Graph Learning as a Basis for Image Segmentation

Wille Eriksson and Kim Lundbeck

***Abstract*—Graph signal processing is a field concerning the processing of graphs with data associated to their vertices, often in the purpose of modeling networks. One area of this field that has been under research in recent years is the development of frameworks for learning graph topologies from such data. This may be useful in situations where one wants to represent a phenomenon with a graph, but where an obvious topology is not available. The aim of this project was to evaluate the usefulness of one such proposed learning framework in the context of image segmentation. The method used for achieving this consisted in constructing graph representations of images from said framework, and clustering their vertices with an established graph-based segmentation algorithm. The results demonstrate that this approach may well be useful, although the implementation used in the project carried out segmentations significantly slower than state of the art methods. A number of possible improvements to be made regarding this aspect are however pointed out and may be subject for future work.**

***Sammanfattning*—Grafsignalbehandling är ett ämnesområde vars syfte är att behandla grafer med data associerat till deras noder, ofta inom nätverksmodelleringen. Inom detta område pågår aktiv forskning med att utveckla tekniker för att konstruera graftopologier från sådana data. Dessa tekniker kan vara användbara när man vill representera ett fenomen med grafer, men då uppenbara grafstrukturer inte finns tillgängliga. Syftet med detta projekt var att utvärdera användbarheten hos en sådan teknik när den appliceras inom bildsegmentering. Metoden som användes bestod i att konstruera grafrepresentationer av bilder med hjälp av denna teknik, för att sedan behandla dessa med en etablerad, grafbaserad segmenteringsalgoritm. Resultaten påvisar att detta tillvägagångssätt under rätt förutsättningar kan producera tillfredsställande bildsegmenteringar. Dock är implementeringen som nyttjats i projektet betydligt långsammare än de metoder som vanligen används inom området. Ett antal förslag till prestandaförbättring utpekas, och kan vara föremål för framtida studier.**

***Index Terms*—Graph Signal Processing, Image segmentation, Graph Learning**

Supervisor: Magnus Jansson

TRITA number: TRITA-EECS-EX-2020:160

I. Introduction

In recent years, much development has been done in the field of graph signal processing (GSP) [1]. The field aims to generalize methods used for processing signals in the time-domain to the less regular domain of graphs. Whereas we typically think of a signal as a set of values recorded and read at certain time instances, another perspective is to associate these values to different vertices of a graph. This latter perspective is often useful in for example network modeling, as well as in other applications of data science [2].

A challenge one often faces in the field of GSP is that even though data points may be available, a suitable graph topology describing the connection between these is not. In recent years research has been put into both developing methods to learn such topologies from the data, as well as finding which applications are suitable for its use [3] [4]. One example involves interpolating temperature data for missing weather stations in a grid [5].

An application that may be of interest is the field of image segmentation, the practice of partitioning images into areas of interest so as to extract higher level information. As an example we have the field of autonomous vehicles, where this could be used to recognize drivable surfaces [6]. Many segmentation methods use graph topologies describing images as their basis. One example of this is the normalized cuts algorithm [7] developed in the early 2000s, another is a method proposed by the professors Pedro Felzenszwalb and Daniel Huttenlocher at MIT and Coronell University [8].

In this project, we use a framework to learn the graph topology of an image and try to evaluate how well this works as a basis for image segmentation.

II. Preliminaries

The purpose of this section is to present the underlying theory enabling our project. We will first discuss a method for learning unknown graph topologies from data observations, and then move on to describing a well established graph based algorithm for image segmentation.

A. Learning Laplacian Through Graph Signal Processing

We will begin by defining some concepts and notation related to graph- and signal theory. In case further background on the field of graph theory is required, we refer the reader to the introductory literature on the subject [9].

Let $G(\mathcal{V}, \mathcal{E}, W)$ be an undirected weighted graph with vertices $\mathcal{V}$, edges $\mathcal{E}$ and weighted adjacency matrix W. We define the *Laplacian* L of G as

$$L = D - W, \tag{1}$$

where D is the degree matrix, a diagonal matrix with the total weight of all edges connected to each vertex as entries.

A *graph signal* can be defined as a function $\mathbf{x}$**:** $\mathcal{V} \to \mathbb{R}^n$ which assigns scalar values to each vertex of a graph [3]. This assignment is what distinguishes it from a regular signal, which can be seen as a vector of data as a function of time. This project is however only concerned with graph signals, so the two terms may be used interchangeably.

The *smoothness* of a signal $\mathbf{x}$ on a graph with Laplacian L is commonly measured by its so-called total variation TV [4], defined as

$$TV(\mathbf{x}) = \mathbf{x}^T L\mathbf{x} = \sum_{i \neq j} W_{ij}(x_i - x_j)^2, \tag{2}$$

where a low *TV*(**x**) implies smoothness of the signal. From this definition, we can make the interpretation that a graph signal **x** is smooth on L if strongly connected vertices assume similar values.

