A Tutorial on Applications of Power Watershed Optimization to Image Processing

Sravan Danda, Aditya Challa, B S Daya Sagar, Laurent Najman

EPJ manuscript No.
(will be inserted by the editor)

A Tutorial on Applications of Power Watershed Optimization to Image Processing

Sravan Danda[1], Aditya Challa[2], and B. S. Daya Sagar[3] and Laurent Najman[4,a]

[1] Computer Science and Information Systems, APPCAIR, BITS-Pilani K K Birla Goa Campus, Goa, India
[2] Computer Science and Automation, Indian Institute of Science, Bangalore, India
[3] Systems Science and Informatics Unit, Indian Statistical Institute, Bangalore, India
[4] Université Paris-Est, LIGM, Equipe A3SI, ESIEE, Paris, France

Abstract. This tutorial review paper consolidates the existing applications of the power watershed (PW) optimization framework in the context of image processing. In literature, it is known that PW framework when applied to some well-known graph-based image segmentation and filtering algorithms such as random walker, isoperimetric partitioning, ratio-cut clustering, multi-cut and shortest path filters yield faster yet consistent solutions. In this paper, the intuition behind the working of PW framework i.e. exploitation of contrast invariance on image data is explained. The intuitions are illustrated with toy images and experiments on simulated astronomical images. This article is primarily aimed at researchers working on image segmentation and filtering problems in application areas such as astronomy where images typically have huge number of pixels. Classic graph-based cost minimization methods provide good results on images with small number of pixels but do not scale well for images with large number of pixels. The ideas from the article can be adapted to a large class of graph-based cost minimization methods to obtain scalable segmentation and filtering algorithms.

1 Introduction

Machine vision applications are diverse ranging from segmentation, filtering, object detection, object localization, instance segmentation, semantic segmentation, image classification, image captioning, and image reconstruction etc [49]. Image segmentation and filtering form the building blocks of many machine vision tasks. Historically, energy-based or cost minimization-based approaches have been popularly used to solve image segmentation and filtering. The cost functions are designed such that the minimizers yields desired results. Also, the cost functions are constructed to incorporate an inductive bias in images namely translation invariance. In the context of image classification, translation invariance means - if object(s) in an image are vertically or horizontally translated within the image, this image is still expected to be classified as the same category (or set of categories). CNNs takes this approach to the next level, by allowing many parameters to be learned by gradient descent, with

[a] e-mail: laurent.najman@esiee.fr

the very same cost function, and allowing the system to compute parameters that are otherwise difficult to predict or to tune. Some works additionally capitalize on rotation invariance by augmenting the training datasets with customized rotations. This helped to improve the performance of the cost minimization-based approaches further.

One of the less exploited aspects of image data is that of contrast invariance. Intuitively, it is clear that a human eye perceives the same set of objects in an image even if the contrast of the image is changed (see Fig 1). In fact, the object boundaries are also expected to be intact when the contrast of the image is altered. An obvious question arises - how does one exploit the contrast-invariance nature of solutions to image processing tasks? In this article, we provide an answer to this question in the context of image segmentation and image filtering. We revisit the power watershed (PW) framework [39] and show that this optimization framework provides a formalism to impose the constraint that the solutions are contrast-agnostic. It is well known that an application of PW to classic graph-based cost minimization methods results in high-quality approximation solutions to the cost minimization problem [1,17,18, 21,11,10,23,22,12,56,54,55]. In other words, PW exploits contrast-agnostic nature of solutions and allows one to apply classic graph-based cost minimization methods to images with very large number of pixels which were otherwise prohibited by computational constraints. In particular, astronomical images which have a huge number of pixels can be segmented and/or filtered with graph-based cost minimization methods by using PW framework.

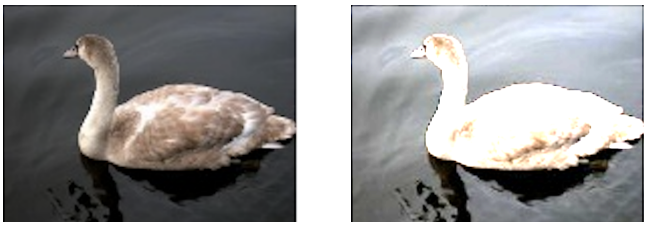

Fig. 1. Left: An image from the Weizmann 1-object database [2]. **Right**: Enhanced contrast of the image on left. Observe that the object boundaries are not expected to change even in the contrast enhanced image. One expects that a segmentation method applied on either of these images yields the same result.

Recall that image segmentation and filtering form the building blocks of many image processing applications. Image segmentation and image filtering are typically used either as a pre-processing or a post-processing step depending on the machine vision task at hand. Image segmentation is an ill-posed problem. The goal of image segmentation is to cluster pixels in an image such that the clusters are 'close' to the clusters obtained by a domain expert w.r.t. some standard measures that compare closeness of clusters. Typically, each cluster obtained by a domain expert contains pixels only from one object and the number of clusters are usually much smaller when compared to the total number of pixels in the image (see Fig 2 for a segmentation of an image obtained by a domain expert using two labels). Image filtering is another closely related ill-posed problem. Image filtering is the process of magnifying certain details while suppressing the others. Informally, image filtering can be interpreted as summarizing the content of an image by removing redundant details. For example, in Fig 2, in order to identify the object boundaries, details such as feathers etc are irrelevant and can be ignored. Although, in this article, we deal with only image

segmentation and image filtering problems specific to 2D images, the techniques we present are generic and can be applied to any kind of image data.

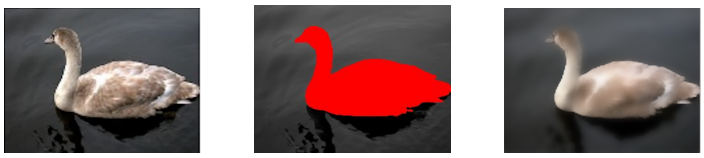

Fig. 2. Left: An image of dimensions 548×402 from the Weizmann 1-object database [2]. **Middle**: Two labels are used to segment the pixels. The pixels belonging to the foreground are identified by the colour-code red. **Right**: The image on left is filtered using a bilateral filter [50]. A bilateral filter suppresses the texture details within objects which are irrelevant for summarizing the information on object boundaries in the image. The boundaries between the foreground and the background are crucial for image segmentation and are preserved.

Recall that a 2D digital image consists of a finite number of pixels. Each pixel represents a small physical area and typically these pixels are square shaped. To each pixel, one associates a scalar or vector that represents the average intensity/color reflected by the corresponding area. A 2D digital image of dimensions $M \times N$ is thus a matrix of scalar/vector-valued entries with M rows and N columns.

$$I : \{0, 1, \cdots, M-1\} \times \{0, 1, \cdots, N-1\} \to \mathcal{Z}^{\rho}, \tag{1}$$

where $\mathcal{Z}$ is a discrete set consisting of non-negative real numbers and and ρ denotes the number of bands ($\rho = 1$ in case of greyscale images and $\rho = 3$ in case of color images). However, in this article, we use edge-weighted graphs to model images and work with those models instead of Eq 1. This is because local changes such as gradients etc that provide information on object boundaries can be captured using the edge weights. Recall that a gradient is a dissimilarity measure between neighbouring pixels. A 2D image is represented as a 4-grid graph or a 4-adjacency graph (Von Neumann neighborhood) with vertices representing the pixels and the edge weights reflecting a similarity/dissimilarity between neighbouring pixels. Formally, if I is an image of dimensions $M \times N$, we have $\mathcal{G}_I = (V, E, W)$ where V, the set of vertices represents the pixels i.e.

$$V = \{(i, j) : 0 \leq i \leq M-1, 0 \leq j \leq N-1\}, \tag{2}$$

the set of edges are given by

$$E = \bigcup_{0 \leq i \leq M-1, 0 \leq j \leq N-1} E_{ij} \tag{3}$$

$$E_{ij} = \{\{(i', j'), (i, j)\} : |i - i'| + |j - j'| = 1 \; and \; 0 \leq i' \leq M-1, 0 \leq j' \leq N-1\}, \tag{4}$$

for each $0 \leq i \leq M-1, 0 \leq j \leq N-1$, and $W : E \to \mathbb{R}^+$ is a non-negative real valued function on the set of edges. The edge weights either represent a similarity or a dissimilarity depending on the application at hand. Typically, the edge weights are obtained by using monotonic functions of a standard norm of the difference of the pixel intensities/colors of pixels incident on the edge.

Recall that the increase (respectively decrease) in contrast of an image in the classic sense is obtained by multiplying intensity/each coordinate in the color vector of each pixel by a constant non-negative number greater (respectively lesser) than one.

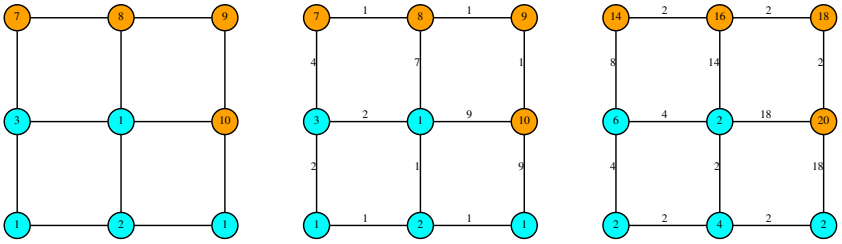

Fig. 3. Left: A toy 2D greyscale image of dimensions 3×3 containing two objects is represented with a vertex-weighted graph. Each pixel in the image corresponds to a vertex in the graph. The pixel intensities are displayed inside the circled vertices. Vertices are coloured with two distinct colours to signify that they belong to different objects. **Middle**: The image is represented as an edge-weighted graph. The weights of edges are obtained by the absolute difference of image intensities and hence represent a dissimilarity measure between the neighbouring pixels. **Right**: A change in the contrast of the image can be represented by a scaling of the weights. Observe the intensities of the pixels and the weights of the edges are doubled.

For the edge-weighted graph model, this is equivalent to: the contrast of an image is increased (respectively decreased) when the edge weights are magnified (respectively diminished). Mathematically, a magnifying (respectively shrinking) operation on the edge weights is a function on the edge weights with derivative greater (respectively lesser) than one. Also, a change in contrast does not alter the relative ordering of edge weights. Mathematically, this is equivalent to restricting the operations on the edge weights to be strictly increasing functions. Thus, the answer to the question posed in the second paragraph i.e. contrast invariance can be obtained by considering only the: '*solutions that are invariant to strictly increasing functions on the edge weights of the graph constructed on the image*'. Typically, a graph constructed to model an image is referred to as the image graph irrespective of whether the edge weights reflect a similarity or a dissimilarity or a combination of both. In the rest of the article, we use the term 'image graph' to refer to the 'graph constructed on an image'.

In literature, there are plenty of graph-based methods to obtain image segmentation and filtering. For segmentation, broadly they fall under two overlapping categories namely graph-partitioning methods [51,3] and variational cost minimization [37]. Both these classes of methods typically optimize a cost defined on the image graph. Some commonly used segmentation methods in these categories are graph-cut [7], shortest-path segmentation [24], random walker [27], ratio cut [53], normalized cut [53,48], watershed cut [19], multi-cut [51] and isoperimetric partitioning [29]. For image filtering, weighted average filters such as shortest-path filters/morphological amoebas [35,30], tree filter [6,57] are graph-based and yield good results. However, in practice, these methods cannot be applied to images with large number of pixels as their asymptotic complexity is very high.

The methods mentioned in the previous paragraph are based on cost minimization on an edge-weighted graph. Hence, the final segmentation/filtering result is a function of the edge weights of the image graph. The algorithms that are originally introduced to implement these methods take the actual weights into consideration. In order to ensure contrast invariance, one needs to consider only the relative ordering of the edge weights instead of the actual weights in the cost minimization. This is the core idea of the PW framework. Given a cost-minimization problem, PW framework transforms the cost-minimization problem into a series of nested cost-minimization problems. The transformation is such that the set of nested cost-minimization problems remain the same even if the edge weights of the graph are changed without altering their relative

ordering. Solving the transformed problem is computationally easier than solving the original minimization problem. Additionally, the results obtained on the transformed problem are similar to those obtained on the original minimization problem. Thus, application of PW to graph-based cost minimization methods helps in scaling these methods to images with large number of pixels.

The rest of the article is organized as follows: In Sec 2, graphs used in image processing within the scope of the article are briefly described. Sec 3 provides an explanation of the PW optimization framework in the context of image processing. The next four sections i.e. Sec 4, Sec 5, Sec 6, and Sec 7 provide a comprehensive survey of applications of PW to classic graph-partitioning and variational cost minimization image segmentation methods. These methods are explained intuitively using toy images, simulated astronomical images, and a few figures replicated from the original articles. Sec 8 contains a brief description of the utility of PW framework in explaining the links between shortest path-based filters and spanning tree-based filters. Sec 9 contains experiments on simulated astronomical sky images [43,31]. It is illustrated that the PW counterparts yield similar results as that of the classic methods at a lower computational cost. The conclusions section summarizes the article and provides some pointers on how to use the ideas from the article to build scalable algorithms in the context of graph-based image segmentation.

