EXERCISE 4 - OPTIMAL TRANSPORT AND GRAPH CUTS

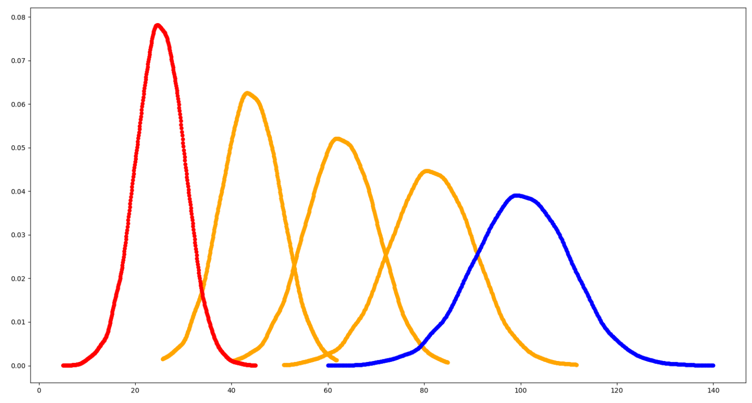
Akiyo WOROU

April 24, 2021

1 PART 1: Optimal Transport

1.1 Task 1 - Optimal Transport in 1D

The Optimal transport value is computed and approximately equals to 75.12. By using the closed form formulation the value is almost 75.86. The error is less than 1%, so the closed approximation can be used for faster computation. The distribution plots obtained for $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$ are represented as follow :



1.2 Task 2 - Color Transfer Using Sliced Optimal Transport

- The theta vector are generated by using spherical coordinates with $r = 1$. The generated vector is given by : $[rcos(\theta)sin(\phi), rsin(\theta)sin(\phi), rcos(\phi)]$.
- For a set of RGB values $\{p_i\}$, the projection set is given by $\{ps_i\}$ with $ps_i = \frac{<p_i,\theta>}{||\theta||^2} =< p_i, \theta >$
- The $\{ps_i\}$ sets are then sorted and used to compute the result. Here is the result obtained:

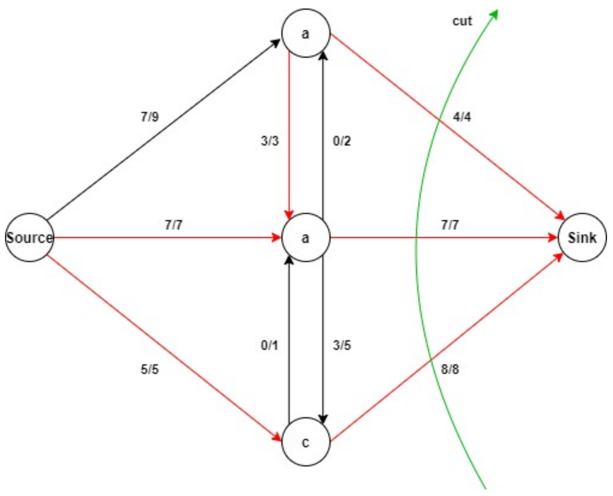
2 PART 2: Interactive Segmentation with Graph Cut

2.1 Task 1 - Handling Max Flow

The graph has 3 nodes and 4 edges. The 3 nodes are renamed respectively from the top to the bottom as a, b and c. The adjacency matrix can be written as follow:

$$\begin{pmatrix} 0 & 3 & 0 \\ 2 & 0 & 5 \\ 0 & 1 & 0 \end{pmatrix}$$

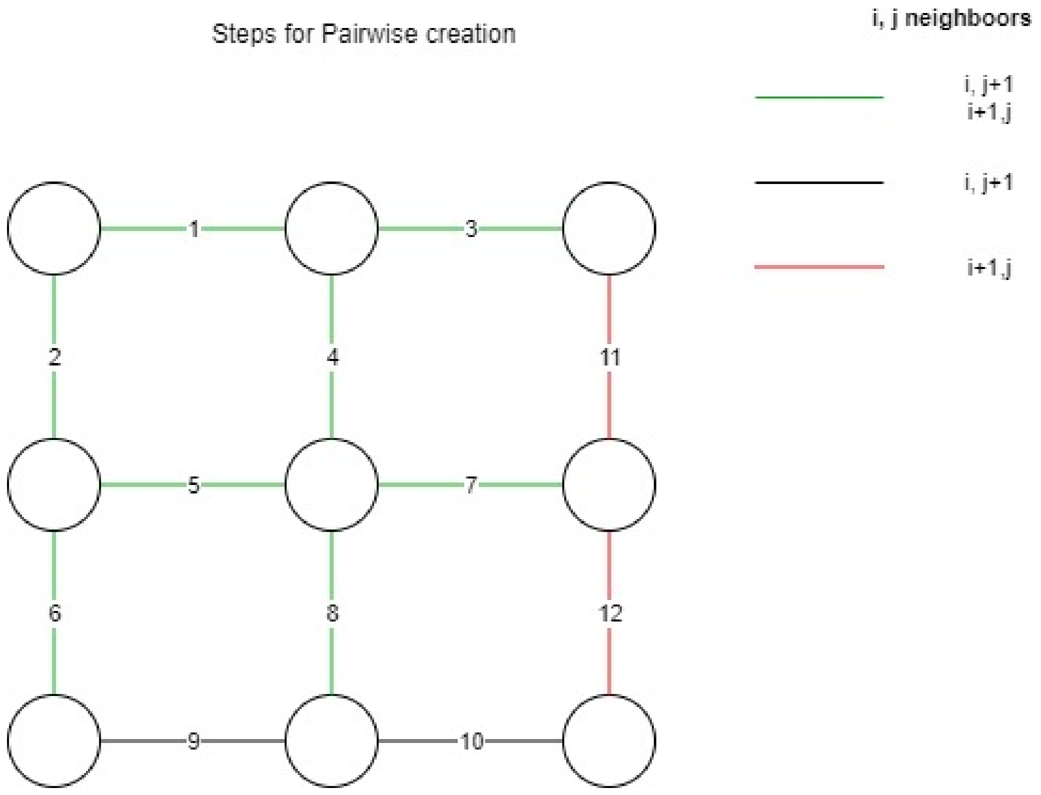
The algorithm and hand testing give 19 as max flow. The result can be seen in the graph below:



The optimal labeling is given by $[False, False, False]$ which is equivalent to $[0, 0, 0]$. The cut is then {Source, a,b,c}, {Sink}.

2.2 Task 2: Interactive Segmentation

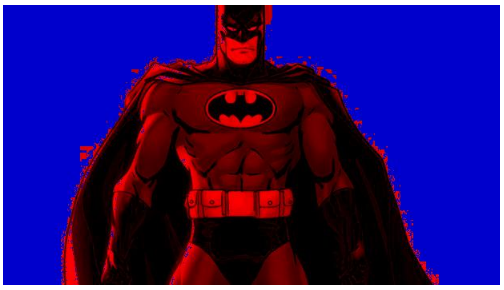
The same process was used for the image segmentation. The following graph shows how the edges and nodes are computed :



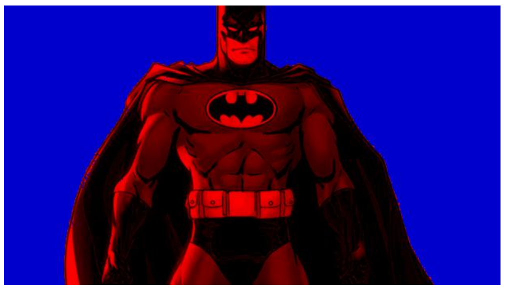
The unaries are computed simply by using the table in the given paper. We can then see the results for different images and different values of λ. I had better results without Gaussian smoothing than with Gaussian smoothing (at least for $\sigma = 7$).

Discussion:

The graph cut can be useful for image segmentation but the results varies in function of the image. We need the good λ for each image. So the algorithm is not that efficient. The algorithm does not guess automatically what without labeling the part of the image that we want in foreground. Furthermore it is computationally expensive and can have some artifacts. Using some deep learning techniques may be the best way to achieve a good and fast image segmentation.