As mentioned in the introduction, a suiting graph topology for a system is not always readily available. However, efforts have been made to develop methods which aim to construct fitting graph topologies based on observed signals. In their paper [10], Dong et al. proposes one such method.

Assume a graph $G(\mathcal{V}, \mathcal{E}, W)$ where W is unknown and $\mathcal{V}$ has cardinality n, and let $X \in \mathbb{R}^{n \times p}$ be a matrix with p observed signals. Dong et al. suggest solving the following optimization problem to obtain a Laplacian $L \in \mathbb{R}^{n \times n}$ which favors smoothness of the signals of X:

$$\begin{aligned} \underset{L,Y}{minimize} \quad & ||X - Y||_F^2 + \alpha \operatorname{tr}(Y^T L Y) + \beta ||L||_F^2 \\ \text{subject to} \quad & \operatorname{tr}(L) = n, \\ & L_{ij} = L_{ji} \leq 0 \quad i \neq j, \\ & L \cdot \mathbf{1} = \mathbf{0}, \end{aligned} \tag{3}$$

where $Y \in \mathbb{R}^{n \times p}$ can be viewed as a noiseless version of X, the subscript F denotes the *Frobenius norm* and $\alpha, \beta \in \mathbb{R}^+$. We will give a brief explanation of how the objective function and constraints may be interpreted.

The first term of the objective function imposes a penalty on discrepancy between X and Y, so that Y does not stray too far from the observed signals. The second term amounts to the sum of total variance, as defined in (2), over all the columns of Y. The purpose of this term is thus to promote smoothness of Y over L. Lastly, the third term imposes a penalty on graphs with too much weight concentrated to few edges, promoting an even distribution of weights over the graph.

The first constraint fixes the L^1 norm of L in order to avoid trivial solutions, and the two following constraints make sure the Laplacian maintains its innately defined properties.

For the purpose of image segmentation, assuming one has access to a smooth graph representation of an image, it would be desirable to group together clusters of strongly connected vertices. Fortunately, algorithms for doing so have been available for a long time, one of which we will explore in the following section.

B. Normalized Cut Image Segmentation

In 1997, Jianbo Shi and Jitendra Malik proposed an algorithm for segmenting images based on solving a modified version of the so-called minimum cut problem [7]. In graph theory, a *cut* between two partitions A and B of the vertices of a graph $G(\mathcal{V}, \mathcal{E})$ is defined as

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v), \tag{4}$$

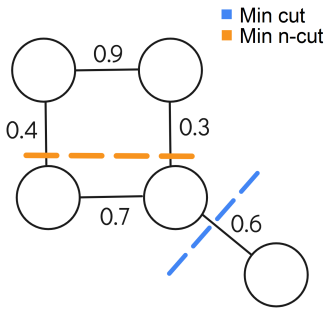


Fig. 1. An example comparing minimum cut and minimum normalized cut in a graph. Here, the minimum normalized cut results in a more intuitive partitioning.

where $w(u, v)$ denotes the weight on the edge connecting vertices u and v. The minimum cut problem consists in finding partitions A and B of $\mathcal{V}$ such that $cut(A, B)$ is minimized.

Solving the minimum cut problem is a way of grouping together vertices of graphs which are connected by heavily weighted edges. As Shi and Malik however point out in their paper, doing so tends to favor small and isolated groupings. To remedy this, they instead propose finding the minimum *normalized cut* for the purpose of vertex clustering, defined as

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, \mathcal{V})} + \frac{cut(A, B)}{assoc(B, \mathcal{V})}, \tag{5}$$

where $assoc(P, V) = \sum_{u \in P, t \in \mathcal{V}} w(u, t)$. The cut between the two partitions $A, B \subset \mathcal{V}$ is here normalized with respect to the weights within each respective partition. An example comparing graph partitioning using the minimum cut and the minimum normalized cut can be seen in Fig. 1.

The authors go on to use this method to segment images, the first step of doing so being to set up a weighted graph $G(\mathcal{V}, \mathcal{E})$ to represent a given image I. Each pixel in I is assigned a corresponding vertex $v \in \mathcal{V}$, with the functions $\mathbf{X}(v)$ and $\mathbf{F}(v)$ respectively describing the position and some feature of this pixel. $\mathbf{F}(v)$ will typically be a measure of color, brightness or some similar aspect of it, but may differ depending on choice and type of image.