2 Graphs in Image Processing

Graphs are discrete mathematical objects. They are popularly used for data analysis as there is abundant literature available on graph algorithms. Although several variants of graph models exist, the description of graphs in this section is restricted to those essential to follow the rest of the article. The simplest of a graph model consists of two objects: a set of vertices, a subset of unordered pairs of vertices called edge set or simply edges. Two vertices are said to be adjacent to each other if the unordered pair of these vertices belongs to the edge set. In the context of images, one can identify each pixel with a vertex in the graph. As pixels in a 2D image are aligned 'nicely' like a grid, a 4-adjacency relation (see Fig 3) is a popular choice for modelling images. In other words, each vertex is adjacent to exactly four other vertices (except at the borders of the image) viz. nearest vertices one in each of left, right, top and bottom. See Eq 3 for a formal definition.

Each pixel has a grey value (or a triplet of values corresponding R, G, and B bands in case of colour images), one can assign a mapping on the set of vertices. However, as described in the previous section, it is convenient to model local changes. This is done by assigning weights to edges. A classic usage is to consider a discrete gradient given by the absolute difference between the pixel values (euclidean distance between triplets) of the adjacent vertices in the case of greyscale images (colour images). This can be viewed as a dissimilarity between the neighbouring pixels (see Fig. 3). In general, if the weights of edges are constructed so as to capture dissimilarity between the adjacent vertices then the graph is said to be a dissimilarity-based edge-weighted graph. These kinds of graphs are useful for image filtering applications (see Sec 8 for details).

On the other hand, for many image processing methods, it is convenient to capture similarity between adjacent pixels. The edge-weighted graphs with edge weights reflecting similarity between adjacent pixels are known as similarity-based edge-weighted graphs. These are particularly useful in applications where one wants to pose the objective as a cost minimization problem (Fig 4 illustrates this fact. See Sec 5, Sec 6 and Sec 7 for details). For example, if one is interested to find a segmentation of the image with a known number of segments k, the problem can be posed as finding

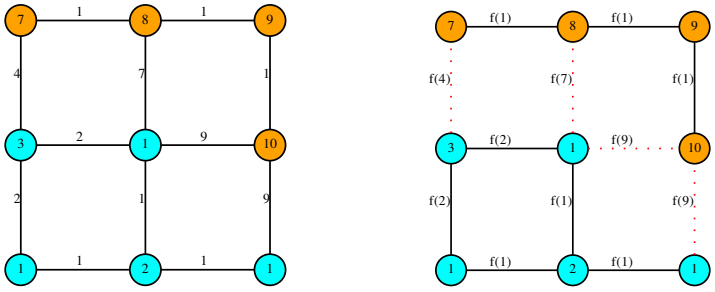

Fig. 4. Left: A toy 2D greyscale image containing two objects is visualized with a edge-weighted graph. Each pixel in the image corresponds to a vertex in the graph. The pixel intensities are displayed inside the circled vertices. Vertices are coloured with two distinct colours to signify that they belong to different objects. The edge weights are given by absolute difference of grey values of adjacent pixels **Right**: The edge weights are transformed to reflect similarities i.e. higher the similarity, higher the weight. Here $f(x) = \exp(-x)$. Image segmentation can be posed as a minimization problem i.e. find a set of edges such that the sum of its weights is minimum and its removal results in two pieces (as there are two objects in the image). An optimal set of edges are indicated by highlighting them as dotted edges.

a set of edges such that the sum of the weights of these edges is minimum and their removal results in k pieces (called 'components' in graph theory terminology). This yields good results in practice as most of the pairs of vertices corresponding to low-weight edges w.r.t. a similarity measure belong to different objects. Fig 4 illustrates for $k = 2$, also known as a graph-cut problem. In general, an image contains more than two objects and the corresponding mathematical generalization of the graph partitioning is referred to as a k-way cut [51].

On some occasions, prior information in the image is available. Labels of some of the pixels might be known. For example, in an astronomical sky image, some background pixels and some pixels corresponding to galaxies/stars can be identified easily. Extreme values along with low variability in the neighbourhood (see Fig 8 in Sec 4 for an illustration on marking seeds) are indicators for such pixels in the case of sky images. A basic yet important task is to segment the galaxies/stars from the background. The pixels with known labels are called seeds in image segmentation terminology. When seeds are available, one can compute the affinity between each pixel and the labelled seeds. Then, the non-seed pixels can be assigned a label corresponding to the label with maximum affinity. More often than not, these affinities are computed using 'paths' that start at the non-seed pixels and terminate at seeds. Recall that a 'path' is a sequence of distinct vertices such that every pair of consecutive vertices in the sequence are adjacent to each other. Random Walker segmentation (see Sec 4) is based on this idea.

There are other kinds of prior information on the image. For example, it might be known apriori that certain set of pairs of pixels have different labels. This information can be imposed as a constraint on the graph partitioning problem thus making it a constrained optimization problem. Multi-cut partitioning [51] is based on this idea. More generally, on some occasions, it is known that certain pairs have the same labels (similar or attractive) and certain other pairs have different labels (dissimilar or repulsive) with varying levels of confidence. A forbidden pair of pixels having the same label can be seen as a dissimilar pair with infinite affinity. Similarly a pair constrained to have the same labels can be visualized as a similar pair with infinite affinity. It is important to note that these pairs are not necessarily adjacent pixels. Hybrid edge-weighted graph models capturing both similarity and dissimilarity are used to model

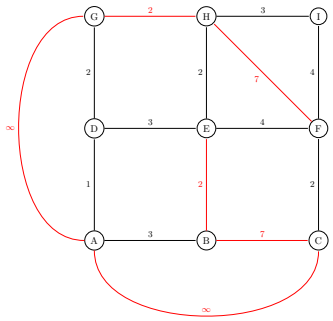

Fig. 5. Left: A hybrid edge-weighted graph capturing similarities, dissimilarities along with pairs of pixels that are forbidden to have the same labels and pairs of pixels that are required to have the same labels. A red coloured edge indicates dissimilarity and a black coloured edge indicates a similarity. The weights displayed on the edges reflects the strength of affinity. For example, a red coloured edge of value ∞ indicates a pair that is forbidden to have same labels.

such situations. We remark that such graphs have additional edges on top of the four neighbours that each pixel has, in order to capture the attractions and repulsions (see Fig 5). Each edge in the graph is either a similarity edge or a dissimilarity edge but not both. An indicator function on the edge set is used to identify whether an edge indicates a similarity or a dissimilarity. Each edge has a weight and the weight of the edge indicates the confidence on the affinity. For example, a dissimilarity edge with a high value indicates that the pair of vertices on the edge are likely to have different labels. Mutex watershed is an algorithm based on this idea. Muti-cut is closely related to Mutex watershed [56,54] and the reader may refer to Sec 7 for details.

In the context of astronomical image processing, domain knowledge of the physical characteristics of the images can be incorporated into the weights of edges in various forms: similarity, dissimilarity or a combination of both similarity and dissimilarity.

3 Power Watershed Optimization and Contrast Invariance

In Sec 2, we mentioned that edge-weighted graphs are well suited to image processing tasks. Specifically, three types of edge-weighted graphs have been described. These are dissimilarity-based, similarity-based, and hybrid edge-weighted graphs capturing both similarity and dissimilarity. Assume that the objective i.e. image segmentation or filtering is cast as a cost minimization problem on the image graph. Further, assume that the cost can be written as a linear combination of edge weights. Allowing a slight generalization, we have the following form for the cost function $Q(\mathbf{x})$:

$$Q(\mathbf{x}) = \sum_{e_{ij} \in E} f(w_{ij}) Q_{ij}(x_i, x_j), \tag{5}$$

where E is the set of edges of the image graph, e_{ij} is the edge connecting the vertices i and j, w_{ij} is the weight of e_{ij}, f is a known monotonic function (either an increasing or a decreasing function) and $Q_{ij}()$ is a smooth function (differentiable). Here $\mathbf{x}$ is a vector of size equal to the number of pixels and represents the set of target labels/values.

PW framework considers the following sequence of optimization problems:

$$Q^{(p)}(\mathbf{x}) = \sum_{e_{ij} \in E} (f(w_{ij}))^p Q_{ij}(x_i, x_j), \tag{6}$$

where p is a positive integer and the other quantities are same as that of Eq 5. Instead of minimizing the cost for each p, PW framework considers the limit of minimizers of a sequence of cost functions $(Q^{(p)}(\mathbf{x}))_{p=1}^{\infty}$ as $p \to \infty$, termed as PW limit.

An image graph is finite, so the number of edges is finite. Further, let the set of edges, E be relabelled in the increasing order of their weights. The Q_{ij} terms corresponding to identical weights can be merged. The cost function can then be as sum of l terms where $w_1 < \cdots < w_l$ are the distinct set of weights ($l \leq |E|$ holds for obvious reasons). It is shown in [39] that the limit of the minimizers can be obtained by the following algorithm:

Algorithm 1 Calculating limit of minimizers [39]

Input A sequence of cost functions indexed with $p \in \mathbb{Z}^+$ given by Eq 6
Output A limit of minimizers to Eq 6 as $p \to \infty$.

1: Set $i = l$ and M_i is the entire solution space.
2: **while** $i > 1$ **do**
3: Compute the set of minimizers $M_{i-1} = \arg\min_{\mathbf{x} \in M_i} Q_i(\mathbf{x})$
4: **Return** arbitrary $\mathbf{x} \in M_1$.

In order to better understand the intuition behind the algorithm, consider the following example (modified from [39]). Let $w > 0$ and $Q^{(p)} : \mathbb{R}^2 \to \mathbb{R}$ be defined as

$$Q^{(p)}(x_1, x_2) = w_1^p((x_1 - 1)^2 + x_2^2) + w_2^p(x_1 - x_2)^2 \tag{7}$$

where $w_1 = w$ and $w_2 = 2w$. Observe that $Q^{(p)}$ is a non-negative function for any $p > 0$. The minimizer of $Q^{(p)}$ can be computed directly and is given by

$$\hat{x_1}^{(p)} = \frac{2^p}{2^{p+1} + 1} \tag{8}$$

$$\hat{x_2}^{(p)} = \frac{2^p + 1}{2^{p+1} + 1} \tag{9}$$

With simple calculus, it is easy to verify that the sequence of minimizers of $Q^{(p)}$ i.e. $((\hat{\mathbf{x}})^{(p)})_{p>0} = ((\hat{x_1}^{(p)}, \hat{x_2}^{(p)}))_{p>0}$ converges to $(\frac{1}{2}, \frac{1}{2})$ as $p \to \infty$.

On the other hand, application of Algorithm 1 translates to rewriting $Q^{(p)}(x_1, x_2)$ as follows:

$$\frac{Q^{(p)}(x_1, x_2)}{w_2^p} = \frac{((x_1 - 1)^2 + x_2^2)}{2^p} + (x_1 - x_2)^2 \tag{10}$$

This quantity behaves like the function

$$(x_1 - x_2)^2 \tag{11}$$

as $p \to \infty$

At the first pass, our search space of solutions is restricted to the subspace $\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 = x_2\}$. At the next pass, within the restricted subspace of the solutions, the point that minimizes $((x_1 - 1)^2 + x_2^2)$ is easily seen to be $(\frac{1}{2}, \frac{1}{2})$.

Recall that in order to ensure contrast invariance, the optimization problem has to remain the same irrespective of changes to the edge weights subject to preserving their relative ordering. It can be seen from Algorithm 1 that the limit of minimizers of $Q^{(p)}(\mathbf{x})$ depends only on the relative ordering of the edges and not the actual weights. Also, it is worth noting that Algorithm 1 decomposes the cost function on the original graph into costs on smaller subgraphs. It is shown in [17,18,21,11,10,23,22,12] that the computation of said limit is easier than minimizing the original cost $Q(\mathbf{x})$. In Sec 4, Sec 5, Sec 6 and Sec 7, it will be demonstrated that the quality of the segmentation is retained while reducing the computational cost.

It is important to mention that the PW limit of a cost of the form Eq 5 depends only on a substructure of the image graph (known as a subgraph in graph theory). This subgraph is either union of maximum spanning trees (UMaxST) or the union of minimum spanning trees (UMinST) depending on whether the edge weights reflect similarity or dissimilarity. UMaxST (respectively UMinST) is the induced subgraph generated by set of edges that belong to at least one 'maximum spanning tree' (respectively 'minimum spanning tree') of the image graph. Recall that a 'spanning tree' is a subset of edges of the image graph such that there exists a unique path between every two vertices in the image graph when the paths are restricted to contain only edges from the subset of edges (see Fig 6 for an illustration). A maximum spanning tree (MaxST) is a spanning tree such that no other spanning tree has the sum of its edge weights larger than that of MaxST. A similar definition holds for a minimum spanning tree (MinST).

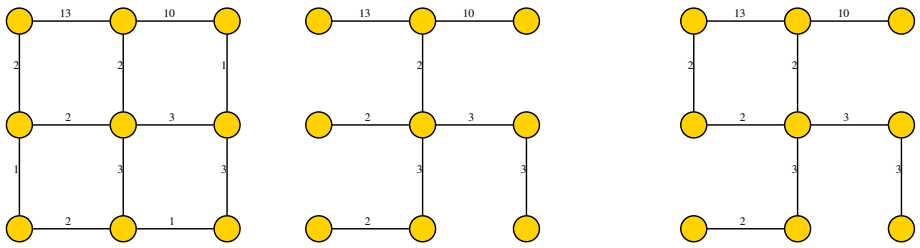

Fig. 6. Left: An edge-weighted graph. **Middle**: A MaxST of the graph on left. **Right** UMaxST of the graph on left.

4 Fast Random Walker Segmentation

Recall that in some practical applications, prior information of labels of some pixels in the image is available. Random Walker is a seeded segmentation method and is handy for such situations.