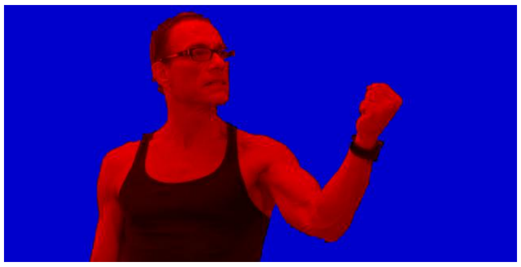


(a) Batman $\lambda = 1.0$

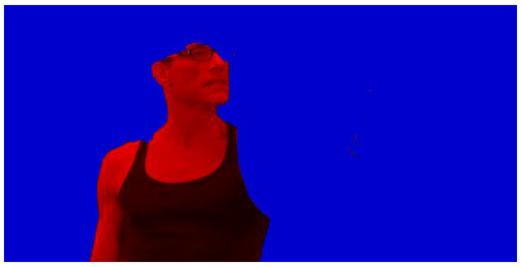


(b) Batman $\lambda = 2 \cdot 10^{-3}$

Figure 1: The Dark Knight Segmented without Gaussian smoothing



(a) Van Damme $\lambda = 1.0$



(b) Van Damme $\lambda = 10^{-4}$

Figure 2: The Dark Knight Segmented without Gaussian smoothing

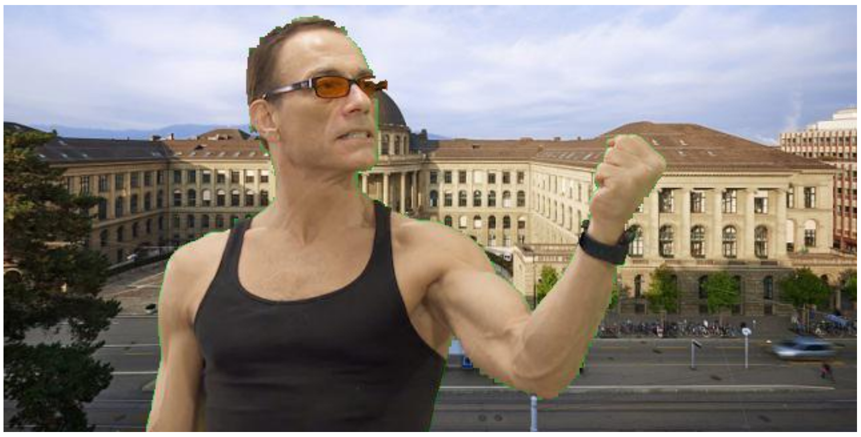


Figure 3: Van Damme at ETHZ $\lambda = 1$

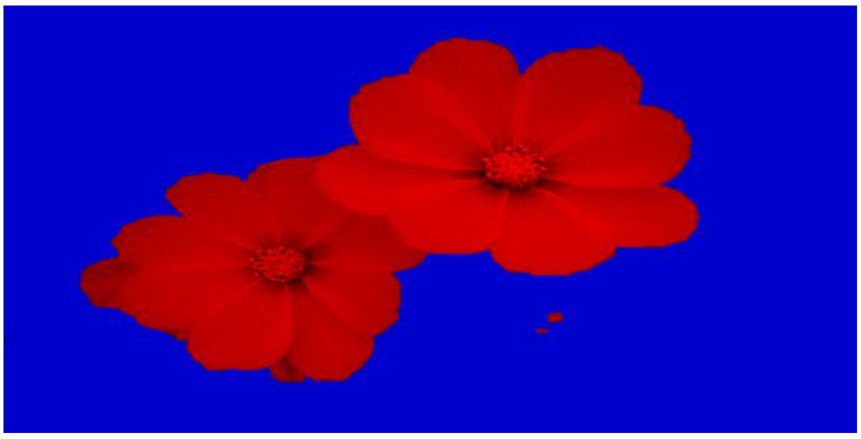


Figure 4: flowers segmented $\lambda = 1$