Report on Fast Approximate Energy Minimization via Graph Cuts

Felix Wechsler
Technische Universität München, Germany
felix.wechsler@tum.de

November 29, 2017

Abstract. In this report we want to give an overview and introduction to the field of graph cuts in computer vision. Most parts of the report can be found in [2] on which my work is based. In the first steps we give an insight how the energy of images can be defined and why the terms look like that. Afterwards we explain how we can create a graph out of an image and what properties they fullfill. Having the knowledge about the graphs we want to show why the graph cut is a proper way to find an energy minimum. For finding the minimum we will shortly dive into the topic of maximum-flow minimum-cut. At the end, we will show a few example images which are minimized by our `Python` implementation.

1 Introduction

In our report, we will call pixels with $p, q \in \mathcal{P}$. p and q are some pixels which are in the set of pixels $\mathcal{P}$. In the set of pixels $\mathcal{P}$ all pixels from an image are contained.
In graph cuts algorithms you assign every pixel a label. All possible labels are contained in $\mathcal{L}$.
The label of pixel p can be written as f_p where $f_p \in \mathcal{L}$.
In image restoration of 8-bit-grayscale images labels are the same as intensities. Therefore the set of labels is $\mathcal{L} = \{x \in \mathbb{N}_0 | x \leq 255\}$. For image segmentation an arbitrary number of labels is chosen. If you want to segment the image into five parts you would choose five labels. Figure 1 shows how an algorithm could change the label of images. The original labeling of all pixels is simply called

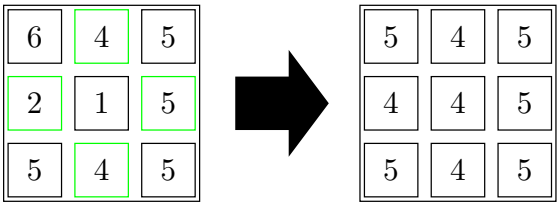


Fig. 1: The left image with label f is the input image. One algorithm could change the label of the image to the right image called $\hat{f}$. Green marked pixels are neighbours of the middle pixel.

f, the updated labeling $\hat{f}$. Extending the previous notation we will use $\mathcal{P}_{\alpha,\beta}$ to express a set which only contains pixels which have the current label α or β. For example in Figure 1 $\mathcal{P}_{2,1}$ would be the set of the pixel in the middle and the left neighbour.
In Figure 1 the green marked pixels are called neighbours of the pixel in the middle. If we write $\mathcal{N}$ then we mean the set of tuples of neighbours. So there would be four tuples where the middle pixel is in a tupel with the surrounding four. We add tuples for all neighbours in the whole image.

2 Energy in images

Many computer vision problems can be formulated in a way that the energy in an image should be minimized. For example intensities of images are normally piecewise smooth but will chance their value rapidly at object boundaries. But intensity does also change if there is some noise at single pixels. To make a compromise of matching the original image but also achieving many areas of smooth labels we can formulate the energy of an image:

$$E(f) = E_{\text{smooth}}(f) + E_{\text{data}}(f) \tag{1}$$

Here f is the above mentioned labeling of all pixels in the image. E_{data} is the part which cares about fitting the original image, E_{smooth} considers the smooth parts.

2.1 Energy of data

Normally the data term is:

$$E_{\text{data}}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \tag{2}$$

For D_p often the quadratic norm is chosen, in image restoration it could be $(f_p - i_p)^2$ where i_p is the original intensity of the pixel. However in my experiments a later presented energy function worked better.

2.2 Smoothness energy

The smoothness term can be defined in many possible ways. In most cases this frame is useful:

$$E_{\text{smooth}} = \sum_{(p,q)\in\mathcal{N}} V_{p,q}(f_p, f_q) \tag{3}$$

This term sums a potential function over all neighbours. One could simply use $V_{p,q}(f_p, f_q) = K \cdot |f_p - f_q|$ but also $V_{p,q}(f_p, f_q) = \min(K, \|f_p - f_q\|)$ (with arbitrary constant K) and many others are possible.

3 Label changing algorithms

In the following part we want to present you one of the main two algorithms for minimizing the energy in an image given in the paper [2].