4.1 Classic Random Walker Segmentation

Random Walker (RW) can be described as follows: A similarity-based edge-weighted graph is constructed on the image. For each non-seed, several random walks are simultaneously propagated to each of its neighbours and the process is recursively repeated. At each step, it is assumed that a random walk splits into multiple unvisited vertices independently. Further, the branching probability of an edge on a vertex is equal to the proportion of similarity of the edge traversed to the sum of similarities of

all possible edges that can be traversed at that step. For example, in Fig 7, a random walker starting at vertex 1 selects the edge $1-2$ with probability $\frac{2}{3}$ as there are two edges emanating from 1 namely $1-2$ with weight 2 and $1-4$ with weight 1. These paths are recursively propagated until they terminate on labelled points i.e. seeds. For example, a random walker starting at the vertex 1 that already chose the first edge as $1-2$ will choose the edge $2-3$ with conditional probability $\frac{1}{4}$. Note that the random walker has only two choices of edges namely $2-5$ and $2-3$ at this stage as vertex 1 is already visited. The probability of a non-seed pixel having a given label is then calculated by summing up probabilities of all the paths that start from the non-seed pixel to all seeds with that label (see Fig 7 for an illustration). A non-seed is then assigned the label to which the RW probability is highest. For example, in Fig 7, vertex N would be assigned label B.

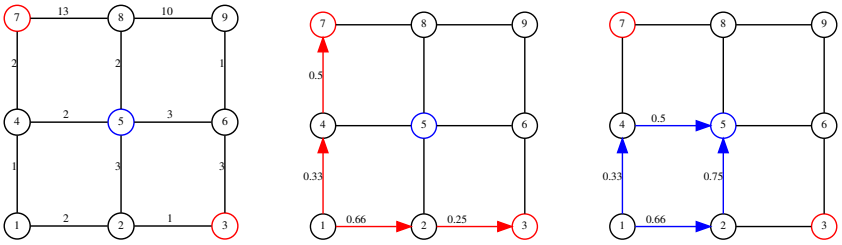

Fig. 7. Left: A similarity-based edge-weighted graph with two labels coloured blue and red with three seeds, two seeds with colour red, and one seed with colour blue. **Middle**: The paths from vertex 1 to seeds with red label are highlighted in red. Random walk probabilities are displayed on the corresponding edges. The probability that the vertical path from vertex 1 to red seed vertex 7 is 0.33*0.5. Similarly, the probabilities are computed for the horizontal path from 1 to 3 which is also a red seed. Adding these probabilities yields the RW probability for vertex 1 to be labelled red. **Left**: The paths from 1 to seed with the label blue are highlighted. The path probability computations are performed similarly. The RW probability for the vertex to be labelled either red or blue would add up to 1.

Recall that a seeded classification problem with multiple labels can also be perceived as a collection of binary classification problems. The number of such binary classification problems would be as many as the number of distinct labels. Let $\mathcal{L}$ denote the set of distinct labels. Each such binary classification problem deals with identifying label l versus NOT label l where $l \in \mathcal{L}$ is a label. Treating the multiple class classification problem this way, it was shown in [27] that RW probabilities of each such binary classification problem can be obtained by minimizing the convex cost function given by Eq 12.

$$RWCost(\mathbf{x}) = \frac{1}{2} \sum_{e_{ij} \in E} w_{ij}(x_i - x_j)^2, \ \textit{subject to } \mathbf{x}_{seed} = \mathbf{f}_{seed}, \tag{12}$$

where vector $\mathbf{x}$ refers to the target labels of all pixels, E is the set of edges, w_{ij} is the weight of e_{ij}, $\mathbf{x}_{seed}$ refers to the target labels of seed pixels and $\mathbf{f}_{seed}$ is the vector of preset labels of corresponding seeds. Note that $\mathbf{f}_{seed}$ is a vector of zeros and ones as we are dealing with a binary classification problem. More particularly, in the l versus NOT l classification problem, a coordinate in $\mathbf{f}_{seed}$ equals 1 if the corresponding pixel is labelled l and 0 if it is labelled but the corresponding label is NOT l.

Let L denote the unnormalized Laplacian of the graph [53]. One can write the cost $RWCost(\mathbf{x})$ in Eq 12 as $\mathbf{x}^T L \mathbf{x}$ upto a constant factor. Rearranging the indices

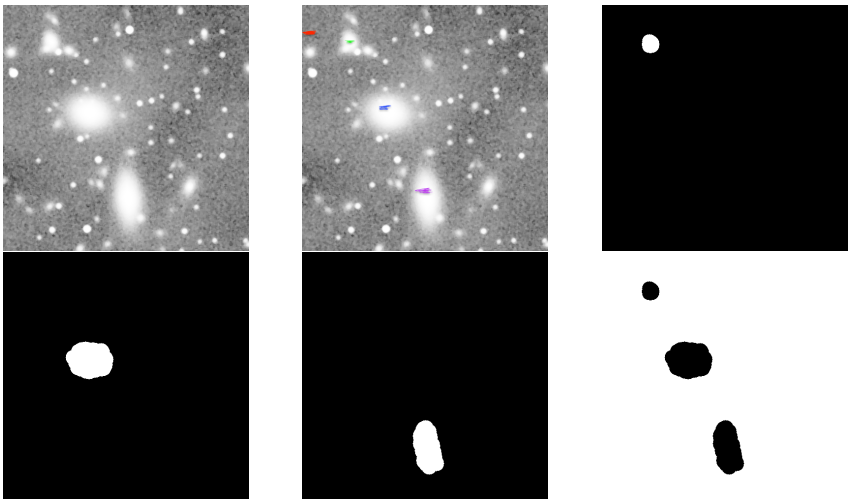

Fig. 8. Top Left: An image to be segmented. **Top Middle**: Seeds are marked with strokes. Four different labels are used for RW segmentation. **Top Left**: Visualization of RW Probability map of green label. **Bottom**: From left to right, each image is a visualization map of label RW probabilities for labels blue, violet and red in that order. White shades indicate high probability and dark indicate low probability.

of seeds and non-seeds so that the indices of all seeds appear before those of non-seeds, one can decompose L as follows:

$$L = \begin{pmatrix} L_{seed} & B \\ B^T & L_U \end{pmatrix} \tag{13}$$

$RWCost(\mathbf{x})$ can then be written as:

$$RWCost(\mathbf{x}) = \frac{1}{2}(\mathbf{x}_{seed}^T L_{seed} \mathbf{x}_{seed} + 2\mathbf{x}_U^T B^T \mathbf{x}_{seed} + \mathbf{x}_U^T L_U \mathbf{x}_U), \tag{14}$$

where $\mathbf{x}_U$ denotes the sub-vector of $\mathbf{x}$ corresponding to the unlabelled points. Applying elementary calculus, it can be seen that the solutions to the minimization problem can be obtained by solving the following linear system of equations:

$$L_U \mathbf{x}_U = -B^T \mathbf{x}_{seed} \tag{15}$$

Recall that the number of distinct labels in the multiple class classification problem is $|\mathcal{L}|$. Hence, one needs to compute an affinity vector of length $|\mathcal{L}|$ to encode probabilities of the unlabelled points to each of the distinct labels. In other words, Eq 15 transforms to solving

$$L_U X = -B^T S \tag{16}$$

where X is a matrix of shape $|\mathbf{x}_U| \times |\mathcal{L}|$ and S is a matrix of size $|\mathbf{x}_{seed}| \times |\mathcal{L}|$

An application of RW on a real image is illustrated in Fig 8 (this demonstration is similar to the illustration on the medical image from the original paper on RW [27]).

4.2 Power Watershed Approximation to Random Walker

It can be seen that Eq 12 is in the form of Eq 5. In [18,17], the RW cost function is recast in the form of Eq 6 using the PW framework. The cost minimization is

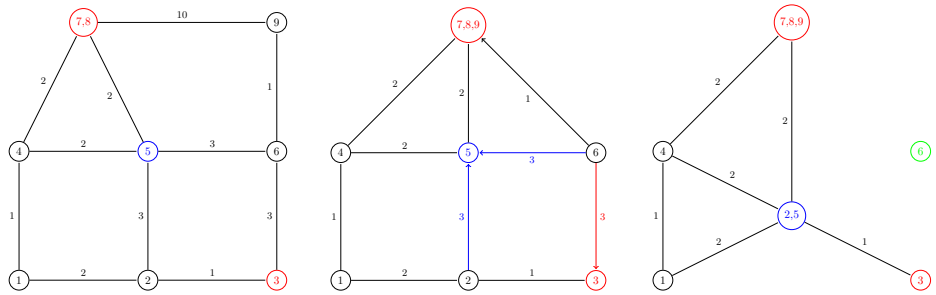

Fig. 9. Consider the toy image from Fig 7. This figure shows the application of PW on RW. **Left**: After first step of the PW sequential optimization of Eq 17, the graph is collapsed to a smaller graph. Vertex 8 is then labelled red. **Middle**: After the second step in PW optimization, the graph collapses further as shown. At this stage, vertex 9 is also labelled red. Using the next highest weights of magnitude 3, random walks are illustrated in respective colours to the labels. **Left**: After solving the RW in the middle figure, the probabilities that vertex 6 is labelled red and blue are computed (probabilities are not shown in the figure). In the next step, the graph is further collapsed and is illustrated here.

transformed into a family of cost minimization problems as given by Eq 17 indexed by p where $p \in \mathbb{N}$.

$$RWCost^{(p)}(\mathbf{x}) = \frac{1}{2} \sum_{e_{ij} \in E} w_{ij}^{p} (x_i - x_j)^2, \ \ subject\ to\ \mathbf{x}_{seed} = \mathbf{f}_{seed}, \tag{17}$$

It was shown in [18] that the results obtained by the limit of minimizers to Eq 17 as $p \to \infty$ are comparable to RW at the benefit of lesser computational cost. Recall Algorithm 1 from Sec 3. The limit of minimizers to Eq 17 as $p \to \infty$ is computed as follows - firstly the sum of terms corresponding to the highest weights in the summation of Eq 17 are considered. This summation is minimized subject to the seed constraints. The solution set is then restricted to those obtained at this level and the sum of terms corresponding to the next highest weight is considered. This summation is minimized subject to seed constraints and the solution set constraint obtained from the minimization problem(s) at the higher weights. The sum of terms corresponding to the next highest weights are then considered. This process is repeated until all the distinct weights are processed.

In case of random walker, this process can be envisioned as follows - consider the subgraph of the image graph with only the highest weights. Examine each of the connected components of this graph. If there is no seed in a connected component, the connected component is collapsed and the target labels of each of the vertices in the connected component are constrained to be same. If there is exactly one seed in a connected component, the connected component is collapsed and the target labels of each of the vertices in the connected component are set as the label of the seed. In case of two or more seeds in a connected component, a random walker is solved on the connected component using Eq 16 as described in Subsec 4.1. The collapsed graph is considered for the sequential processing. The pixels that are already labelled are also treated as seeds for the subsequent steps. Next, the edges of second highest weight are considered and the process is repeated. This process is continued until all the edges of the image graph are processed. Fig 9 illustrates an example of PW approximation to RW. A formal algorithm to solve a random walker segmentation with two labels is given by algorithm 2. For more than two labels, the idea is similar and one needs to work with one-hot encoding to obtain probabilities.

Algorithm 2 PW Approximation to Random Walker Segmentation [18]

Input An edge-weighted graph $\mathcal{G} = (V, E, W)$ and seeds with two labels $x_i = 1$ for $i \in F$ and $x_j = 0$ for $j \in B$, $F, B \subset V$ with $F \cap B = \emptyset$

Output Random walker probabilities for each x_k for $k \in V \setminus (F \cup B)$

Sort the edge set E in decreasing order of weights.
Decompose E into sets of edges with different weights $E = \cup_{i=1}^{l} E_l$ where E_r contains edges with weight w_r for each $1 \leq r \leq l$ with $w_r > w_{r'}$ if $r > r'$.
Denote V_r as the set of vertices that are incident on E_r for each $1 \leq r \leq l$. Set $i = l$.
while $i > 0$ **do**
    Solve Eq 12 on the graph $\mathcal{G}_i = (V_i, E_i, W|_{E_i})$ on each connected component separately
    if Connected component does not contain any vertex with known label probability **then**
        Collapse the vertices of the connected component into a single vertex. The label probability values will be identical on each of these vertices eventually when computed
    else
        The solver yields fixed label probabilities. Consider all the label probabilities known for these vertices
    Set $i = i - 1$
Return label probabilities x_i for each $i \in V$.

Essentially, the application of PW implies that the linear system of equations used for computing random walk probabilities have to be solved on subgraphs of the image graph sequentially. In general, the sizes of the subgraphs obtained at each step are not deterministic in nature. However, typically these subgraphs end up being very small when compared to the image graph. This would significantly reduce the computational cost. Fig 10 from [18] shows a comparison of the time taken to obtain segmentation results on 2D images [44] as a function of number of pixels by random walker algorithm (in green), PW approximation to random walker (in red) along with a few other popular segmentation methods. It can be seen that for images with relatively larger number of pixels, the PW approximation to RW scales well. The downside is that there are no theoretical guarantees on the approximation to the optimal solution.

5 Fast and Robust Isoperimetric Segmentation

Recall from Sec 2 that a graph-cut separates the graph into two components by removing a set of edges. The sum of weights of these edges is minimum among all such sets whose removal results in breaking the graph into two components. Mathematically, it boils down to minimizing the following quantity over all subsets A of the vertex set.

$$W(A, \bar{A}) = \sum_{e_{ij} \in E, i \in A, j \in \bar{A}} w_{ij}, \tag{18}$$

where E denotes the set of edges of the graph and w_{ij} denotes the weight of edge e_{ij}. The summation runs over all edges such that one endpoint of the edge is in A and the other in $\bar{A}$.

