Graph Partitioning for Image Segmentation using Iso-Perimetric Algorithm

[1]PROF. ANIKET V. GOKHALE* [2]Er. AMIT S. DIKHOLKAR*
Department of Electronics Engineering
**Yeshwantrao Chavan College of Engineering,*
Nagpur INDIA.
amit.dikholkar@gmail.com

Abstract: **- Graph cuts proved to be a useful multidimensional optimization tool which can enforce piecewise smoothness while preserving relevant sharp discontinuities. This paper is mainly intended as an application of isoperimetric algorithm of graph theory for image segmentation and analysis of different parameters used in the algorithm like generating weights, regulates the execution, Connectivity Parameter, cutoff, number of recursions,. We present some basic background information on graph cuts and discuss major theoretical results, which helped to reveal both strengths and limitations of this surprisingly versatile combinatorial algorithm.**

***Keywords*: - graph theory, image partitioning, Isoperimetric algorithm, connectivity.**

*****

1 Graph Theory for image partitioning

Graph processing algorithms have become increasingly popular in the context of computer vision. Typically, pixels are associated with the nodes of a graph and edges are derived from a 4- or 8-connected lattice topology. Some authors have also chosen to associate higher level features with nodes. For purposes of importing images to spacevariant architectures, we adopt the conventional view that each node corresponds to a pixel. Graph theoretic algorithms often translate naturally to the proposed space-variant architecture. Unfortunately, algorithms that employ convolution (or correlation) implicitly assume a shift invariant topology. Although shift-invariance may be the natural topology for a lattice, a locally connected space-variant sensor array (e.g., obtained by connecting to K-nearest-neighbors) will typically result in a shift-variant topology. Therefore, a reconstruction of computer vision algorithms for space-variant architectures requires the use of additional theory to generalize these algorithms.

2: Iso-Perimetric Algorithm

2.1 Iso-perimetric Problem

Graph partitioning has been strongly influenced by properties of a combinatorial formulation of the classic isoperimetric problem: For a fixed area, find the region with minimum perimeter.
Defining the **isoperimetric constant I** of a manifold as:

$$I = \frac{\inf(|\partial S|)}{V} \text{ ---------(1)}$$

Where S is a region in the manifold, *V* denotes the volume of region S, |¶S| is the area of the boundary of region S, and I is the infimum of the ratio over all possible S. For a compact manifold volume of region S is less than or equal to 50 % of total volume and for a non-compact manifold, volume of the region $< \infty$
We show in this paper that the set (and its complement) for which ***I*** takes a minimum value defines a good heuristic for data clustering and image segmentation. In other words, finding a region of an image that is simultaneously both large (i.e., high volume) and that shares a small perimeter with its surroundings (i.e., small boundary) is intuitively appealing as a good image segment. Therefore, we will proceed by defining the isoperimetric constant on a graph, proposing a new algorithm for approaching the sets that minimize I, and demonstrate application to data clustering and image processing.

2.2 The Isoperimetric Partitioning Algorithm

A Graph is a pair G=(V,E) with nodes v ϵ V and edges e ϵ E⊆V x V. An edge,e, spanning two vertices, Vi and vj, is denoted by e_{ij}. A weighted graph has a non negative and real value assigned to each edge called a weight, wij, since weighted graphs are more general than unweighted graphs; we have developed our results for weighted graphs. The degree d_i of vertex v_i is

$$d_i = \sum_{e_{ij}} w(e_{ij}) \text{ ----------(2)}$$

for e_{ij} ϵ E
For a graph, G, the isoperimetric constant is defined as given in equation (1). In graphs with a finite node

set, the infimum in (1) becomes a minimum. Since we will be computing only with finite graphs, we will
henceforth use a minimum in place of an infimum. The boundary of a set **S** is defined as