Recall that finding the minimum normalized cut of a graph amounts to grouping together strongly connected nodes. In image segmentation, we want to cluster pixels with similar features. For this reason, the weights of G must be a measure of similarity between the features of the pixels in I. Shi and Malik propose the following weight assignment:

$$w(u, v) = e^{-\frac{||\mathbf{F}(u) - \mathbf{F}(v)||_2^2}{\sigma_I}} * \begin{cases} e^{-\frac{||\mathbf{X}(u) - \mathbf{X}(v)||_2^2}{\sigma_X}} & \text{if } ||\mathbf{X}(u) - \mathbf{X}(v)||_2 < r \text{ ,} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where σ_I, σ_X are normalizing constants and r is the maximum allowed distance between two pixels for them to be connected.

Once G is set up, the minimum normalized cut $Ncut_{min}$ is found. Shi and Malik show that by exploiting certain properties of our given problem this can be done in approximately $\mathcal{O}(N^{3/2})$, N being the number of pixels in I.

The obtained $Ncut_{min}$ is compared to a predetermined threshold value, which we will call τ. If it is less than τ, G is divided along the cut and new minimum normalized cuts are calculated for the resulting partitions. This process is then repeated recursively until the threshold is exceeded in every partition. To summarize, the algorithm consists in:

1) Setting up a graph representation G of a given image I.
2) Finding the minimum normalized cut of the current graph partition.
3) Deciding whether the current partition should be subdivided by comparing obtained value to τ.
4) Recursively repeating from step 2 if necessary.

Equipped with this method for vertex clustering, we will investigate if a graph topology obtained from solving the previously discussed problem (3) will serve as a good basis for image segmentation.

III. Graph Construction

We will now examine constructing a suitable graph representation of an image based on the theory of the previous section. The first step of doing so is to adapt the method presented in section II-A to take an image as input. Furthermore, we suggest making a few restrictions to the edges of the obtained graph in order to achieve better results.

A. Adapting Learning Method to Image Segmentation

Given a color image I, we aim to find an undirected graph representation $G(\mathcal{V}, \mathcal{E}, W)$ of it with associated Laplacian L using the method presented in (3). First of all, a word on why doing so may be motivated in the purpose of image segmentation is appropriate.

As previously mentioned, for a signal to be smooth over a graph with Laplacian of fixed L^1- norm it is implied that vertices taking on similar values are connected with greater weights. If different regions of an image are said to be vertices in a graph and features of these regions can be used as signals, smoothness of the signals on the graph would imply strong connections between similar regions. Going back to the section about normalized cuts, this is precisely the quality desired from a graph representation of I. We thus wish to cast the problem of finding such a representation as one of solving (3).

The first part of doing so is to interpret I as one or multiple graph signals. To begin with, we will let each pixel of I be represented by a vertex in $\mathcal{V}$, although an adjustment to this will be made later on. An evident choice for graph signals describing the features of these pixels would be their color. Hence, we choose to interpret I as three different signals: the red, green and blue values for each pixel.

Relating this to (3), X becomes a three column matrix representation of RGB values obtained directly from the image file. As mentioned in section II-A, Y can be viewed as a version of the signal matrix X free from noise. Because of how the entries of X are collected, we will however assume the signals to be noiseless from the beginning. Under this assumption the term $||X - Y||_F^2$ in the objective function can be set to zero, simplifying (3) to

$$\begin{aligned} \underset{L}{min} \quad & \alpha \,\mathrm{tr}(X^T L X) + \beta ||L||_F^2 \\ \text{s.t.} \quad & \mathrm{tr}(L) = n, \\ & L_{ij} = L_{ji} \leq 0 \quad i \neq j, \\ & L \cdot \mathbf{1} = \mathbf{0}, \end{aligned} \tag{7}$$

The authors of [10] suggest using operator splitting methods such as ADMM [11] for solving the above problem for graphs with a large number of vertices. As we however deemed developing such a method to be beyond the scope of this project, we settled for using CVX, a general solver of convex optimization problems for MATLAB [12]. CVX simply takes a completely formulated problem as input and finds a solution. This is obviously advantageous in a perspective of convenience but comes at the cost of being significantly slower, which motivates a discussion of how problem (7) scales with the size of input.

The Laplacian of a graph with N vertices is an $N \times N$ matrix. Since it is symmetric and the diagonal entries are dependent on the row and column they belong to, it contains $\frac{N(N-1)}{2}$ independent entries. In [7] Shi and Malik suggest setting up the graph representation of I by assigning a vertex to each pixel, which we also assumed to do in the beginning of this section. Doing so would mean that finding a Laplacian for a 100×100 pixel image as described above implies solving an optimization problem in approximately $5 \cdot 10^7$ variables. For this reason, we choose to limit input size by clustering images into so-called *superpixels*.

Clustering an image into superpixels means grouping its pixels into perceptually meaningful regions. Such clusterings in combination with the normalized cut algorithm have been proposed before with the aim to achieve faster segmentations [13]. We argue that segmenting a group of superpixels equally well would demonstrate the usefulness of our method as segmenting each individual pixel of an image, while significantly decreasing the size of input to the optimization problem.