Each of these components can be viewed as an object. The sum of edge weights is likely to be less when the edges removed are less in number. In practice, more often than not, a graph-cut results in small components as removal of a few edges is sufficient to separate the graph into a small and a big component (see Fig 11). Small components

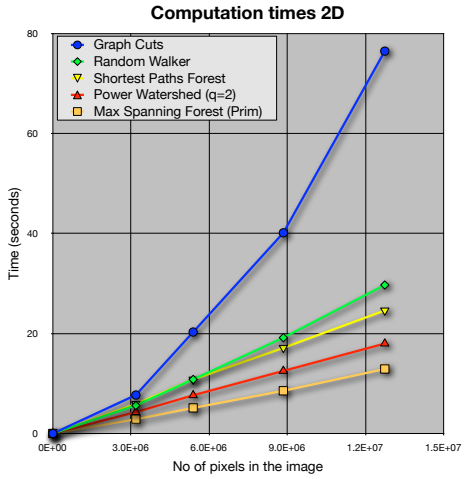

Fig. 10. Comparison of computation times of 2D seeded image segmentation methods on [44] (this figure is replicated from [18]). For each dimension, the times were generated by segmenting the same image scaled down. Power watershed ($q = 2$) corresponds to the power watershed approximation to random walker (in red) and random walker is highlighted in green. Notice that for images with relatively larger number of pixels, the difference in time taken is significant.

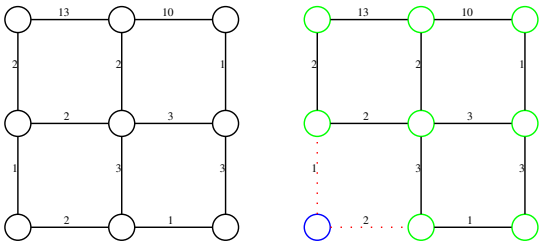

Fig. 11. Left: A similarity-based edge-weighted graph. **Right** Small Cut problem illustrated. Eight vertices in one component and one vertex in the other resulted by graph-cut. The cut edges are highlighted as dotted red edges and the vertices belonging to different components are colour-coded.

are not desired as they practically do not represent any meaningful objects. One of the common approaches to avoid small components is to impose a penalty on the size of the components in the cost function. Ratio Cut [8,53], Normalized cut [53], Isoperimetric cut [29,36,28] are some of the popular methods that impose a penalty on the size. Isoperimetric cut can be described as the following cost minimization problem:

$$IsoCost(A) = \frac{W(A, \bar{A})}{min\{|A|, |\bar{A}|\}}, \tag{19}$$

where A is a subset of the vertices and $\bar{A}$ denotes its complement. The numerator in the cost is the cost function minimized by graph cut (see Eq 18). The denominator penalizes the cost on small components so that minimizing the cost does not yield them. However, this is an NP-hard problem.

5.1 Classic Approach to Isoperimetric Graph Partitioning for Image Segmentation

Notice that the numerator in Eq 19 can be rewritten as

$$W(A,\bar{A}) = \sum_{e_{ij}\in E, i\in A, j\in\bar{A}} w_{ij} = \sum_{e_{ij}\in E} w_{ij}(x_i - x_j)^2,$$

where $x_i = 1$ if $i \in A$ and $x_i = 0$ if $x_i \in \bar{A}$. Using the notion of unnormalized graph Laplacian [53], one can compactly rewrite the cost function in Eq 19 as

$$IsoCost(A) = \frac{\mathbf{x}^T L \mathbf{x}}{min\{\mathbf{x}^T\mathbf{1}, (\mathbf{1}-\mathbf{x})^T\mathbf{1}\}}, \tag{20}$$

where $\mathbf{x}$ is the vector of vertex labels i.e. $x_i = 1$ if $i \in A$ and $x_i = 0$ of $i \in \bar{A}$. L is the unnormalized graph Laplacian.

Typically, in order to obtain an approximation solution to NP-hard problems with a discrete constraint set, the constraints are relaxed. The transformed problem is known as a continuous relaxation of the corresponding NP-hard problem. The solutions to continuous relaxation can be obtained easily. All possible thresholds of such solutions are examined and an optimal threshold i.e. the discretized solution with lowest cost is identified as a heuristic approximation. In practice, such heuristics work well. However, in this particular case, allowing continuous relaxation without further constraints would lead to meaningless solutions. This is because of the cost in Eq 20 is always non-negative (a graph Laplacian is a positive semi-definite [53]). For the relaxed constraints, the cost can be made arbitrarily close to zero (which would be the minimum cost) for every possible partition of the graph (see [22] for details). Hence an additional constraint is added i.e. the label of an arbitrary vertex r is set to be zero without loss of generality. The transformed problem can be written as:

$$Minimize\ \frac{\mathbf{x}_{-r}^T L_{-r} \mathbf{x}_{-r}}{min\{\mathbf{x}_{-r}^T\mathbf{1}, (\mathbf{1}-\mathbf{x}_{-r})^T\mathbf{1}\}}, subject\ to\ each\ x_i \in [0,1] \tag{21}$$

where $\mathbf{x}_{-r}$ denotes the vector $\mathbf{x}$ with the x_r removed and L_{-r} denotes L after deletion of r^{th} row and columns.

Using Lagrange multipliers, solving Eq 21 is equivalent to solving:

$$L_{-r}\mathbf{x}_{-r} = \mathbf{1} \tag{22}$$

The reader may refer to Fig 24 in Sec 9 for an illustration of segmentation on a simulated astronomical image obtained by solving the relaxed isoperimetric partitioning.

5.2 Spanning Tree-Based Approaches to Isoperimetric Graph Partitioning

In [26], it was suggested that solving the continuous relaxation problem on a MaxST of the image graph scales well. To elaborate, recall that the average number of non-zero, non-diagonal elements in each row of a graph Laplacian is the average degree of a vertex in the graph (number of neighbours of a vertex is known as the degree of the vertex in graph theory). For example, from 2D images to 3D images, the sparsity of the graph Laplacian decreases due to an increase in average degree from four to six. Hence, solving the linear system in Eq 22 is slower on 3D images and can be

prohibitively slow for denser graphs. On the other hand, the graph Laplacian of a MaxST always has an average of two non-zero, non-diagonal elements in a row. Also, the results obtained by solving the continuous relaxation on an arbitrary MaxST were shown to be reasonably good in terms of quality for 3D medical image segmentation.

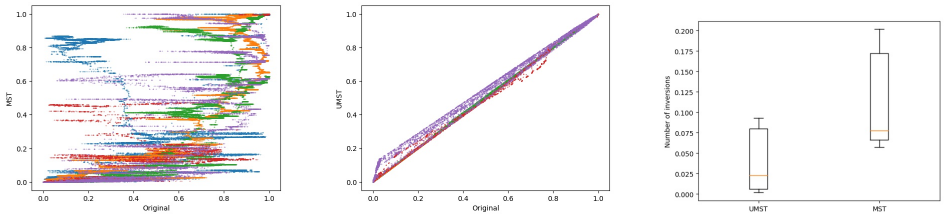

Fig. 12. The relative ordering of the solutions to the continuous relaxation problem of the isoperimetric partitioning (this figure is replicated from [22]). Each colour in the left and middle figures correspond to one image in the Weizmann 1-object dataset [2]. A strictly increasing plot implies perfectly consistent solutions w.r.t. solving Eq 22 on the image graph. In both the figures in left and middle, the X-axis corresponds to the image graph. **Left**: Arbitrary MaxST versus image graph. **Middle**: UMaxST versus image graph. **Right**: A box plot indicating the proportion of inversions on the solutions obtained by an arbitrary MaxST and UMaxST w.r.t. the solutions on image graph.

Later, in [22] it was shown that the solutions obtained by solving Eq 22 on an arbitrary MaxST are not consistent with those solved on the image graph (see Fig 12 replicated from [22]). It was established in [22] that application of PW framework theoretically implies that it is enough to solve Eq 22 on UMaxST of the image graph. Algorithm 3 provides a formal algorithm to obtain the PW approximation to the isoperimetric graph partitioning.

Algorithm 3 PW Approximation to Isoperimetric Graph Partitioning [22]

Input An edge-weighted graph $\mathcal{G} = (V, E, W)$

Output A heuristic approximation to the isoperimetric graph partition i.e. labels $x_i = 0$ or $x_i = 1$ for each $i \in V$.

1: Choose an arbitrary vertex $r \in V$. Set $x_r = 0$
2: Compute the UMaxST of $\mathcal{G}$. Denote it by $\mathcal{G}_{UMaxST}$
3: Compute the unnormalized graph Laplacian L_{UMaxST} of $\mathcal{G}_{UMaxST}$.
4: Delete the r^{th} row and column of L_{UMaxST}. Denote this by $L_{(UMaxST,-r)}$.
5: Denote the vector of labels for each vertex as $\mathbf{x}$. Remove the r^{th} coordinate from $\mathbf{x}$ and denote it by $\mathbf{x}_{-r}$.
6: Solve $L_{(UMaxST,-r)}\mathbf{x}_{-r} = \mathbf{1}$ for $\mathbf{x}_{-r}$ and denote the optimal solution by $\mathbf{x}_{-r}^{opt}$
7: Let $\mathbf{x}^{opt}$ denote the vector with $x_r = 0$ inserted at r^{th} coordinate.
8: Compute all possible thresholds of $\mathbf{x}^{opt}$, check the cost obtained by each such threshold solution using Eq 20.
9: **Return** the solution corresponding to the lowest cost in the previous step.

Further, it was supported by empirical evidence that these solutions are consistent with those solving Eq 22 on the image graph (see Fig 12). Although the number of edges in the UMaxST varies in general and can be as large as that of the image graph itself, it was shown empirically on several 2D image databases that the reduction in a number of edges on an average is about 25 percent. Fig 13 from [22] shows

histograms on the percentage of redundant edges removed i.e. the edges present in the image graph but not in the UMaxST of the image graph. These edges do not contribute to the final segmentation results. However, if they are not discarded, the linear solver in Eq 22 can be prohibitively slow especially when the number of edges are large in the image graph. This is because, the sparser the adjacency matrix of the graph, the more zeros in the corresponding Laplacian.

To summarise, the solution obtained by application of PW to the relaxed isoperimetric partitioning cost minimization is a robust yet fast approximation to the relaxed isoperimetric partitioning cost minimization problem. In particular, the PW approximation is very useful for segmenting astronomical images which usually have very large number of pixels. The downside is that there are no theoretical guarantees on how close the PW approximation solution is from the optimal solution to the relaxed problem. For experiments on simulated astronomical images, the reader may refer to Sec 9.

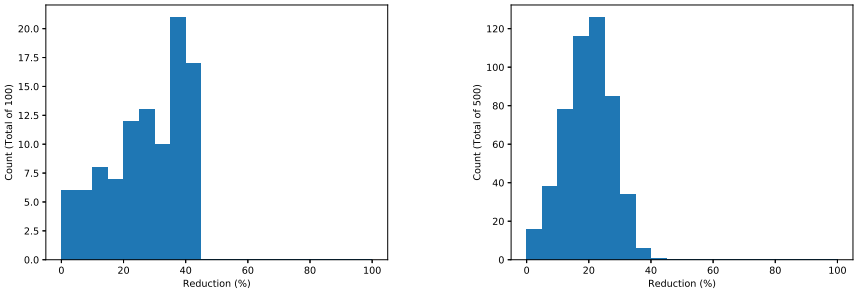

Fig. 13. Histograms indicating the percentage of redundant edges removed by UMaxST. x-axis represents the percentage reduction obtained (this figure is replicated from [22]). y-axis represents the number of images achieving the given amount of reduction. The results are computed on **Left**: Weizmann 2-Object dataset [2], **Right**: on BSDS500 dataset [3].

