IT3020E - Discrete Mathematics
Application of Minimum-cut algorithm in Image Segmentation

Hoang Long Vu - 20204897

January, 2022

1 Introduction

Image segmentation is the process of partitioning a digital image into multiple image segments, also known as image regions or image objects (set of *pixels*). One of the most basic problems relating to image segmentation is that of *foreground/background segmentation*. In this problem, we wish to label each pixel in an image as belonging to either the foreground of the scene or the background.

The (i) input image, and (ii) a possible segmentation of the image.

Figure 1: An example of image segmentation

A digital image is an image composed of picture elements, also known as pixels, we can picture the pixels as constituting a grid of dots, and the *neighbours* of a pixel are the ones that are directly adjacent to it in the grid, as shown in the Figure 2. Now let V be the set of pixels in the image that we are analysing, and E is the set of all pairs of neighbouring pixels. We obtain a pixel graph $G(V, E)$ as an input.

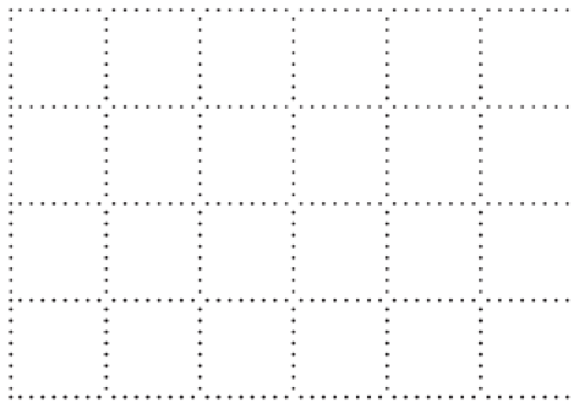

Figure 2: A pixel graph

2 Approach

A natural approach is letting f_i be the likelihood that the pixel i belongs to the foreground ($f_i \geq 0$). Similarly, $b_i \geq 0$ is the likelihood that the pixel i belongs to the background. Therefore, we would want to label pixel i as belonging to the foreground if $f_i > b_i$, or to the background otherwise. However, we may make the labeling "smoother" by minimising the amount of foreground/background boundary by forcing the decision made on pixel i affected by the decisions on its neighbour. Concretely, if many of i's neighbours are labelled "background", then we should be more inclined to label i as "background" too. Therefore, for every two adjacent pixels i and j, we have a *separation penalty* $p_{ij} \geq 0$ which is the "price" of placing one of i or j in the foreground and the other in the background (i.e. separating i from j).

Now our objective (*optimal labelling*) in the Image segmentation problem is: Find a partition of the set of pixels into sets F and B (foreground and background) that maximises:

$$q(F, B) = \sum_{i \in F} f_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E, |F \cap \{i,j\} = 1|} p_{ij} \tag{1}$$

Specifically, we will be "rewarded" for having high likelihood values and penalised for having neighbouring pairs (i, j) with one pixel in F and the other in B.

3 Transforming to Minimum-cut problem

The optimal labelling problem above bears significance resemblance to the Minimum-cut problem. Now we wish to make some modifications to transform it to Minimum-cut problem:

1. Firstly, Minimum-cut problem deals with minimisation problems, whereas Equation (1) refers to a maximisation one. Notice that we can rewrite the equation as:

$$q(F, B) = \sum_{i \in V} (f_i + b_i) - (\sum_{i \in B} f_i + \sum_{j \in F} b_j + \sum_{(i,j) \in E, |F \cap \{i,j\} = 1|} p_{ij})$$

Therefore, maximising $q(F, B)$ is now equivalent fo minimising $u(F, B)$:

$$u(F, B) = \sum_{i \in B} f_i + \sum_{j \in F} b_j + \sum_{(i,j) \in E, |F \cap \{i,j\}=1|} p_{ij} \tag{2}$$

2. Our graph G is an undirected graph, thus we may transform to Minimum-cut problem by replacing each undirected edge connecting (i, j) with two directed edges, (i, j) and (j, i), each with capacity p_{ij}. This is reasonable because in any s-t cut, at most one of these two oppositely directed edges can cross from the s-side to the t-side of the cut.

3. Our problem is still missing the source and the sink, therefore we can create a new supersource s to represent the foreground, and a supersink t representing the background. Next, we need to attach each of s and t to every pixel and use f_i, b_i to define appropriate capacities on the edges between pixel i and the source/sink. Concretely, for each pixel i we may add an edge (s, i) with capacity f_i, and an edge (i, t) with capacity b_i.