$$S = \{e_{ij} \mid i \in S \mid; j \in S\},$$

where **S** denotes the set complement, and

$$|\partial S| = \sum_{e_{ij} \in \partial s} W(e_{ij}) \text{---------- (3)}$$

For a given set, **S**, we term the ratio of its boundary to its volume the isoperimetric ratio, denoted by **I(S)**. The isoperimetric sets for a graph, **G,** are any sets **S** and **S** for which **h(S) = I** (note that the isoperimetric sets may not be unique for a given graph). The specification of a set satisfying equation (3), together with its complement may be considered as a partition and therefore we will use the term interchangeably with the specification of a set satisfying equation (1). Throughout this paper, we consider a good partition as one with a low isoperimetric ratio (i.e., the optimal partition is represented by the isoperimetric sets themselves). Therefore, our goal is to maximize volume of surface while minimizing Sj. The algorithm considers a heuristic for finding a set with a low isoperimetric ratio that runs in low-order polynomial time.

2.3 Derivation of Isoperimetric Algorithm

Define an indicator vector, X, that takes a binary value at each node

Xi = 1 if vi ϵ S,
= 0 otherwise

Note that a specification of X may be considered a partition. Define the n × n matrix, L, of a graph as

di if i = j
L_{vivj} = - W(e_{ij}) if e_{ij} ϵ E,
= 0 otherwise

The notation L_{vivj} is used to indicate that the matrix L is being indexed by vertices v_i and v_j. This matrix is also known as the admittance matrix in the context of circuit theory or the Laplacian matrix
in the context of finite difference methods By definition of admittance matrix,

$$|\partial S| = X^T L X \text{ ---------- (3)}$$

and $V = X^T d$, where d is the vector of node degrees. If r indicates the vector of all ones, minimizing (3) subject to the constraint that the set, S, has fixed volume may be accomplished by asserting

$V=X^Td=k$---------- *(4)*

where $0 < k < 1/2r^T$ and d is an arbitrary constant and r represents the vector of all ones. We shall see that the choice of k becomes irrelevant to the final formulation. Thus, the isoperimetric constant of a Graph, G, may be rewritten in terms of the indicator vector as

$$IG = \min_x \frac{x^T L x}{x^T d} \text{----------(5)}$$

Subject to (4). Given an indicator vector, X, then I(x) is used to denote the isoperimetric ratio associated with the partition specified by X. The constrained optimization of the isoperimetric ratio is made into a free variation via the introduction of a Lagrange multiplier and relaxation of the binary definition of X to take nonnegative real values by minimizing the cost function.

$Q(X)=X^TLX-A(X^Td-K)$----------(6)

Since L is positive semi-definite and X^Td is nonnegative, Q(x) will be at a minimum for any critical point. Differentiating Q(X) with respect to X yields

$$\frac{dQ(X)}{dX} = 2LX - Ad$$

Thus, the problem of finding the X that minimizes Q(x) (minimal partition) reduces to solving the linear system

$2LX = Ad$---------- *(7)*

Henceforth, we ignore the scalar multiplier 2 and the scalar A. As the matrix L is singular: all rows and columns sum to zero which means that the vector r spans its null space. So finding a unique solution to equation (6) requires an additional constraint.

We assume that the graph is connected, since the optimal partitions are clearly each connected component if the graph is disconnected (i.e., I(x) = I = 0). Note that in general, a graph with **c** connected components will correspond to a matrix *L* with rank (n - c). If we arbitrarily designate a node, vg, to include in S (i.e., fix xg = 0), this is reflected in (6) by removing the gth row and column of L, denoted by L0, and the gth row of X and d, denoted by X0 and d0, such that

L0X0 = d0;

which is a non-singular system of equations. Solving equation (6) for X0 yields a real-valued solution that may be converted into a partition by setting a threshold. In order to generate a clustering or segmentation with more than two parts, the algorithm may be recursively applied to each partition separately, generating sub partitions and stopping the recursion if the isoperimetric ratio of the cut fails to meet a predetermined threshold. We term this predetermined threshold the stop parameter and note that since $0 \leq I(x) \leq 1$, the stop parameter should be in the interval (0; 1). Since lower values of I(x) correspond to more desirable partitions, a stringent value for the stop parameter is small; while a large value permits lower quality partitions (as measured

by the isoperimetric ratio). The partition containing the node corresponding to the removed row and column of L must be connected, for any chosen threshold i.e., the nodes corresponding to X0 values less than the chosen threshold form a connected component.