6 Fast Spectral Clustering

Ratio cut is another variant of graph-cut designed to avoid small components. Suppose we know that an image contains k objects, ratio cut can be described as minimizing the following cost splitting the vertex set V of the image graph into disjoint subsets $A_1, \cdots, A_k$:

$$RatioCost(A_1, \cdots, A_k) = \sum_{t=1}^{k} \frac{W(A_t, \bar{A}_t)}{|A_t|}, \tag{23}$$

6.1 Classic Approach to Ratio Cut

where $W(A, \bar{A})$ is given by Eq 18. However, the ratio-cut problem is NP-hard. In order to obtain an approximation solution, typically a continuous relaxation is constructed. A basic construction is done as follows:

Set $x_i = \sqrt{\frac{|\bar{A}|}{|A|}}$ if $i \in A$ and $x_i = -\sqrt{\frac{|A|}{|\bar{A}|}}$ if $i \in \bar{A}$. These x_is are supposed to be visualized as one-dimensional representations of the vertices. The ratio-cut cost can then be rewritten as

$$RatioCost(A, \bar{A}) = \left(\frac{2}{|A| + |\bar{A}|}\right) \mathbf{x}^T L \mathbf{x}, \tag{24}$$

where L is the unnormalized Laplacian of the image graph. Observe that $\mathbf{x}^T\mathbf{x} = |A| + |\bar{A}| = |V|$ is a constant by the choice of each x_i. The relaxed ratio cut problem can be expressed as:

$$Minimize\ \mathbf{x}^T L \mathbf{x}\ subject\ to\ \mathbf{x}^T\mathbf{x} = |V| \tag{25}$$

This is the classic eigenvector problem i.e. finding an eigenvector with smallest eigenvalue. It is easy to see that the ratio cut cost is always non-negative. Further, a zero cost is attained when $\mathbf{x}$ is a constant vector. However, a constant vector does provide any information on partitioning the vertices as the corresponding x_i values are identical. Hence an additional constraint should be added to avoid constant vectors. As we are working with Eigenspaces, it is enough to search in the orthogonal space to the linear space spanned by constant vectors i.e.:

$$Minimize\ \mathbf{x}^T L \mathbf{x}\ subject\ to\ \mathbf{x}^T\mathbf{x} = |V|,\ and\ \mathbf{x}^T\mathbf{1} = 0 \tag{26}$$

In general, representations with more than one dimension carry more information. As a thumb rule, for partitioning the graph into k components, a k-dimensional representation is used. A continuous relaxation of the ratio cut problem can thus be rewritten as the following minimization problem:

$$Minimize\ Tr(H^T L H)\ subject\ to\ H^T H = I_k, \tag{27}$$

where $Tr()$ denotes the trace operator of a matrix i.e. sum of diagonal elements of a matrix, I_k is an identity matrix of size k, H denotes a $|V| \times k$ matrix with $x_{ti} = \sqrt{\frac{|\bar{A}_t|}{|A_t|}}$ if $i \in A_t$ and $x_{ti} = -\sqrt{\frac{|A_t|}{|\bar{A}_t|}}$ if $i \in \bar{A}_t$. Theoretically, the solution to Eq 27 is obtained by finding the first k eigenvectors of Laplacian L of the image graph.

The row vectors of H obtained by solving Eq 27 are called spectral embeddings. Simple Euclidean distances between spectral embeddings capture the objects reasonably well in an image. Typically, standard algorithms such as k-means [32] are applied to use these distances in order to obtain the final image segmentation (see Fig 14 for an illustration on a general edge-weighted graph). It is worth mentioning that multi-scale combinatorial grouping [4,41] which uses spectral clustering as a building block achieved state-of-the-art on a popular image segmentation dataset when published.

An application of ratio cut on a simulated astronomical image shown in the left image of Fig 8 is illustrated in Fig 15. The ratio cut is performed as follows: firstly, a histogram equalization is performed on the noisy image. This is followed by a median filter. A 4-adjacency similarity graph is constructed on this image. An opening is then performed on the median filtered image. On the opened image, pixels with intensities less than a low preset threshold and pixels with intensities greater than a high preset threshold are identified as two groups of pixels. Each of pixels in the first group are connected to an auxiliary vertex. Similarly, each of the pixels in the second group are connected to a second auxiliary vertex. The weights of the edges incident on either of the auxiliary vertices are set to the same value slightly larger than the highest weight among the edges of the 4-adjacency graph. A ratio cut is then performed on this graph with $k = 2$ clusters. The auxiliary vertices are then discarded and the labels of the other vertices i.e. image pixels are returned.

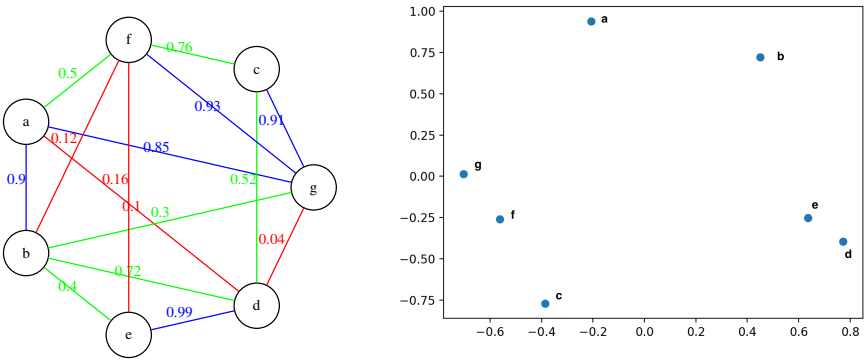

Fig. 14. Left: A similarity-based edge-weighted graph with seven vertices. The edge weights are displayed on the edges and are colour-coded to indicate the strength of similarity. Red, green, and blue indicate low, medium, and high respectively. **Right**: 2 dimensional spectral embedding of the vertices obtained by solving the continuous relaxation of ratio cut. Observe that most of the pairs of vertices with higher similarity are closer and the pairs with low similarity are farther in the embedded space. Hence a simple distance-based clustering would work well on the embeddings for clustering purposes.

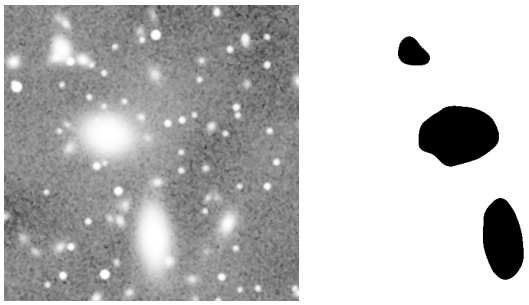

Fig. 15. Left: A simulated astronomical image to be segmented (same as left image in Fig 8). **Right**: Clusters obtained by performing a ratio cut on the image on left. The details of implementation are provided in the text.

6.2 Power Ratio Cut

One of the major issues with spectral clustering is computational complexity. A ratio cut requires $\mathcal{O}(|V|^{\frac{3}{2}})$ computations where V is the set of vertices of the graph. Intuitively, it is expected that pairs of vertices with 'high' similarity should belong to the same component. This can be done by collapsing high similarity vertices together to hybrid vertices thereby reducing the size of the graph (see Fig 16 for an illustration). However, two questions need to be answered: (1) how to quantify a 'high' similarity edge? (2) what are appropriate weights for the edges incident on the hybrid vertices so that spectral clustering on the collapsed graph yields similar results as that of spectral clustering on the original graph? PW framework answers these questions.

Applying the PW framework to Eq 27, one can obtain the following collection of minimization problems labelled with p, where $p \in \mathbb{N}$:

$$Minimize \sum_i w_i^p Tr(H^T L_i H) \; subject \; to \; H^T H = I_k, \tag{28}$$

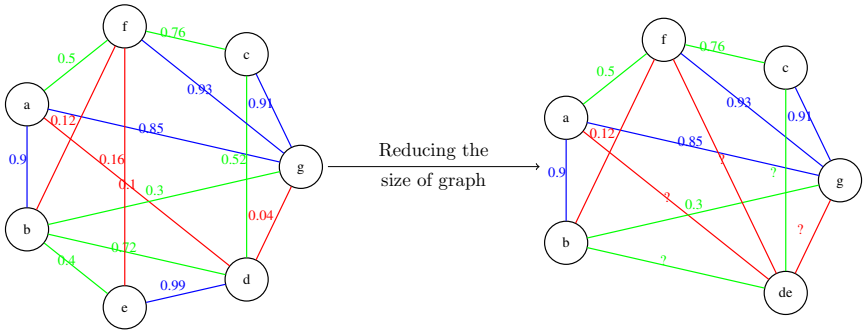

Fig. 16. Left: A similarity-based edge-weighted graph with seven vertices (this figure is replicated from [12]). The edge weights are displayed on the edges and are colour-coded to indicate the strength of similarity. Red, green, and blue indicate low, medium, and high respectively. **Right**: The vertices d and e are merged to form hybrid vertex de. This hybrid vertex is adjacent to every vertex that either of d and e are adjacent. However, it is not clear on how the edge weights for the edges containing hybrid vertices have to be set so that ratio cut on this collapsed graph yields similar results as that of the original graph. PW answers this question.

where the summation is over a distinct set of weights in the graph, L_i is the Laplacian of the subgraph induced by the edges of weight w_i (see Fig 17 for an example) resetting the weights to 1. It was shown in [12] that limit of minimizers to Eq 28 as $p \to \infty$, also known as power ratio cut, can be computed as follows: The edges of the image graph are sorted in decreasing order of weights say $w_l \geq \cdots \geq w_1$. A graph is constructed with edges of highest weights w_l of the image graph $\mathcal{G}$ i.e. the induced subgraph $\mathcal{G}_{\geq w_l}$. Edges are added to this graph gradually in decreasing order of weights. Then a critical value i.e. $1 < r \leq l$ is found such that $\mathcal{G}_{\geq w_r}$, the induced subgraph generated with edges of weights greater than or equal to w_r, has number of connected components greater than or equal to k but $\mathcal{G}_{\geq w_{r-1}}$ has less than k components. A spectral clustering is performed on $\mathcal{G}_{\geq w_{r-1}}$ subject to the condition that the representations on each of the connected components of $\mathcal{G}_{\geq w_r}$ are same. See Algorithm 4 from [12] for a formal algorithm.

Observe that the collapsed graph on which spectral clustering is performed is of much smaller size when compared to the original graph as $k \ll |V|$. Hence, the computation cost for the spectral clustering part is negligible in PW approximation. However, there is a sorting step involved and hence this algorithm runs in $\mathcal{O}(|V|log|V|)$. This is a significant improvement as the classic ratio cut runs in $\mathcal{O}(|V|^{\frac{3}{2}})$. Fig 18 from [12] shows the comparison of time taken to cluster simple blobs dataset [33] with $n_{features} = 2$ and $centers = 2$. Notice that as the number of data points increase, the PW approximation to ratio cut is a better option in terms of computations.

Also, it was shown in [12,10,11] that power ratio cut yields segmentation results comparable to that of the ratio cut. An intuitive explanation on why the approximation is good is that the number of objects are much lesser in an image when compared to the number of pixels in the image. Hence, expensive computations (such as spectral clustering steps) are required only near the object boundaries and a simple greedy algorithm such as a maximum spanning tree based label propagation suffices in the object interiors. In particular, for astronomical image segmentation, power ratio cut is a practical solution to implementing an approximation to the NP-hard ratio cut minimization as these images usually have a large number of pixels.

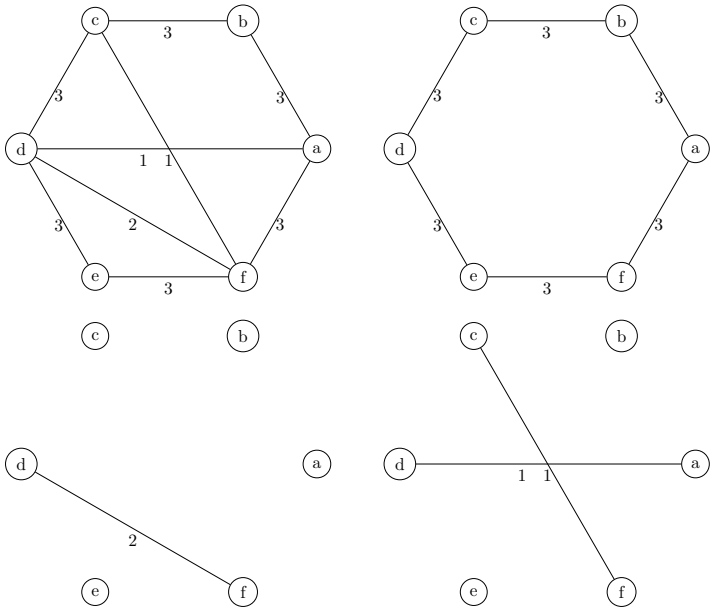

Fig. 17. Top Left: A similarity-based edge-weighted graph with six vertices. The edge weights are displayed on the edges.**Top Right, Bottom Left and Bottom Right**: The edge-weighted graph is decomposed into several subgraphs, each consisting of edge weights of a fixed value in decreasing order of weights. These are called subgraphs induced by edges of fixed weight of the graph on top left.

Algorithm 4 PW Approximation to Ratio Cut Partitioning [12]

Input An edge-weighted graph $\mathcal{G} = (V, E, W)$ with bucketed weights $w_1 < \cdots < w_l$

Output A representation of the subspace spanned by the PW approximation to ratio cut optimal solution.

1: Set $i = l$
2: **while** Number of connected components of $\mathcal{G}_{\geq w_i}$ is greater than or equal to k **do**
3: Set $i = i - 1$.
4: Let $\{C_i\}$, $i \in \{1, \cdots, n_c\}$ be the connected components in $\mathcal{G}_{\geq w_i}$.
5: Let I_{C_i} be the vector

$$I_{C_i}(x) = \begin{cases} 1/\sqrt{|C_i|} & if\ x \in C_i \\ 0 & otherwise \end{cases} \tag{29}$$

6: Construct matrix N with I_{C_i} as column vectors
7: Let $\mathcal{G}_1$ be the graph with vertex set same as that of $\mathcal{G}$. Let L_1 be the unnormalized Laplacian of $\mathcal{G}_1$.
8: Let $\bar{L}_1 = N^t L_1 N$.
9: Calculate the first k eigenvectors of $\bar{L}_1$ and construct A using these eigenvectors as columns.
10: **Return** NA

7 Mutex Watershed and Power Watershed Optimization

In the previous sections, we have seen random walker, spectral clustering and isoperimetric partitioning. Observe that for these methods to work well, a good estimate of the number of objects in the image should be computed beforehand. In the case of random walker, at least one seed pixel should be provided. Isoperimetric partitioning

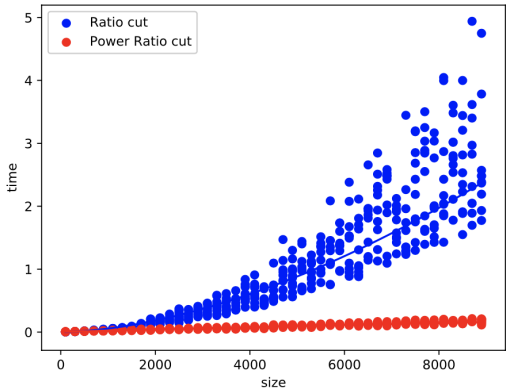

Fig. 18. A comparison of time complexities of PW approximation to ratio cut and ratio cut as a function of data size on blobs dataset [33] with parameters -$n_{features}$ = 2 and *centers* = 2 (this figure is replicated from [12]). Observe that for small data sizes the difference between PW approximation and ratio cut is not significant, while for relatively larger data sizes, the difference is significant.

has to be applied recursively as many times as the number of objects in the image. Spectral clustering uses the number of objects as the input k and partitions the graph into k components.