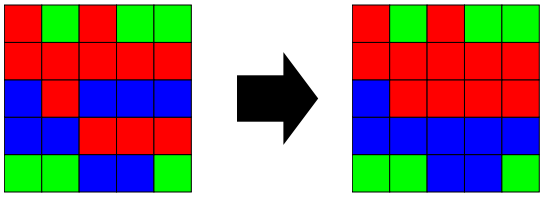
3.1 α-β-swap

The pure algorithm for optimizing a imagen via α-β-swap is given in 1. The run-

Algorithm 1 α-β-swap algorithm

```
1:  procedure SwapMove(f)
2:      f ← arbitrary labeling
3:      s ← 0
4:      for s = 1 do
5:          s ← 0
6:          for all {α, β} ⊂ L do
7:              f̂ ← arg min E(f') within one α-β-swap
8:              if Ê(f̂) < E(f) then
9:                  f ← f̂
10:                 s ← 1
11:     return f
```

time is obviously in $\mathcal{O}(|\mathcal{L}|^2)$ and furthermore it can be proven that the algorithm converges after $\mathcal{O}(|\mathcal{P}|)$ cycles [5]. One cycle is defined as line 6 to 10 in algorithm 1. In line 6 we use the the α-β-swap to find the optimum. In the α-β-swap you only work with pixels in $\mathcal{P}_{\alpha,\beta}$. Changing only these pixels you want to find a new labeling which either asigns α or β to the pixels. Figure 2 shows that if α = red and β = blue only pixels of these two labels are touched. All pixels with other labels aren't changed in that cycle but will be changed in the next cycles if α and β change. section 4 will show which pixel receives which label. Which pixels receive which colours will be part of the section 4. How to find the energy minimum via α-β-swap is explained in the following two sections.



labeling f labeling $\hat{f}$

Fig. 2: α-β-swap changes only red and blue pixels

3.2 α-expansion

For the α-expansion a similar algorithm can be constructed. Due to page restrictions, this part is left out and can be found in [2].

4 From image to graph

Now we want to concentrate us now on the α-β-swap but for the α-expansion the procedure is similar.

4.1 α-β-swap

In Figure 3a the constructed graph for a 1D-image is shown. In the first steps you add all pixels which have label α or β as nodes to the graph. For the labels itself you introduce two terminal nodes. Nodes which are neighbours in the original image are connected by an n-link. All nodes have an edge to the terminal nodes, called t-links. Note that there is no n-link w and y because they aren't neighbours, pixel x is between. But x hasn't label α or β so it does not appear in the graph.
To include the information about the energy we add some edge weights which are given in Table 1. They are chosen like that to transform the energy minimizing problem into a graph-cut.

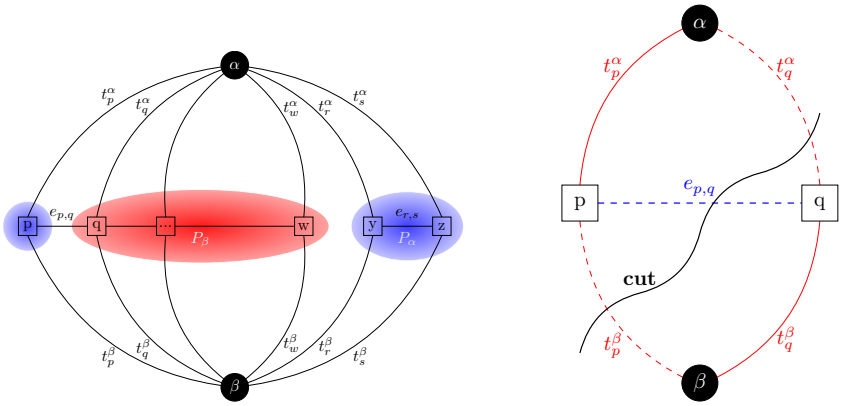
5 Minimum graph cut

In the previous section we saw how the graph was constructed. Now we want to show how we can find the minimum energy with graph cuts.

5.1 Procedure with graph cut

A cut [1] on our graph means that there is no way from node α to node β. Minimum cut means that the sum of all cut edges weights must be minimal.