3 Experimental Results

The above algorithm, (flow chart is given in Appendix A) was applied to different images to analyze the input parameters are valScale, stop, connectivity parameter, cutoff, RecursionCap.

4.1 ValScale

This parameter is used for generating weights and forms the similarity matrix of the pixels. It depends upon the type of image and changes the segmentation efficiency. The figure below shows the segmentation of peppers.png for values of valScale at 200 and 300.

Fig (1) Segmentation for ValScale=300

Fig (2) Segmentation for ValScale=200

For this image 200 is a better ValScale Value.

4.2 Stop

This parameter regulates the execution of the isoperimetric Algorithm. This parameter is the maximum iso perimetric ratio allowable above which the algorithm execution stops. The below figures give the results of the isoperimetric algorithm for two values of Stop.

Fig (3) Segmentation for Stop=1e-5

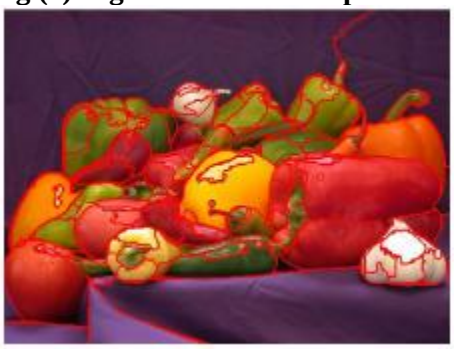
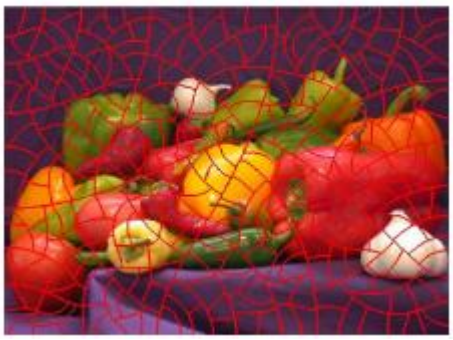


Fig (4) Segmentation for Stop=1e-4

These results show that 1e-5 is a better option for Stop parameter.

4.3 Connectivity Parameter

The pixels are compared with the neighboring pixels on the connectivity basis. We used two connectivity schemes which are 8-connectivity and 4-connectivity.



Fig(5) Segementation using 4-connectivity

Fig(6) Segementation using 8-connectivity

8 is a better option

4.4 Cutoff

This parameter gives the minimum number of nodes that a segment can contain above which the segment is considered valid segment and next partition is considered for partitioning if any.

Fig (7) Segmentation for Cutoff =5

Fig (8) Segmentation for Cutoff =1000

4.5 RecursionCap

This parameter specifies the number of recursions that can take place for the partitioning algorithm on a particular segment and if exceeds considers the next partition for partitioning. It should be chosen such that over segmentation can be avoided. Practically in order to reduce the segmentation time it should be as less as possible.