In practical applications, it is often the case that the number of segments in an image are unknown and difficult to obtain a reasonable estimate. In such cases, one needs to apply algorithms that can yield good quality results which do not require the number of segments as an input. In this section, one such algorithm namely multi-cut problem and its PW approximation [56,54] are reviewed in detail.

7.1 Multi-Cut: NP-Hard Problem

Recall from Sec 2, in some application areas, one can obtain information on certain pairs of pixels that are known to have the same labels and certain pairs to have different labels, each with a varied level of confidence. Hybrid image graphs capturing similarities and dissimilarities between pixels are used for such applications. One can then generalize the notion of graph-cut as follows: *find a set of edges, possibly both similarity and dissimilarity edges (called cut edges) such that sum of their weights is minimum.* Here the segmentation is obtained by discarding the dissimilarity edges after the removal of the cut edges from the image graph. One expects that the end-points of the discarded dissimilarity edges belong to different segments. Hence, there is a requirement of an additional condition on the selected cut edges which is: removal of the cut edges should not result in a cycle with exactly one dissimilarity edge (see Fig 19 for an illustration on violation of this condition).

Observe that the sum of weights of all edges in the image graph is constant. Hence, minimizing the sum of weights of cut edges is the same as minimizing the negative-sum of weights of edges that do not belong to the cut. Denote the set of edges that do not belong to the cut as A, the multi-cut problem can be stated as the following optimization problem:

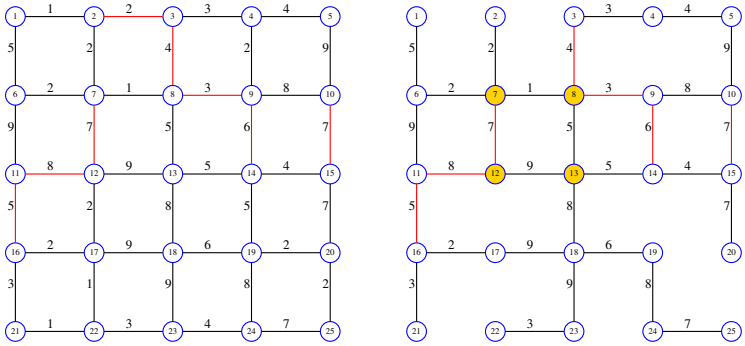

Fig. 19. Left: A hybrid graph on a toy image capturing similarities and dissimilarities between the pixels. Each vertex represents a pixel. Red coloured edges indicate dissimilarity and black coloured edges indicate similarity. The weights on the corresponding edges indicate the strength of similarity/dissimilarity. **Right**: A set of cut edges removed on the image graph on left. Observe that these cut edges violate a desired condition. The cycle $< 7, 8, 13, 12 >$ consisting of edges , $\{7, 8\}$, $\{8, 13\}$, $\{13, 12\}$, $\{12, 7\}$ have exactly one dissimilarity edge namely $\{12, 7\}$. Removal of the edge $\{12, 7\}$ would still lead to pixels 12 and 7 to contain in the same segment.

$$\begin{aligned} &Minimize\ Q(\mathbf{a}) = -\sum_{e \in E} a_e w_e \\ &subject\ to\ \mathbf{a} \in \{0,1\}^{|E|},\ \mathcal{C}_1(A) = \emptyset \text{ with } A = \{e \in E | a_e = 1\} \end{aligned} \tag{30}$$

where E denotes the set of edges of the image graph, $\mathcal{C}_1(A)$ denotes the set of cycles in A with exactly one dissimilarity edge. Unfortunately, Eq 30 is a NP-hard problem.

7.2 Interpreting Mutex Watershed as Power Watershed Approximation to Multi-Cut

In [56, 54], the authors propose an approximation solution to Eq 30 with a greedy algorithm. This algorithm combined with a CNN architecture to learn the edge weights achieved state-of-the-art results on EM segmentation challenge [5] when published. It was shown that this algorithm can be interpreted as an application of PW on the NP-hard multi-cut problem.

$$\begin{aligned} &Minimize\ Q^{(p)}(\mathbf{a}) = -\sum_{e \in E} a_e w_e^p \\ &subject\ to\ \mathbf{a} \in \{0,1\}^{|E|},\ \mathcal{C}_1(A) = \emptyset \text{ with } A = \{e \in E | a_e = 1\} \end{aligned} \tag{31}$$

Decomposing the edges of the graph into different weights in decreasing order of magnitudes $w_l > \cdots > w_1$ (see Fig 17), i.e. $E_r = \{e \in E | w_e = w_r\}$ for each $1 \leq r \leq l$. The first level in the PW nested minimization problem is given by:

$$\begin{aligned} &Minimize\ -\sum_{e \in E_l} a_e \\ &subject\ to\ \mathbf{a} \in \{0,1\}^{|E_l|},\ \mathcal{C}_1(A) = \emptyset \text{ with } A = \{e \in E_l | a_e = 1\} \end{aligned} \tag{32}$$

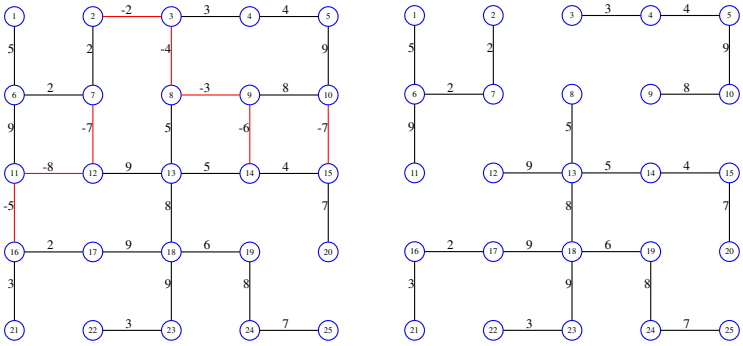

Fig. 20. Left: The set of edges selected by mutex watershed when applied on Fig 19 left image. Observe that there are no cycles with exactly one red edge. Hence there is no ambiguity in the partitions. **Right**: The final partition obtained by mutex watershed. This is obtained by discarding the red edges in the left figure.

Let the solution space be denoted by A_l.

Minimizing Eq 32 is no longer NP-hard as all the weights in the subgraph are of the same weight. One can then greedily add edges so as to satisfy the cycle condition [15]. At the next level, we have the subgraph with edges E_{l-1}:

$$\begin{aligned} &Minimize \; - \sum_{e \in E_{l-1}} a_e \\ &subject \; to \; \mathbf{a} \in \{0,1\}^{|E_{l-1}|}, \mathcal{C}_1(A) = \emptyset \text{ with } A = A_l \cup \{e \in E_{l-1} | a_e = 1\} \end{aligned} \tag{33}$$

Let the solution space be denoted by A_{l-1}. This process is continued all the way until edges are exhausted i.e. E_1. The solution obtained is A_1. Discarding the dissimilarity edges from A_1 yields a segmentation satisfying the cycle constraint. See Algorithm 5 from [54] for a formal algorithm. Here $\mathcal{C}_0(A)$ denotes the set of cycles in A with no dissimilarity edges. The result obtained by applying mutex watershed on Fig 19 left image is illustrated in Fig 20. Fig 21 replicated from [56,54] shows an image from EM segmentation challenge [5] and illustrates how mutex watershed works on a real image.

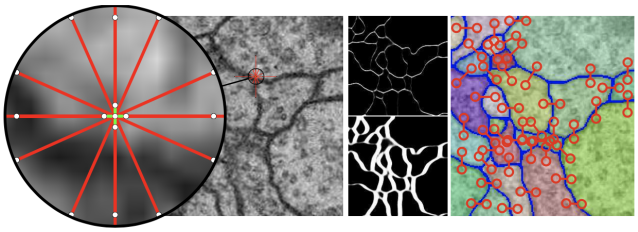

Fig. 21. This figure is replicated from [56,54]. **Left**: Overlay of raw data from the ISBI 2012 EM segmentation challenge and the edges for which attractive (green) or repulsive (red) interactions are estimated for each pixel using a CNN. **Middle**: vertical / horizontal repulsive interactions at intermediate / long range are shown in the top / bottom half. **Right**: Active mutual exclusion (mutex) constraints that the proposed algorithm invokes during the segmentation process.

Algorithm 5 Mutex Watershed [54]

Input An edge-weighted graph $\mathcal{G} = (V, E = E^+ \cup E^-, W = W^+ \cup W^-)$
Output Clusters defined by $A^* \cap E^+$.

Initialization $A = \emptyset$
for $e \in E^+ \cup E^-$ in the descending order of $W^+ \cup W^-$ **do**
    if $\mathcal{C}_0(A \cup \{e\}) = \emptyset$ and $\mathcal{C}_1(A \cup \{e\}) = \emptyset$ **then**
        $A = A \cup \{e\}$
$A^* = A$
Return A^*

To summarise, application of PW to multi-cut allows one to practically implement a multi-cut minimization and obtain high quality image segmentation results. In case of astronomical images, learning appropriate edge weights (using a neural network architecture) suitable for mutex watershed algorithm is a potentially useful research direction.

8 Explaining the Links between Spanning Tree Filters and Shortest Path Filters using PW

Recall from Sec 1 that image filtering is the process of summarizing the image by removing redundant details. The relevant information and the redundant details depend on the application at hand. A popular class of filters are edge-preserving image filters i.e. an operation on the image that blurs the details within objects and preserves the object boundaries. The bilateral filter illustrated in Fig 2 is an example of an edge-preserving filter. There is a vast amount of literature on edge-preserving filters.

In this section, we restrict the discussion to two families of graph-based weighted average filters namely shortest path filters and spanning tree based filters. As the name suggests, graph-based filters implies that the filtering is performed using a graph model on the image. The filtered value at each pixel is given by a weighted average of the other pixels in the image. The pairwise weights in the shortest path filters arise from a shortest path between pairs of pixels. On the other hand, the pairwise weights in the spanning tree-based filters are computed from spanning trees.

8.1 Shortest Path Filters

Recall from Sec 2 that dissimilarity based edge-weighted graphs are used for image filtering. The family of the shortest path filters can be described as follows. The filtered value at pixel i is given by:

$$SPF_i = \sum_{j \in V} g_i(j) I_j \ , \tag{34}$$

where $g_i(j) = \frac{exp\left(-\frac{\Theta(i,j)}{\sigma}\right)}{\sum_{q \in V} exp\left(-\frac{\Theta(i,q)}{\sigma}\right)}$. $\Theta(i,j)$ is the smallest number of edges on paths among all shortest path between pixels i and j on the image graph. σ is a smoothing parameter. Shortest paths can be defined in many ways [35,24,13]. Fig 22 illustrates a standard definition of a shortest path distance. Intuitively, pixels which are in the same object are separated by shorter paths when compared to pixels across objects.

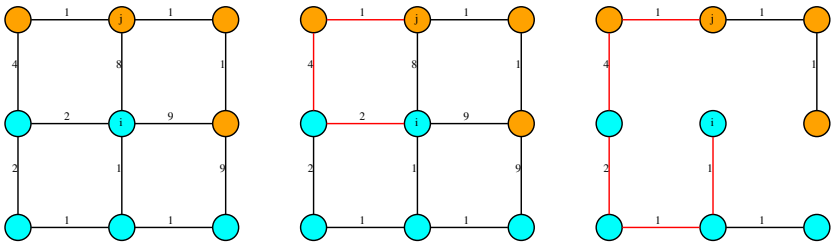

Fig. 22. Left: A toy image represented with an edge-weighted graph. Vertices represent pixels and the edge weights represent dissimilarity between adjacent vertices. The image contains two objects and the pixels corresponding to different objects are highlighted in different colours. **Middle**: The shortest path between the pixels i and j is highlighted in red. Here, a shortest path is determined by the sum of the weights of its edges. Observe that every other path between i and j has sum of the edge weights strictly greater than that of the path highlighted in red. Thus $\Theta(i,j) = 3$. **Right**: The unique path between the pixels i and j is highlighted in red i.e. $D(i,j) = 5$. Notice that for many pairs of pixels across objects i.e. pairs such that both pixels belonging to different objects, the separation on the MinST is larger than that of their spatial distance.

Thus, it is expected that such a filter would result in smoothing of the image keeping the object boundaries intact.

Although, shortest path filters are theoretically promising, they are computationally expensive. This is because an exact computation [25] requires finding shortest paths between all pairs of vertices in the graph which is $\mathcal{O}(|V|^3)$ where $|V|$ is the number of vertices of the image graph.

