a spatial constraint decreases slightly the BR, UE, and ASA performances of the algorithm in comparison to a potential based on color distances only (Equation 3), but produces more compact superpixels. Moreover, these performances are still higher than the popular algorithm SLIC.

4. Extensions

In this section, we propose several extensions to our initial framework, that illustrates the flexibility of the approach.

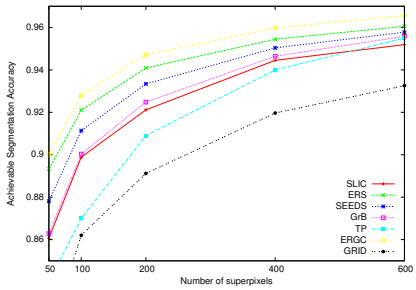
4.1. 3D extension

ERGC has been introduced for 2D images. It naturally extends to 3D volumes to produce supervoxels with minor modifications of the algorithm. Fig. 8 displays a supervoxels segmentation of a 3D volume from the MICCAI-2007 Grand Challenge dataset. For display purposes, only the interior part of the body is shown.

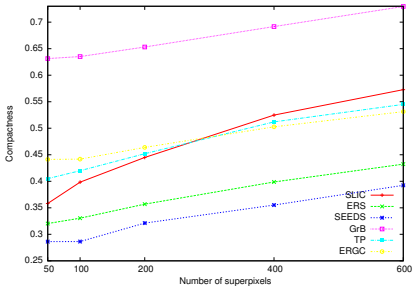


(a) Boundary Recall

(b) Undersegmentation Error



(c) Achievable Segmentation Accuracy



(d) Compactness

Figure 6: Quantitative comparison of boundary recall (a), under–segmentation error (b), achievable segmentation accuracy (c), and compactness (d) with different numbers of superpixels.

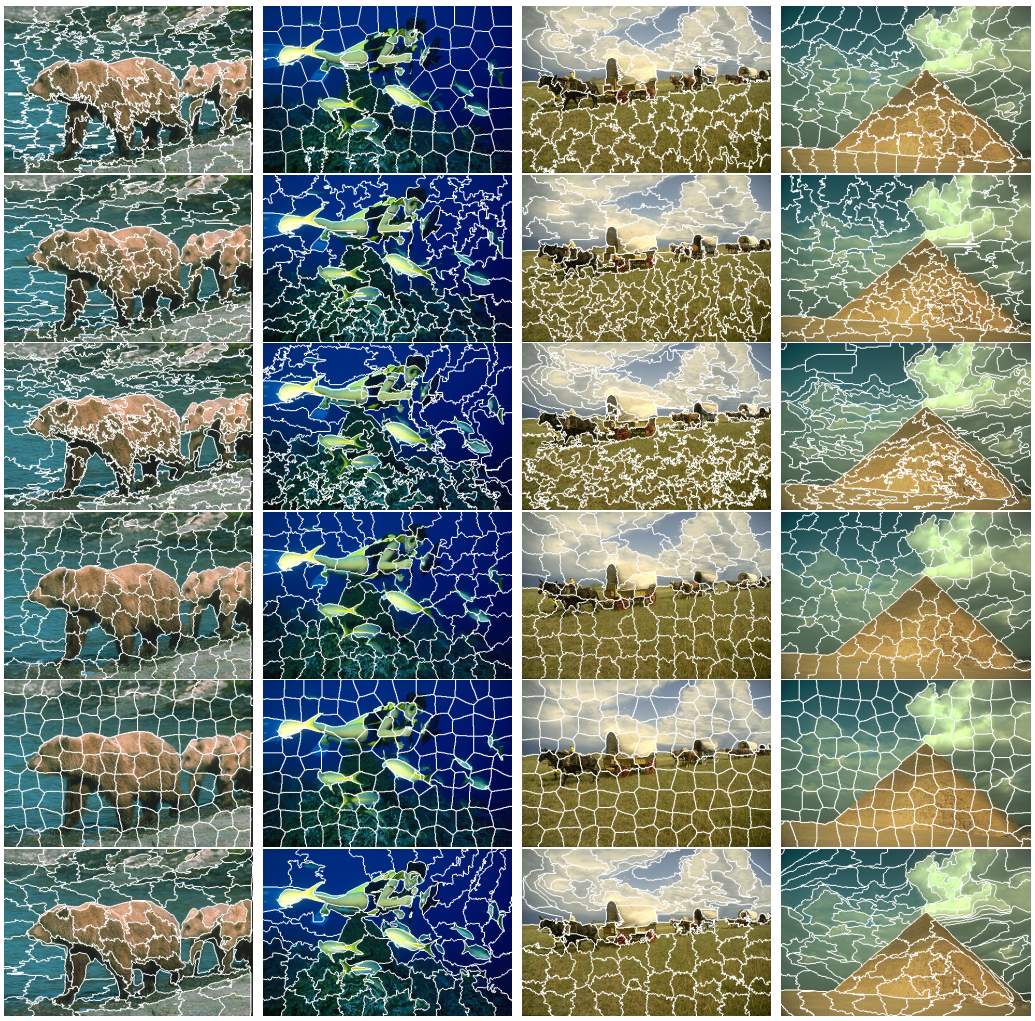
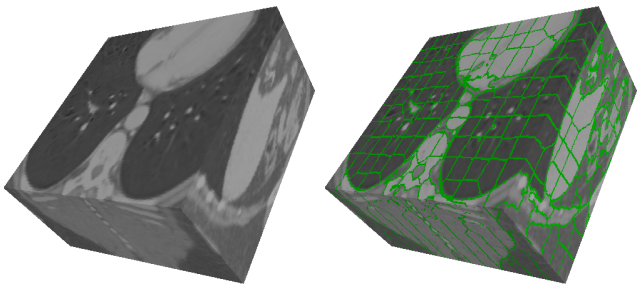