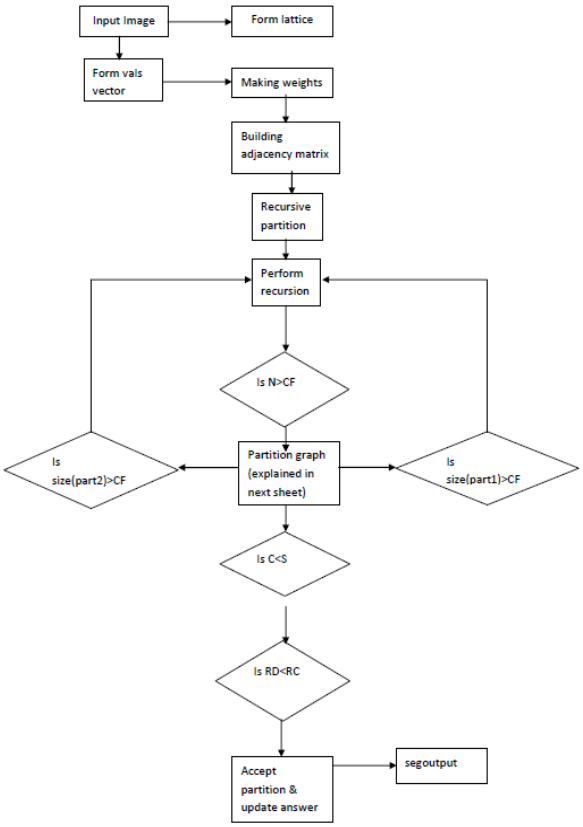
Fig (9) Segmentation for RecursionCap=9

Fig (10) Segmentation for RecursionCap=90

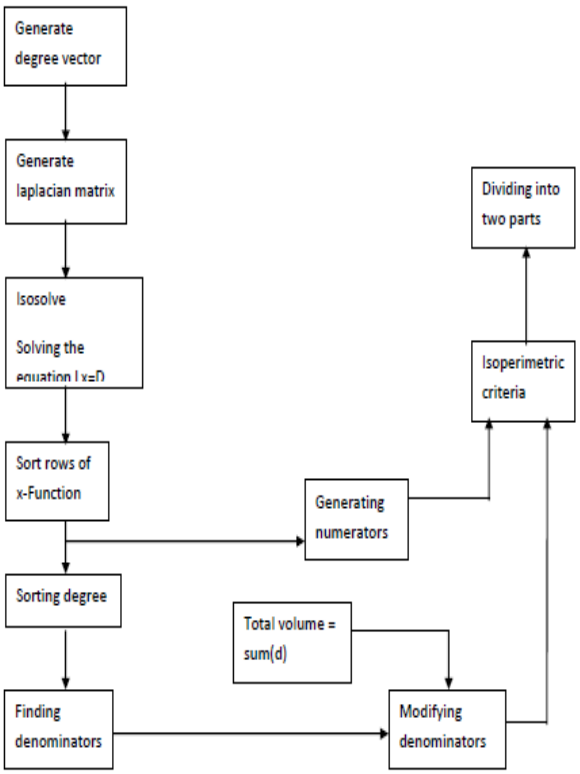
5. Conclusion

In the due course of study, it was founded that all the above parameters should be controlled in order to get the optimum segmentation. The vlaue of these parameter depends on the level of segmentation to be performed. For better segmentation with less time it can be suggested that Valscale and cutoff value can be high and stop and recursioncap vlaue can be low with 8 connectivity.

APPENDIX A (1)



APPENDIX A (2)



References:

[1] Yuri. Boykov and Marie-Pierre Jolly,"Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images", In Proceeding of "International Conference on Computer Vision", Volume I, 105-112, July 2011

[2] Yuri. Boykov and Vladimir Kolmogorov, "An Experiment Comparison of Min-Cut / Max-Flow Algorithms for Energy Minimization in Vision", IEEE Transactions on PAMI, 26 (9): 1124-1137, September 2010

[3] Yuri. Boykov and Vladimir Kolmogorov, "Computing Geodesic and Minimal Surfaces via Graph Cuts", In Proceeding of "International Conference on Computer Vision", Volume II, 26-33, October 2009

[4] Vladimir Kolmogorov and Ramin Zabih, "What Energy Functions can be Minimized via Graph Cuts?",IEEE Transactions on PAMI, 26 (2): 147-159, February 2009

[5] Y. Boy kov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," IEEE Transactions on PAMI, 23 (11): 1222-1239, November 2009

[6] Sudipta Sinha, "Graph Cut Algorithms in Vision, Graphics and Machine learning, An Integrated Paper", UNC Chapel Hill, November 2009

[7] Yuri Boykov and Olga Veksler, "Graph Cuts in Vision and Graphics: Theories and Applications", Chapter 5 of The Handbook of Mathematical Models in Computer Vision, 79-96, Springer, 2009

[8] Thomas Cormen, Charles Leiserson, Ronald Rivest and Clifford Stein, "Maximum Flow", Chapter 26 of Introduction to algorithms, second edition, 643-698, McGraw-Hill, 2005

[9] L. R. Floulds, "An Introduction to Transportation Networks", Section 12.5.3 of Graph Theory Applications, 246-256, Springer, 1992