8.2 Spanning Tree Filters

On the other hand, spanning tree based filters were introduced independently w.r.t. shortest path filters. The spanning tree filters are weighted average filters. The pairwise similarity weights are computed on an arbitrary minimum spanning tree (MinST) on the image graph. A specific spanning tree filter namely Tree filter [6] can be described as follows. The filtered value at pixel i is given by:

$$\mathrm{TF}_i = \sum_j t_i(j) I_j, \tag{35}$$

where $t_i(j) = \frac{\exp(-\frac{D(i,j)}{\sigma})}{\sum_q \exp(-\frac{D(i,q)}{\sigma})}$. $D(i,j)$ is the number of edges on an arbitrary MinST of the graph (this quantity is well-defined as given an arbitrary spanning tree, there exists a unique path between every pair of vertices in the graph). σ is a smoothing parameter. An illustration of the spanning tree filter on a toy example is provided in Fig 22. Fig 23 provides an illustration of the tree filter on a simulated astronomical image. In general, for pairs of pixels i and j belonging to different objects, $D(i,j)$ is large. However, there is at least one boundary edge in any arbitrary MinST. In practice such edges are usually negligible in number. Thus, tree filter works reasonably well in practice except for a small leak at object boundaries. A tree filter can be computed efficiently in linear time [58] (using two passes, bottom-up and top-down).

In [21,23], it was shown using PW framework that the tree filter is a fast approximation to the contrast-invariance version of a shortest path filter. The exact contrast invariant version of a shortest path filter was characterized as a weighted average filter with the pairwise weights computed on UMinST of the image graph. Recall that

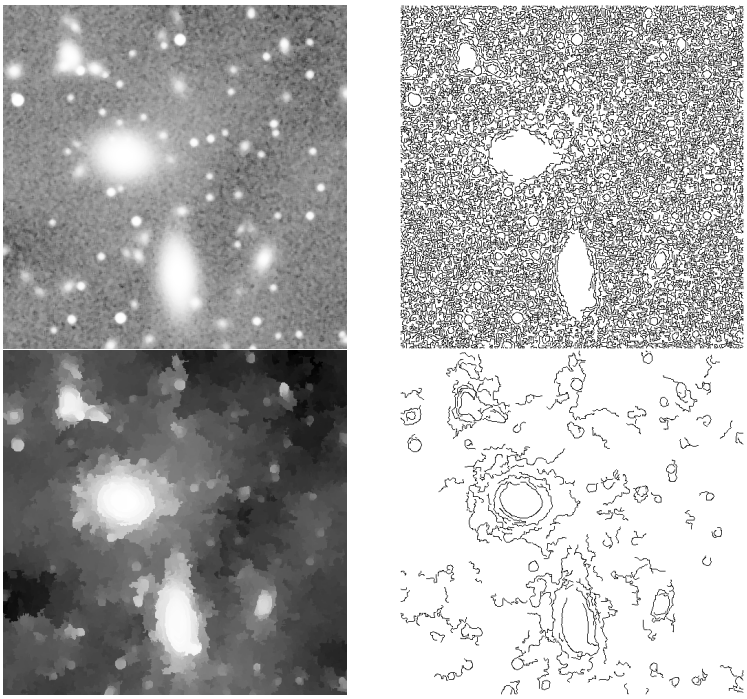

Fig. 23. Top Left: A simulated astronomical image to be segmented (same as left image in Fig 8). **Top Right**: A Canny edge detector applied on the image on top left. **Bottom Left**: A tree filter is implemented on the image on left using Eq 35. **Bottom Right**: A Canny edge detector applied on the tree filtered image. The boundaries identified by a simple edge detector such as Canny edge detector are more reliable on the tree filtered image.

UMinST of a graph is a subgraph generated by edges of all MinST's of the graph. Two other approximations to obtain the contrast invariant version of a shortest path filter, namely order-based approximation and depth-based approximation have been proposed in [23]. Algorithms 6, 7, and 8 provide details on implementation of these approximations. The key idea to implementing these approximations is that for every pixel in the image, there exists a spanning tree from which the pairwise weights for filtering this pixel can be computed. Such a tree is termed as an adaptive spanning tree in [23]. These adaptive spanning trees can be computed in parallel to obtain these approximations more efficiently. Further, it was shown empirically that the tree filter and these approximations yield similar results in practice.

Algorithm 6 Generic Algorithm to compute UMinST Filter [23]

Input A 4-adjacency graph $\mathcal{G} = (V, E, W)$ of an image I, Adaptive Spanning Trees T_i for each $i \in V$, smoothing parameter σ

Output Filtered image S.

1: **for** $i \in V$ **do**

2: Starting from i on T_i, use $S_p = I_p + \sum_{q \in \text{children of p}} exp(\frac{-1}{\sigma}) S_q$ recursively to compute S_i

3: **Return** S

Algorithm 7 Depth-truncated Adaptive Spanning Tree [23]

Input UMinST of the graph $\mathcal{G} = (V, E, W)$, depth d and pixel i
Output Depth-Truncated Adaptive Spanning Tree $T_{i,d}$.

Set $X = \{i\}$ and $T_{i,d} = (i, \emptyset)$
while True **do**
  break = True
  for e in shortest edges from X to X^c **do**
    if $dist(e, i, T_{i,d}) < d$ **then**
      Add e to the edge-set of $T_{i,d}$
      break = False
  if break = True **then**
    Return $T_{i,d}$.

Algorithm 8 Order-truncated Adaptive Spanning Tree [23]

Input UMinST of the graph $\mathcal{G} = (V, E, W)$, kernel size N and pixel i, path cost function f determined by reverse lexicographic ordering of the edges in the path [23]
Output Order-Truncated Adaptive Spanning Tree $\hat{T}_{i,N}$.

Set $\hat{T}_{i,N} = \emptyset$, $\mathcal{Q} = I$, $Parent(j) = null$ for each $j \in I$ and $count = 0$
while $\mathcal{Q} \neq \emptyset$ and $count < N$ **do**
  Remove from $\mathcal{Q}$ a pixel j such that $f(P^*(j))$ is minimum and add it to $\hat{T}_{i,N}$
  $count+ = 1$
  for each pixel k such that $(j, k) \in E$ **do**
    if $f(P^*(j) \cdot < j, k >) < f(P^*(k))$ **then**
      set $Parent(k) = j$
Return $\hat{T}_{i,N}$

9 Experiments on Simulated Astronomical Sky Images

In this section, experiments are performed on simulated astronomical sky images. It is a common practice to use simulations [47,52,40,31] as it is difficult to obtain ground truth segmentation for real astronomical images. The sky images are generated using the R code developed by authors in [43]. Firstly, noise-free sky images with light sources are generated. A threshold is then applied on the noise-free simulations to obtain a ground truth foreground and background. Typical noise such as Viking object shot-noise, sky noise and Pareto noise are then added to simulate real sky images. See Fig 24 for an illustration. The noisy images are used for segmentation and filtering.

The goal of segmentation is to separate the foreground from the background i.e. separate the background sky from the light sources. Recall that one of the the aims of the article is to demonstrate that the implementation of classic graph-based cost minimization methods and the corresponding PW versions yield similar results. F-ratio and AUC curves are used evaluating segmentation and filtering results. As the tutorial article does not attempt to achieve state-of-the-art results, sophisticated measures such as pairs-of-pixels method [42] or tailor-made evaluation measures for astronomical sky images such as described in [31] are not used. The second aim of the article is to show that the PW versions scale well when compared to the classic implementations. This is demonstrated experimentally by implementing both the algorithms on a same machine (Intel(R) Xeon(R) CPU E5620 at 2.40GHz with RAM size of 16 GigaBytes) and comparing the computation times. All the segmentation methods described in the article are similar to implement. Hence, segmentation experiments are performed comparing only one segmentation method namely isoperimetric partitioning i.e. the classic implementation of isoperimetric partitioning versus the corresponding PW

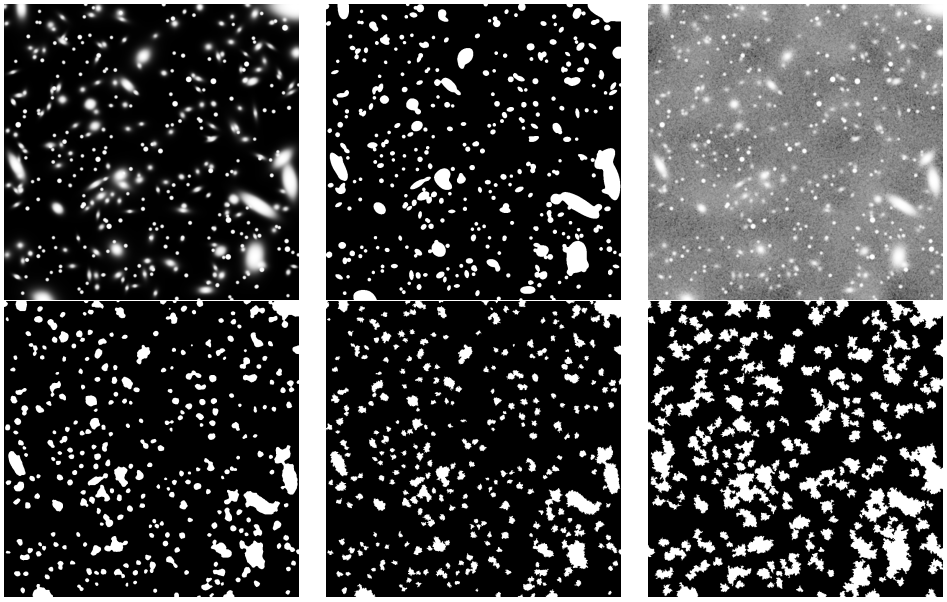

Fig. 24. Top Left: A simulation of a noise-free astronomical sky image. **Top Middle**: A ground truth of foreground and background obtained by thresholding the image on left. **Top Right**: Viking object shot-noise, sky noise and Pareto noise added to the noise-free image to simulate a real astronomical sky image [43]. A median filtered version of the noisy image is illustrated for better visualization. **Bottom:** All three segmentations are obtained on the raw noisy image as described in the text. **Bottom Left**: Result of segmentation obtained by implementation of classic approach to isoperimetric segmentation (see Eq 21). **Bottom Middle**: PW implementation of isoperimetric segmentation (i.e. solving Eq 22 on UMaxST) yields similar results to that of the classic approach. **Bottom Right**: Result of solving Eq 22 on an arbitrary MaxST i.e. the method in [26]. Observe that qualitatively these results appear different when compared to the figure on bottom left. This observation is consistent with left figure in Fig 12.

version. Also, these segmentation methods yield hard labels. Hence, F-ratio is used to evaluate the quality of the segmentation. The higher the F-ratio, the better the segmentation results.

The isoperimetric partitioning is performed as follows: the noisy images are pre-processed with a threshold operation followed by a morphological opening [45]. The vertices corresponding to the bright pixels in the pre-processed image are identified as the foreground pixels. A 4-adjacency similarity graph is constructed on the image. Each of the vertices corresponding to the identified foreground pixels in the pre-processing step are additionally connected to an auxiliary vertex. The weights of the edges incident on the auxiliary vertex are set to a value slightly larger than the highest weight among the edges of the 4-adjacency graph. Isoperimetric partitioning is recursively performed on this graph. The auxiliary vertex is then discarded and the labels of the other vertices i.e. image pixels are returned. This yields an over-segmentation on some of the images i.e. more than two labels in the image. Some of the labels are merged together such that the merging results in only two labels. Table 1 contains a comparison of the F-ratios obtained on 30 simulated images of sizes 1000×1000 for classic implementation of isoperimetric segmentation, PW counterparts, and on an arbitrary MaxST as proposed in [26]. It can be observed from the table that the

classic implementation and PW implementation yields a similar F1 ratio while the MaxST implementation yields a lower F1 ratio.

Table 1. F-Ratio

	Mean F1-ratio on 30 simulated images		
	Iso	**PW Iso**	**MST Iso**
Mean F-ratio	0.74	0.75	0.68
Std. Dev. F-ratio	0.07	0.06	0.08

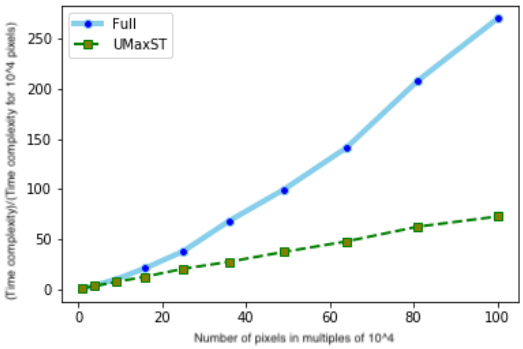

Fig. 25. X-axis represents the number of pixels (in multiples of 10^4) in the image on which segmentation is performed. A point (x_0, y_0) has to be interpreted as follows: y_0 represents the ratio of mean time taken for implementation of a segmentation method on an image with $x_0 \times 10^4$ pixels to the mean time taken to implement the same method on an image with 10^4 pixels. The blue plot corresponds to the classic implementation of isoperimetric partitioning and the green plot corresponds to the PW version. It can be clearly seen that the PW version scales much better when compared to the classic implementation.

Fig 25 compares the scalability of the PW-based isoperimetric segmentation with the classic implementation of isoperimetric segmentation. Simulated astronomical images of 10 different sizes and 30 images of each size, starting from a smallest size of 100×100 to a largest size of 1000×1000 are used. The l^{th} size has width and height each of $100l$ i.e. its number of pixels are l^2 times the number of pixels in the smallest size. The ratio of the mean computational time for each image size to the mean computational time of the smallest size are plotted on the y-axis. The values on the x-axis have to be interpreted as the ratio of number of pixels in the image to the number of pixels in a 100×100 image. For example, a value of 81 on the x-axis indicates that the image is of the shape 900×900 and has 81×10^4 pixels in total. It can be seen that PW counterpart of the isoperimetric segmentation scales well (the green plot) while the classic version (the blue plot) is not scalable.