Figure 7: Visual comparison of algorithms with 100 superpixels per image. First row: SLIC, second row: ERS, third row: SEEDS, fourth row: GrB, fifth row: TurboPixels, sixth row: ERGC.



(a) (b)

Figure 8: ERGC supervoxels for a Computed Tomography volume of a human body. Only the interior part of the body has been considered for visualization purposes.

4.2. Spatial constraint

For clarity of the display purposes, a spatial constraint has been added to F to generate images of Figure 3, 4 and 8. This constraint, similar to the one proposed in [1], penalizes pixels $\boldsymbol{p}$ that are far from an initial seed $\boldsymbol{s}_i$, and is of the form $\frac{\|\boldsymbol{p}-\boldsymbol{s}_i\|_2^2}{S} \times m$ where m is the constraint parameter. In this case, the potential function is of the form:

$$F = \|\boldsymbol{C_p} - \boldsymbol{C}_i\|_2^2 + \frac{\|\boldsymbol{p} - \boldsymbol{s}_i\|_2^2}{S} \times m \tag{12}$$

Adding such a spatial constraint increases the superpixels compactness and tends to produce square superpixels in flat areas, see Figures 3, 4. Nevertheless, as this is a constraint applied on the diffusion, it decreases slightly BR, UE and ASA performances.

4.3. Combining ERGC with Edges

The proposed approach can easily be extended with additional terms to produce more powerful potential functions. We propose to combine the initial

color-based potential function with edges maps. Based on structured forest, the approach proposed in [4] gives for each pixel $\boldsymbol{p}$ the probability $E(p)$ that $\boldsymbol{p}$ belongs to an edge. Within the fast-marching algorithm, the potential of pixels belonging to region R_i is computed as :

$$F = \|\boldsymbol{C_p} - \boldsymbol{C}_i\|_2^2 \cdot \left(\epsilon + \max_{\gamma_{\boldsymbol{p}_i \to \boldsymbol{s}_i}} E\right) \tag{13}$$

where ϵ is a small constant to avoid F being zero, and $\max_{\gamma_{\boldsymbol{p}_i \to \boldsymbol{s}_i}}(E)$ is the maximum edge probability along the geodesic $\gamma_{\boldsymbol{p}_i \to \boldsymbol{s}_i}$ (see Figure 9).

Adding such an edge information to the potential is useful in case of microtextures such grass for instance. Edges probability is such areas is close to zero, making the potential of the pixels low. The propagating front is then not slowed down by the locally varying colors of the texture. Figure 11 compares the performances of this variant of ERGC, together with the initial ERGC method (without edges) and the top performer among the state-of-the-art methods for each metric. Note that compacity values are not plotted for this method since there is no significant changes from the initial ERGC method (Figure 6).

