CS736 Medical Image Computing

Image Segmentation Using Constrained s-t Cuts

Hastyn Doshi
Sarthak Mittal
Kartik Gokhale

Department of Computer Science & Engineering
IIT Bombay

2022

Introduction

- A central problem in image processing is the **segmentation of an image** into various coherent regions, which broadly means associating a 'label' to every pixel in an image.
- In medical imaging, there is a high demand for extracting a region of interest(ROI) from an image, which can be formulated as a problem of **foreground/background segmentation**: labeling each pixel in an image as belonging to either the 'foreground' or the 'background'.
- This involves 'separating' the image into 2 parts. Thus, this can be re-modelled to the **maximum-flow/minimum-cut** problem that can be solved **efficiently** by various algorithms
- The efficacy of such a formulation can be increased by introducing some constraints to be met by the generated s-t cut

Current Approaches

- The standard solution using s-t cuts involves graph generation, that is, creation of an undirected graph $G = (V, E)$ having V as the set of pixels with E denoting set of all pairs of **neighboring** pixels along with a 'source' and 'sink' to represent a flow network
- We assign labels to pixels based on selection of an optimal partition such that a certain **objective function**, which depends on likelihood and separation parameters, is maximized.
- However, the **accuracy** of this approach, measured by percentage of correctly classified pixels, can be improved by introducing **constraints**. These constraints involve using a shape/label prior or using user-defined scribbles or generating new pixels (super-pixels) entirely

Current Approaches

Using prior shape constraints

A new objective function was designed which takes into account shape information, image appearance and spatial dependence. Rigid registration was done to align training images, followed by use of a spatial interaction model and then graph cuts optimization.

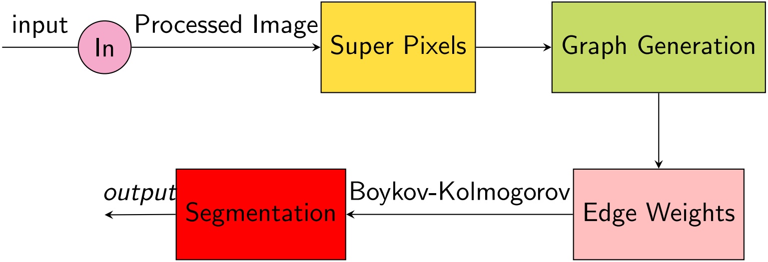
Using image with scribbles

Automation of deciding between foreground and background can be challenging, so assistance is taken from the user in form of a few scribble lines. Using that, the algorithm identifies the partition and completes the segmentation.

Overview

The framework broadly comprises 2 steps

- Generating Super-Pixels
- Generating a graph and performing constrained s-t cuts on it



SuperPixel Generation

Simple Linear Iterative Clustering is an algorithm for converting an image into Superpixels. SLIC gives the desired number of compact superpixels with the least computational overhead.

This algorithm performs **local clustering** over 5-dimensional data.

This algorithm takes the desired number of superpixels as an input K. Therefore an image with N pixels will have N/K **superpixels**, and approximately for every $\sqrt{N/K}$ grid interval, there will be a superpixel center.

SLIC algorithm follows taking a sample of K cluster centers; later move them to the lowest gradient position in a $\mathbf{3} \times \mathbf{3}$ **neighborhood**. If a cluster center search area overlaps with a pixel, then that pixel is assigned to that cluster center. After all pixels are associated, new cluster centers will be calculated using an average. This process will continue until **convergence**.

Flow Networks

So what exactly is a flow and a flow network graph? A useful intuition for this problem is to think of a network of pipes which carry fluid, each of different capacities. You are given a source of fluid and the task at hand is to transport maximum amount of fluid as possible to a sink while ensuring no pipe carries more than their capacity and that at every junction of more than one pipe there is as much fluid entering as is leaving. With this intuition we can look at some notation.

A flow network G = (V,E) is a directed graph in which each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$. We further require that if E contains an edge (u, v), then there is no edge (v, u) in the reverse direction. (We shall see shortly how to work around this restriction.) If $(u, v) \notin E$, then for convenience we define $c(u, v) = 0$, and we disallow self-loops. We distinguish two vertices in a flow network: a source s and a sink t.

Max Flow

We are now ready to define flows more formally. Let G = (V,E) be a flow network with a capacity function c. Let s be the source of the network, and let t be the sink. A flow in G is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ that satisfies the following two properties

Capacity Constraint

For all $u, v \in V$, we require

$$0 \leq f(u, v) \leq c(u, v)$$

Flow Conservation

For all $u \in V - s, t$, we require

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

When $(u, v) \notin E$, there can be no flow from u to v, and $f(u, v) = 0$

Generating a Graph

The method considered Vertices of G is comprised of the Superpixels (SP). V also contains two extra terminal nodes called Source s and Sink t and they represent the object and background B labels respectively.