For comparing the filtering methods, a bilateral filter is used to compare against the tree filter instead of a shortest path filter. This is because a bilateral filter is a widely used as an edge-preserving filter in practice. Also, shortest path filters are prohibitively slow to implement. The filters are compared as follows: a noisy image is filtered using both the filtering methods. As these filters are supposed to smooth pixel values within objects without losing the boundary details, simple thresholds on the

filtered images are expected to achieve good segmentation results. The foreground-background segmentation problem as described earlier is used for this purpose. The filtered images are post-processed with an opening operator [46] and each threshold, the segmentation result is compared against the ground truth w.r.t. AUC. An average AUC measure over each threshold is used as a quality measure of the segmentation. A higher average AUC indicates a better quality segmentation and thus a better filtering approach. It can be seen from Table 2 that tree filter outperforms the classic bilateral filter used for edge-preserving filtering.

Table 2. AUC

	Mean AUC on 30 simulated images	
	BF	**TF**
Mean	0.61	0.79
Std. Dev.	0.03	0.03

A tree filter, an approximation to the contrast invariance implementation of shortest path filter thus yields good filtering results. It is easy to see that the bottleneck of the tree filter is the computation of a MinST. This is because the filtered values are obtained on the tree in two passes and can be obtained in linear time. On the other hand, a MinST computation required a sorting step which is $\mathcal{O}(|V|log|V|)$. Thus tree filter scales better than a shortest path filter which has asymptotic complexity of $\mathcal{O}(|V|^3)$.

10 Conclusions and Perspectives

In this article, several popular graph-based approaches to image segmentation and filtering namely random walker segmentation, isoperimetric partitioning, ratio cut, multi-cut and shortest path edge-preserving filters are revisited. The applications of PW framework to these methods are surveyed and analysed from the perspective of contrast invariance. It is shown that the PW versions of these methods can be visualized as contrast-agnostic fast approximations to the corresponding methods. These methods fall under a large class of cost minimization problems on finite graphs. This class encompasses all cost functions that can be written as a weighted linear combination of pairwise penalties on pixel labels such that the weights are monotonic functions of the corresponding weights of the edges connecting the pixels. On this particular class of cost minimization problems, the PW framework operates on a specific substructure of the graph i.e. either of UMinST or UMaxST. For graph-based cost minimization approaches to image processing with more general cost functions, the PW framework can still be applied by considering the sequence of nested minimization problems as mentioned in the article. This results in scalable algorithms and is potentially useful in image processing applications such as astronomy where images with massive number of pixels needs to be processed.

A recent trend in the usage of graph-based optimization in imaging applications is to learn the edge weights of the underlying image graph using deep neural networks. For example, an end-to-end learned random walker proposed in [9] achieved state-of-the-art results on some image segmentation database [20]. However, these methods do not scale well. PW is compatible with such end-to-end learned graph-based methods and can be applied at the test phase. This would be useful in building scalable state-of-the-art models for image segmentation and filtering. As a current instance of such

research direction, the mutex watershed [56,54,55] was shown to achieve state-of-the-art results on a popular image segmentation database [5] and high quality results on popular semantic segmentation databases [14,38], using edge-weights obtained thanks to a CNN architecture. In case of astronomical images, learning appropriate edge weights (using a neural network architecture) suitable for mutex watershed algorithm is a potentially useful research direction. Essentially, domain knowledge of the physical characteristics of astronomical images can be incorporated into the weights of edges. PW has also been used for other interesting applications such as surface reconstruction [16] and estimation of separating planes between touching 3D objects [34]. Anisotropic diffusion for L0 [17] is another interesting direction of research.

Acknowledgments

Sravan Danda would like to acknowledge the funding received from BPGC/RIG/2020-21/11-2020/01 (Research Initiation Grant provided by BITS-Pilani K K Birla Goa Campus) and thank APPCAIR, and Computer Science and Information Systems, BITS-Pilani Goa. Aditya Challa would like to thank Indian Institute of Science (IISc) for the Raman Post Doctoral fellowship. The work of B. S. D. Sagar was supported by the DST-ITPAR-Phase-IV project under the Grant number INT/Italy/ITPAR-IV/Telecommunication/2018. Laurent Najman would like to acknowledge the funding received from - ANR-15-CE40-0006 CoMeDiC, ANR- 14-CE27-0001 GRAPHSIP research grants and Programme d'Investissements d'Avenir (LabEx BEZOUT ANR-10-LABX- 58). He would also like to thank the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 721463 to the SUNDIAL ITN network.

References

1. Cédric Allène, Jean-Yves Audibert, Michel Couprie, Jean Cousty, Renaud Keriven, et al. Some links between min-cuts, optimal spanning forests and watersheds. In *ISMM (1)*, pages 253–264, 2007.
2. "Sharon Alpert, Meirav Galun, Ronen Basri, and Achi Brandt". Image segmentation by probabilistic bottom-up aggregation and cue integration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.
3. Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.
4. Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014.
5. Ignacio Arganda-Carreras, Srinivas C Turaga, Daniel R Berger, Dan Cireşan, Alessandro Giusti, Luca M Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9:142, 2015.
6. Linchao Bao, Yibing Song, Qingxiong Yang, Hao Yuan, and Gang Wang. Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. *IEEE TIP*, 23(2):555–569, 2014.
7. Yuri Boykov and Olga Veksler. Graph cuts in vision and graphics: Theories and applications. In *Handbook of mathematical models in computer vision*, pages 79–96. Springer, 2006.
8. Wallace Casaca, Afonso Paiva, Erick Gomez-Nieto, Paulo Joia, and Luis Gustavo Nonato. Spectral image segmentation using image decomposition and inner product-based metric. *Journal of Mathematical Imaging and Vision*, 45(3):227–238, Mar 2013.

9. Lorenzo Cerrone, Alexander Zeilmann, and Fred A Hamprecht. End-to-end learned random walker for seeded image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12559–12568, 2019.
10. Aditya Challa, Sravan Danda, B. S. Daya Sagar, and Laurent Najman. An introduction to Gamma-convergence for spectral clustering. In *Discrete Geometry for Computer Imagery - 20th IAPR International Conference, DGCI 2017, Vienna, Austria, September 19-21, 2017, Proceedings*, pages 185–196, 2017.
11. Aditya Challa, Sravan Danda, B. S. Daya Sagar, and Laurent Najman. Power spectral clustering on hyperspectral data. In *2017 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2017, Fort Worth, TX, USA, July 23-28, 2017*, pages 2195–2198, 2017.
12. Aditya Challa, Sravan Danda, BS Daya Sagar, and Laurent Najman. Power spectral clustering. *Journal of Mathematical Imaging and Vision*, 62(9):1195–1213, 2020.
13. Krzysztof Chris Ciesielski, Alexandre Xavier Falcão, and Paulo AV Miranda. Path-value functions for which dijkstra's algorithm returns optimal mapping. *Journal of Mathematical Imaging and Vision*, 60(7):1025–1036, 2018.
14. Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
15. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
16. Camille Couprie, Xavier Bresson, Laurent Najman, Hugues Talbot, and Leo Grady. Surface reconstruction using power watershed. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 381–392. Springer, 2011.
17. Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Anisotropic diffusion using power watersheds. In *2010 IEEE International Conference on Image Processing*, pages 4153–4156. IEEE, 2010.
18. Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(7):1384–1399, 2011.
19. Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(8):1362–1374, 2009.
20. CREMI. Miccai challenge on circuit reconstruction fromelectron microscopy images. 2017.
21. Sravan Danda, Aditya Challa, B. S. Daya Sagar, and Laurent Najman. Power tree filter: A theoretical framework linking shortest path filters and minimum spanning tree filters. In *Mathematical Morphology and Its Applications to Signal and Image Processing - 13th International Symposium, ISMM 2017, Fontainebleau, France, May 15-17, 2017, Proceedings*, pages 199–210, 2017.
22. Sravan Danda, Aditya Challa, BS Daya Sagar, and Laurent Najman. Revisiting the isoperimetric graph partitioning problem. *IEEE Access*, 7:50636–50649, 2019.
23. Sravan Danda, Aditya Challa, BS Daya Sagar, and Laurent Najman. Some theoretical links between shortest path filters and minimum spanning tree filters. *Journal of Mathematical Imaging and Vision*, 61(6):745–762, 2019.
24. Alexandre X Falcao, Jorge Stolfi, and Roberto de Alencar Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE PAMI*, 26(1):19, 2004.
25. Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.
26. Leo Grady. Fast, quality, segmentation of large volumes - isoperimetric distance trees. In *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part III*, pages 449–462, 2006.
27. Leo Grady. Random walks for image segmentation. *IEEE PAMI*, 28(11):1768–1783, 2006.

28. Leo Grady and Eric L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(3):469–475, 2006.
29. Leo Grady and Eric L. Schwartz. Isoperimetric partitioning: A new algorithm for graph partitioning. *SIAM J. Scientific Computing*, 27(6):1844–1866, 2006.
30. Jacopo Grazzini and Pierre Soille. Edge-preserving smoothing using a similarity measure in adaptive geodesic neighbourhoods. *Pattern Recognition*, 42(10):2306–2316, 2009.
31. Caroline Haigh, Nushkia Chamba, Aku Venhola, Reynier Peletier, Lars Doorenbos, Matthew Watkins, and Michael HF Wilkinson. Optimising and comparing source extraction tools using objective segmentation quality criteria. *arXiv preprint arXiv:2009.07586*, 2020.
32. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.
33. http://scikit-learn.org/stable/modules/ generated/sklearn.datasets.make blobs.html. scikit-learn datasets. *accessed: 2017-12- 12.*
34. Clara Jaquet, Edward Andó, Gioacchino Viggiani, and Hugues Talbot. Estimation of separating planes between touching 3d objects using power watershed. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 452–463. Springer, 2013.
35. Romain Lerallut, Étienne Decencière, and Fernand Meyer. Image filtering using morphological amoebas. *Image and Vision Computing*, 25(4):395–404, 2007.
36. B. Mohar. Isoperimetric numbers of graphs. *J. Comb. Theory Ser. B*, 47(3):274–291, December 1989.
37. Jean-Michel Morel and Sergio Solimini. *Variational methods in image segmentation: with seven image processing experiments*, volume 14. Springer Science & Business Media, 2012.
38. Jacob M Musser, Klaske J Schippers, Michael Nickel, Giulia Mizzon, Andrea B Kohn, Constantin Pape, Jörg U Hammel, Florian Wolf, Cong Liang, Ana Hernández-Plaza, et al. Profiling cellular diversity in sponges informs animal cell type and nervous system evolution. *BioRxiv*, page 758276, 2019.
39. Laurent Najman. Extending the PowerWatershed framework thanks to Γ-convergence. *SIAM Journal on Imaging Sciences*, 10(4):2275–2292, November 2017.
40. Thanh Xuan Nguyen, Giovanni Chierchia, Laurent Najman, Aku Venhola, Caroline Haigh, Reynier Peletier, Michael HF Wilkinson, Hugues Talbot, and Benjamin Perret. Cgo: Multiband astronomical source detection with component-graphs. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 16–20. IEEE, 2020.
41. Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):128–140, 2016.
42. Jordi Pont-Tuset and Ferran Marqués. Supervised evaluation of image segmentation and object proposal techniques. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(7):1465–1478, 2016.
43. ASG Robotham, LJM Davies, SP Driver, S Koushan, DS Taranu, S Casura, and J Liske. Profound: source extraction and application to modern survey data. *Monthly Notices of the Royal Astronomical Society*, 476(3):3137–3159, 2018.
44. Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.
45. Jean Serra. Mathematical morphology, vol. i. *London, UK: Academic*, 1982.
46. Jean Serra et al. Mathematical morphology for boolean lattices. *Image Analysis and Mathematical Morphology, II: Theoretical Advances*, pages 37–58, 1988.
47. JL Sérsic. Observatorio astronomico. *Cordoba, Argentina*, 1968.
48. Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 731–737. IEEE, 1997.

49. Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
50. Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision, 1998. ICCV 1998*, pages 839–846. IEEE, 1998.
51. Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.
52. Aku Venhola. Evolution of dwarf galaxies in the fornax cluster. *PhD Thesis*, http://hdl.handle.net/11370/1bcc02c2-2c78-4cff-b801-147c31b000a8, 2019.
53. Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.
54. Steffen Wolf, Alberto Bailoni, Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A Hamprecht. The mutex watershed and its objective: Efficient, parameter-free graph partitioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
55. Steffen Wolf, Yuyan Li, Constantin Pape, Alberto Bailoni, Anna Kreshuk, and Fred A Hamprecht. The semantic mutex watershed for efficient bottom-up semantic instance segmentation. In *European Conference on Computer Vision*, pages 208–224. Springer, 2020.
56. Steffen Wolf, Constantin Pape, Alberto Bailoni, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A. Hamprecht. The mutex watershed: Efficient, parameter-free image partitioning. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, volume 11208 of *Lecture Notes in Computer Science*, pages 571–587. Springer, 2018.
57. Lijuan Xu, Fan Wang, Laura Dempere-Marco, Qi Wang, Yan Yang, and Xiaopeng Hu. Path-based analysis for structure-preserving ima ge filtering. *Journal of Mathematical Imaging and Vision*, 62(2):253–271, 2020.
58. Qingxiong Yang. Stereo matching using tree filtering. *IEEE PAMI*, 37(4):834–846, 2015.