4.4. Iterative ERGC

The method proposed in this paper (detailed in Section 2) computes the superpixels in only one iteration. In this section, we go beyond this limitation and propose an iterative version of ERGC. Iterating the process allows to refine the seeds of the superpixels such that the next iteration produces better regions.

Given an initial superpixel R_i with its seed $\boldsymbol{s}_i$, the refinement is performed as follows :

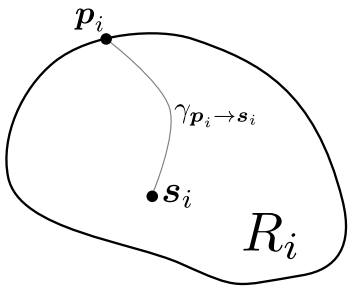


Figure 9: Computation of the edge value of $\boldsymbol{p}_i$ as the maximum edge probability along the geodesic $\gamma_{\boldsymbol{p}_i \to \boldsymbol{s}_i}$.

1. Select all the pixels $\boldsymbol{p}_i \in R_i$ such that their color is the closest from $\boldsymbol{C}_i$,
2. from this set of seed candidates, select only those that are spatially the closest from $\boldsymbol{s}_i$.

This refinement scheme is illustrated in Figure 10. It may produce several new seeds per superpixels. In such a case, these seeds share the same label, and form, after the diffusion, a sole superpixel. The spatial selection is performed to avoid the generation of too many new seeds that may produce a degenerate superpixel after several iterations.

In our experiments, the number of iteration is fixed to 10 and the configuration of seeds that lowers $\int_I U$ is retained.

At the cost of multiple iterations, this iterative variant enhances the whole performances of the algorithm (Figure 11).

For completeness purposes, Figure 11 also shows the performances of the iterative variant of ERGC with the use of edges maps. Finally, a comparison of relative processing times among all the methods is provided in Table 1.

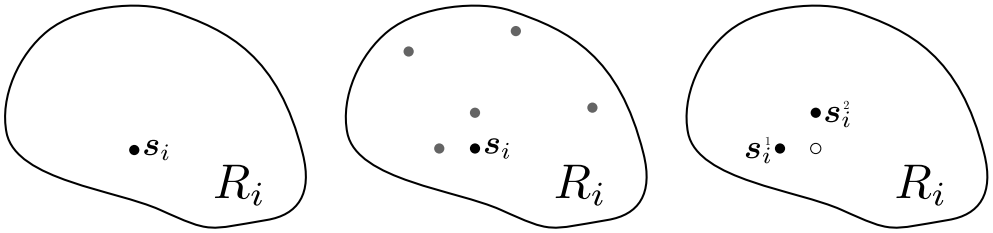
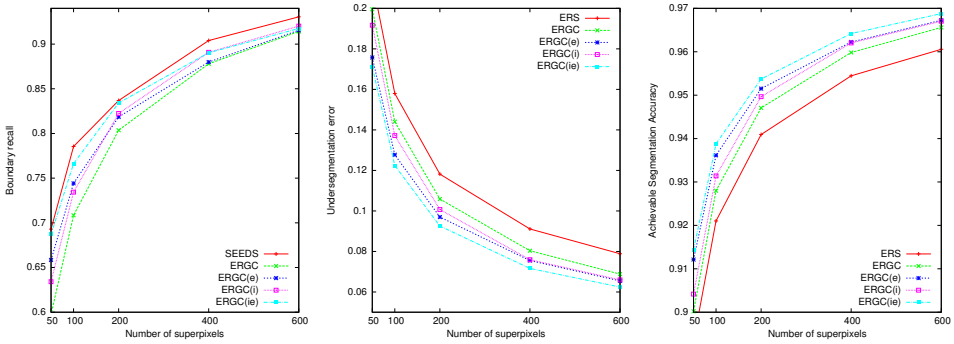


Figure 10: Illustration of the iterative refinement of the seed $\boldsymbol{s}_i$ of a superpixel R_i (left). The pixels whose color is the closest from the superpixel color C_i form a candidate set (gray dots, middle figure). From this set, only the spatially closest pixels ($\boldsymbol{s}_i^1$ and $\boldsymbol{s}_i^2$) are retained (right). Both $\boldsymbol{s}_i^1$ and $\boldsymbol{s}_i^2$ take the label of $\boldsymbol{s}_i$.