[1] sometimes called s-t-cut or sink-flow-cut



(a) Constructed graph of an 1D-image (b) Possible graph cut

Fig. 3: Graph and graph cut

edge	**weight**	**for**
t_p^α	$D_p(\alpha) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha,\beta}}} V_{p,q}(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha,\beta}$
t_p^β	$D_p(\beta) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha,\beta}}} V_{p,q}(\beta, f_q)$	$p \in \mathcal{P}_{\alpha,\beta}$
$e_{p,q}$	$V_{p,q}(\alpha, \beta)$	$\{p,q\} \in \mathcal{N} \wedge p, q \in \mathcal{P}_{\alpha,\beta}$

Table 1: Edge weights for α-β-swap

One example of a cut is given in Figure 3b. Depending on which of the t-links are cut, new labels for the pixels are assigned. In this example, p would have assigned the label β and q α. In general, if the t-link from pixel p to α is cut then the new assigned label is α. If the t-link to β is cut then, label β would be assigned.

5.2 Connection between energy and graph

In a mathematical formulation the cost of a cut can be expressed with

$$|\mathcal{C}| = \underbrace{\sum_{p \in \mathcal{P}_{\alpha,\beta}} \left|\mathcal{C} \cap \{t_p^\alpha, t_p^\beta\}\right|}_{\text{cost of t-edges}} + \underbrace{\sum_{\substack{\{p,q\} \in \mathcal{N} \\ \{p,q\} \subset \mathcal{P}_{\alpha,\beta}}} |\mathcal{C} \cap e_{p,q}|}_{\text{cost of n-links}} \tag{4}$$

where $\mathcal{C}$ is the set of cut edges. This sum can be rewritten to

$$|\mathcal{C}| = E(f^{\mathcal{C}}) - K. \tag{5}$$

K is a constant which is independent of the cut and $E(f^{\mathcal{C}})$ is the energy of the labling after the cut. One does see if we minimize the left term via the graph cut we find a minimum of the energy in the constructed graph. Note that this is not strictly a global minimum of the image and its labeling itself. However, it can be shown that it is within a known range of the optimum [5].

5.3 Finding a minimum cut

Quite a few algorithms exist in literature for the minimum graph cuts. One famous theorem is the *Max-flow min-cut* which states that the maximum flow through a network is equivalent to the cost of a minimum cut. Out of that the *push-relabel* or *Ford-Fulkerson* based algorithms were derived.

6 Implementation

For our graph structure Kolmogorov and Boykov showed that their algorithm works in practice faster than other known. The algorihtm and benchmarks can be found in [1] and [3]. Kolmogorov himself also provides an fast `C++` implementation called `Maxflow`[2] [4]. I did also a small implementation [3] which uses `PyMaxflow`[4] for the graph cuts. The objective of our code was not speed and memory efficiency. It's only purpose was to test different energy functions on images to see the change in the results.

7 Experiments

In our experiments we found out that using $\hat{D}_p(f_p) = \sqrt{|f_p^2 - i_p^2|}$ and $\hat{V}(f_p, f_q) = |f_p - f_q|$ delivered the best results. Choosing $\hat{D}_p$ can be motivated by the fact that the previous shown expressions for D_p and $V(f_p, f_q)$ have a mismatch in scaling behaviour. The first scales quadratic the other linear. To eliminate this we choose $\hat{D}_p$ which scales linearly but also has a *quadratic component*.

7.1 Example images

Figure 6 shows to the right the output image (produced with our α-β-swap implementation) using the classical euclidean energy function and the absolut different potential. The middle image is the output using our energy function.

[2] `http://pub.ist.ac.at/~vnk/software.html`
[3] `https://github.com/roflmaostc/Fast-Approximate-Energy-Minimization-via-Graph-Cuts`
[4] `http://pmneila.github.io/PyMaxflow/`

One can easily notice that the right image is very smooth and many parts of the originally structured skin is lost.

(a) 20% noise

(b) 10 cycles; $\hat{D}_p$; $\hat{V}$

(c) 10 cycles; $D_p = (f_p - i_p)^2$; $V = 80 \cdot |f_p - f_q|$

Fig. 4: Comparison of different outputs of a noisy leopard

Using Equation 1 it is also possible to print out the energy after every cycle. Figure 5 shows that the biggest energy drop occurs always during the first cylce. Furthermore the energy doesn't change much after the second and third cycle. This implies that in practice you shouldn't wait for convergence for a fast result but stop after a few cycles.