If centroid of a super-pixel is within the range of 5d (where d is the approximate diameter of a super-pixel) of another, both are considered as neighbors. Let P be the arbitrary set of Superpixels in a given image and N represents a neighborhood system among these SP's by a set of unordered pairs p, g A is a binary vector $\{A1, A2.\ldots..A|p|\}$, where value A_p determines whether p belongs to the set F or B.

(a) Image with seeds

(b) Graph with n-links and t-links

Utilising Duality

Max flow and Min Cut

So far we have been looking at a flow problem, but the problem of segmentation was a minimum cut problem so why would be looking at maximum flow algorithms.

The answer is that both these problems are two sides of the same coin, They are what is called a dual.

This means that the value of maximum flow we obtained via Ford Fulkerson is exactly the value of the minimum cut in the graph.

This is the consequence of a well known theorem called Max-flow Min-cut theorem which has fun proofs which would require more properties of cuts and flows we have seen so we will not discuss it in this presentation.

Ford Fulkerson Algorithm

Ford Fulkerson Algorithm

```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min {c_f(u, v) : (u, v) is in p}
    for each edge (u, v) in p
        if (u, v) ∈ E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
```

Figure: Ford Fulkerson

The process of greedily pushing as much fluid as possible through a path is known as augmenting a path. Also note that the standard implementations of this algorithm use search techniques like BFS to find paths.

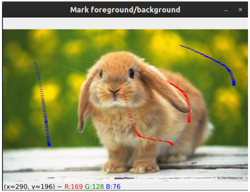
Boykov-Kolmogorov Algorithm

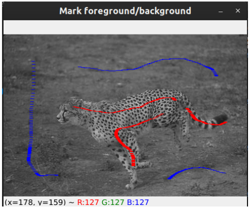
Boykov-Kolmogorov Min-Cut/Max-Flow

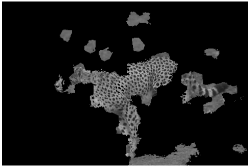
In Ford Fulkerson algorithm, the algorithm will start building a breadth-first tree for path $s \rightarrow t$ when a path of given length go exhausted. Such operations will be very expensive in images since breadth-first tree generation involves scanning through all pixels/SP. To overcome this disadvantage, the method will be using an improved algorithm from the family of augmenting path algorithms, called Boykov-Kolmogorov Min-Cut/Max-Flow algorithm. This algorithm creates two search trees from source and sinks nodes, and these trees are reused to avoid further traversals. This algorithm is having a complexity of $\mathcal{O}(mn^2|C|)$ which theoretically is higher than the standard algorithm, but it outperforms them by its unique tree reusing idea. Algorithm runs in 3 different phases - growth, augmentation and adoption.

Results

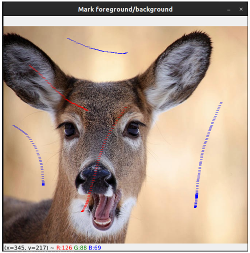
We show few of the segmentation outputs along with their input images.

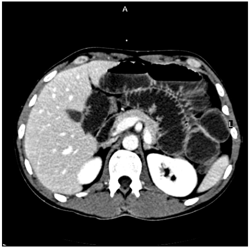


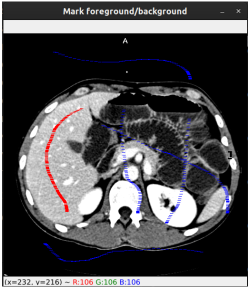


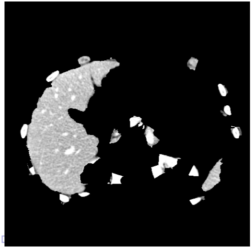


Results


Mark foreground/background
(x=345, y=217) ~ R:126 G:88 B:69


A
L


Mark foreground/background
A
L
(x=232, y=216) ~ R:106 G:106 B:106



Which Method?

Using shape prior

This method combined graph cuts with shape constraints for segmentation. Shape variations were estimated using a distance probabilistic model. Image signal were approximated with a linear combination of Gaussians, and spatial interaction was estimated using MRF. A new energy function was used for obtaining optimal segmentation. Experimental results showed that shape constraints were able to overcome gray level inhomogeneity and the graph cuts resulted in much more accurate segmentations (with mean error 5.7% and standard deviation 0.9%) compared to graph cuts without shape constraints (mean error 49.8% and standard deviation 24.3%).

Which Method?

Using scribbles

Computational complexity was significantly reduced by a combination of superpixels and CIDE2000. Color spaces play an important role in metric calculation and processing. The approximation done by Boykov-Kolmogorov algorithm gives faster running time than traditional graph cut algorithms.

Conclusion

The first method works well when there is inhomogeneity and random noise is involved. This method can possibly be improved by incorporating ideas from the second method. The second method works well when the color difference between object and surrounding is distinguishable. This method can be further extended using more visual cues like texture.

Conclusion

Thank you!