Now we yield the resulted new flow graph G' as shown in the figure below:

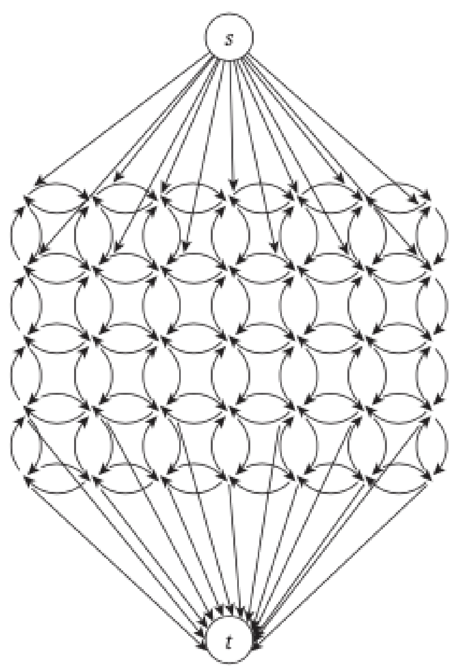

Figure 3: Resulted flow graph

Now, an s-t cut(F, B) corresponds to a partition of the pixels into sets F and B. We can group the edges crossing the cut(A, B) into three categories:

- Edge (s, i) where $i \in B$: contributes f_i to the capacity of the cut.
- Edge (j, t) where $j \in F$: contributes b_j to the capacity of the cut.

- Edge (i, j) where $i \in F$ and $j \in B$: contributes p_{ij} to the capacity of the cut.

Figure 4 illustrates a sample s-t cut on a graph constructed as above.

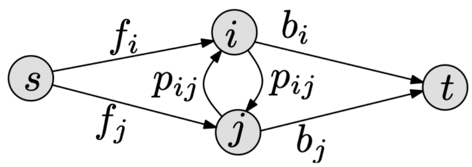

Figure 4: An s-t cut on a simple graph with 2 pixels

Consider the capacity of the cut (F, B):

$$c(F, B) = \sum_{i \in B} f_i + \sum_{j \in F} b_j + \sum_{(i,j) \in E, |F \cap \{i,j\} = 1|} p_{ij} \equiv u(A, B)$$

This is exactly the quantity $u(A, B)$ we wish to minimise in Equation 2. Therefore, applying Minimum-cut in the resulting graph G' would corresponds to the required segmentation. A simple desired result is illustrated in the figure 5

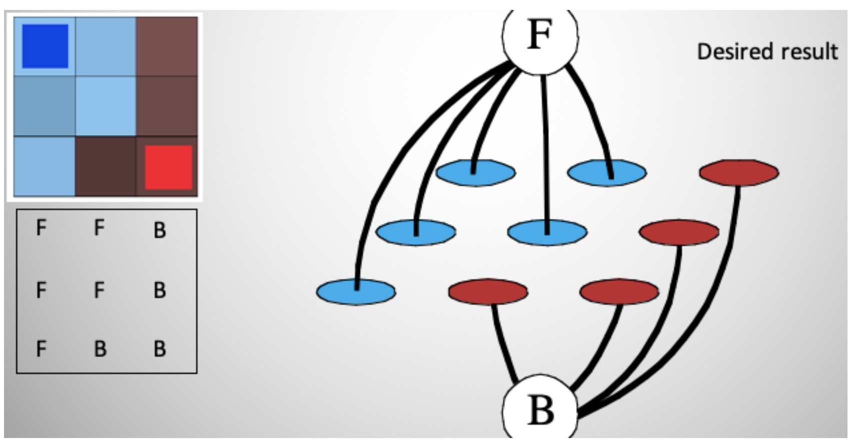

Figure 5: A desired result after applying Minimum-cut

4 Conclusion

The solution to the Image Segmentation Problem can be obtained by a minimum-cut algorithm in the graph G' (Figure 3) constructed above. Moreover, by **The minimum-cut max-flow theorem**, we yield that: One can solve the segmentation problem, in *polynomial time*, by computing the max flow in the graph G'.

References

Eva Tardos Jon Kleinberg. *Algorithm Design*, chapter 7. Cornell University.

Dr. Ulas Bagci. *Advanced Image Segmentation (Optimization Problem)*. http://www.cs.ucf.edu/~bagci/teaching/computervision16/Lec13.pdf.

Kevin Wayne. *Maximum Flow Applications*. https://www.cs.princeton.edu/~wayne/cs423/lectures/max-flow-applications.