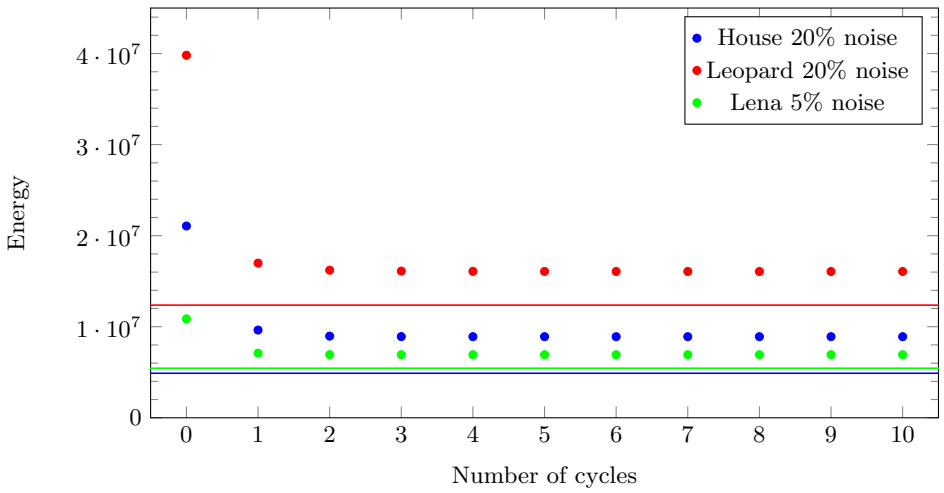
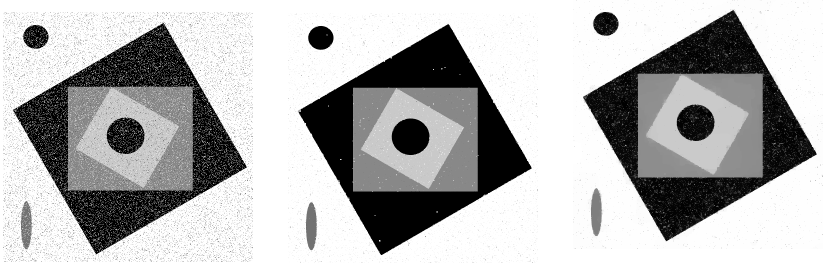


Fig. 5: Energy over cycles obtained by our implementation. The horizontal lines are the energy of the original, clean image



(a) 15% noise

(b) 10 cycles; $\hat{D}_p$; $\hat{V}$

(c) 10 cycles; $D_p = (f_p - i_p)^)$; $V = \max\left(500, (f_p - f_q)^2\right)$

Fig. 6: Comparison of different outputs

8 Conclusion

Graph-cuts are still state of the art in computer vision because they can minimize the energy in the whole image and not only piecewise. However today the α-expansion algorithm is used more because in total you have far less graph cuts to find. Doing the α-β-swap for grayscale images you've already got ≈ 30000 iterations due to the swap of all different pairs of labels. Furthermore, in each swap the graphs are constructed and a minimum cut has to be found. The α-expansion has bigger graphs but only a few hundred which is much less. Excluding deep learning graph cuts in computer vision are still a very interesting topic for doing task like segmentation.

References

[1] Yuri Boykov and Vladimir Kolmogorov. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.9 (Sept. 2004), pp. 1124–1137.

[2] Yuri Boykov, Olga Veksler, and Ramin Zabih. "Fast Approximate Energy Minimization via Graph Cuts". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 23.11 (2001), pp. 1222–1239.

[3] Vladimir Kolmogorov. "Graph Based Algorithms for Scene Reconstruction from Two or More Views". AAI3114475. PhD thesis. Ithaca, NY, USA, 2004.

[4] Vladimir Kolmogorov. *Maxflow*. Version 3.04. Oct. 30, 2017. URL: `http://pub.ist.ac.at/~vnk/software.html`.

[5] Olga Veksler. "Efficient Graph-based Energy Minimization Methods in Computer Vision". PhD thesis. Ithaca, NY, USA, 1999.