(a) BR comparison (b) UE comparison (c) ASA comparison

Figure 11: Performances comparison between ERGC with the use of edges (ERGC(e)), iterative ERGC (ERGC(i)), iterative ERGC with edges (ERGC(ie)), ERGC, and the top performer among the state-of-the-art methods for each metric.

Method	reference	Processing time factor
SLIC	[1]	0.94
ERS	[12]	5.2
SEEDS	[18]	0.91
GrB	[3]	0.95
TP	[10]	20.2
ERGC	This paper	1 (0.4s)
ERGC(e)	This paper	2.8
ERGC(i)	This paper	11
ERGC(ie)	This paper	13.8

Table 1: Relative processing times of several methods according to ERGC on a Berkeley dataset image, with 100 superpixels. These processing times may slightly vary with different numbers of superpixels. ERGC(e) and ERGC(ie) processing times include edge maps computation.

5. Conclusion

Superpixels have become an important preprocessing tool for many image based applications. In this paper, we proposed a method based on the eikonal equation that quickly creates accurate superpixels. Through the empirical experiments, we showed that ERGC outperforms existing superpixel methods in two of three metrics, while being outperformed by SEEDS and ERS on Boundary Recall. This last result has to be appreciated in the light of their superpixel compactness, the lowest within the state-of-the-art.

Proposed here for the generation of superpixels/supervoxels, we think that the diffusion framework can offer an interesting choice in many other image processing tasks. ERGC can also be extended to arbitrary graphs to perform local or non-local image and data processing, and data clustering

(supervertices).

6. Acknowledgements

This work is part of the SIRTDose project funded by the grant "Physique Cancer" 2012 (INSERM - Plan Cancer). The authors wants to thank the authors of [1] and [18] for the fruitlul discussions about superpixels and their evaluation. The authors also would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[2] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[3] Xavier Desquesnes, Abderrahim Elmoataz, and Olivier Lézoray. Eikonal equation adaptation on weighted graphs: Fast geometric diffusion process for local and non-local image and data processing. *Journal of Mathematical Imaging and Vision*, pages 1–20, 2013.

[4] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *International Conference on Computer Vision*. IEEE, 2013.

[5] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. in press.

[6] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[7] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670–677. IEEE, 2009.

[8] Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3):300–316, 2008.

[9] S. Kim. An $\mathcal{O}(n)$ level set method for eikonal equations. *SIAM journal on scientific computing*, 22(6):2178–2193, 2001.

[10] A. Levinshtein, A. Stere, K.N. Kutulakos, D.J. Fleet, S.J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2290–2297, 2009.

[11] O. Lézoray and L. Grady. *Image Processing and Analysis with Graphs: Theory and Practice*. Digital Imaging and Computer Vision. CRC Press / Taylor and Francis, 2012.

[12] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2097–2104. IEEE, 2011.

[13] Alastair P Moore, Simon JD Prince, and Jonathan Warrell. 'lattice cut'-constructing superpixels using layer constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2117–2124. IEEE, 2010.

[14] A.P. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[15] G. Peyré, M. Péchaud, and R. Keriven. *Geodesic methods in computer vision and graphics*. Now publishers Inc, 2010.

[16] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[17] J.A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[18] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. Seeds: superpixels extracted via energy-driven sampling. In *Computer Vision–ECCV 2012*, pages 13–26. Springer, 2012.

[19] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. *Computer Vision–ECCV 2008*, pages 705–718, 2008.

[20] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. *Computer Vision–ECCV 2010*, pages 211–224, 2010.

[21] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE transactions on pattern analysis and machine intelligence*, 13(6):583–598, 1991.

[22] Yi Yang, Sam Hallman, Deva Ramanan, and Charless Fowlkes. Layered object detection for multi-class segmentation. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3113–3120. IEEE, 2010.

[23] L. Yatziv, A. Bartesaghi, and G. Sapiro. $\mathcal{O}(n)$ implementation of the fast marching algorithm. *Journal of computational physics*, 212(2):393–399, 2006.

[24] C Lawrence Zitnick and Sing Bing Kang. Stereo for image-based rendering using image over-segmentation. *International Journal of Computer Vision*, 75(1):49–65, 2007.

[25] CW Zitnick, Nebojsa Jojic, and Sing Bing Kang. Consistent segmentation for optical flow estimation. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1308–1315. IEEE, 2005.