For our implementation, we use the SLIC (*simple linear iterative clustering*) algorithm to do this [14]. The algorithm takes as input an image and an integer k, which is the desired approximate number of superpixels. As output, a grouping of the image into superpixels is given. We assign to each superpixel $S \subset I$ a color $C(S) \in \mathbb{N}^3$, with

$$C_i(S) = \Big\lfloor \frac{1}{|S|} \sum_{p \in S} C_i(p) \Big\rfloor, \tag{8}$$

where $C(p) \in \mathbb{N}^3$ denotes the RGB values for a pixel p. Above clustering and coloring is done using the scikit-image library for Python [15]. An example is displayed in Fig. 2.

In conclusion, we will use superpixels as vertices in G instead of pixels, with superpixel $S(v)$ corresponding to vertex $v \in \mathcal{V}$. The matrix of $C(S)$ for all S becomes the input for (7), which means that we can control input size with the variable k. The SLIC algorithm runs in $\mathcal{O}(N)$ where N is the number

Fig. 2. Clustering of an image into 115 superpixels using the SLIC algorithm. Each superpixel is assigned the average color of the pixels contained within it.

of pixels in the image, and run time is thus independent of this k.

The learning method has now been adapted to learn a graph representation of an image. The next section will discuss some further adjustments that need to be made to the resulting graph.

B. Edge Adjustments

So far the only feature that has been taken in to account for linking superpixels together is color, but another important aspect is their position. In the obtained Laplacian, the weight of an edge connecting two vertices is independent of whether these are located next to each other or on opposite sides of the image. We however want our segments to be locally connected for the segmentation to be meaningful.

One could imagine managing this by somehow including the position of superpixels as graph signals, but doing so would give us the problems of determining position inputs for wide regions as well as having to compare these to colors. For this reason, we simply choose to impose some restrictions on our edges $\mathcal{E}$. More specifically, we only include edges between vertices with corresponding superpixels adjacent to each other, or in mathematical notation

$$\mathcal{E}_a = \Big\{(v_1 \times v_2), v_i \in \mathcal{V} \,\Big|\, \exists\, p_1, p_2 \in S(v_1), S(v_2) \Rightarrow ||\mathbf{X}(p_1) - \mathbf{X}(p_2)||_2 = 1\Big\}, \tag{9}$$

where $\mathbf{X}(p) \in (x, y)$ denotes the position of a pixel $p \in I$.

Another matter to take into account when it comes to edges are self-loops. In the classic graph construction for the normalized cut algorithm presented in (6), these are included and may play a role in the resulting segmentation.

For Laplacians, we have that

$$L_{ii} = \sum_{j=1}^{n} w_{ij} - w_{ii} = \sum_{j \neq i} w_{ij}, \tag{10}$$

so diagonal entries do by definition not include any information about the weighting of self-loops. We are thus free to define these arbitrarily for G.

A consequence of (6) is that weight on edges of identical vertices, in other words self-loops, all equal 1. With our graph construction there are however no such obvious values we can assign these. Looking at the definition of the normalized cut in (5), self-loops contribute to $assoc(A, \mathcal{V})$ and $assoc(B, \mathcal{V})$ but do not affect $cut(A, B)$. Consequently, self-loops with higher weights promote more evenly sized segments while possibly downplaying perceptual discrepancy between regions.

For our experiments, we define

$$\mathcal{E}_s = \Big\{(v \times v), v \in \mathcal{V}\Big\}, \tag{11}$$

set $\mathcal{E} = \mathcal{E}_a \cup \mathcal{E}_s$ and assign

$$w(v, v) = \sqrt{N} \qquad \forall v \in \mathcal{V}. \tag{12}$$

Although this choice is somewhat arbitrary, it will serve for our purposes.

In conclusion, we obtain a graph representation $G(\mathcal{V}, \mathcal{E}, W)$ of I by:

1) Segmenting I into N superpixels and representing these as vertices in $\mathcal{V}$.
2) Obtaining a Laplacian L by solving (7) with RGB values as input.
3) Setting $\mathcal{E} = \mathcal{E}_a \cup \mathcal{E}_s$ with $\mathcal{E}_a$ and $\mathcal{E}_s$ as described in (9) and (11).
4) For each edge $(u, v) \in \mathcal{E}$ assigning weights

$$w_{uv} = \begin{cases} -L_{uv} & \text{if } u \neq v \\ \sqrt{N} & \text{if } u = v \end{cases}. \tag{13}$$

We are now ready to evaluate our graph construction in the context of image segmentation, which will be done by running experiments as described in the following section.

IV. Experiments

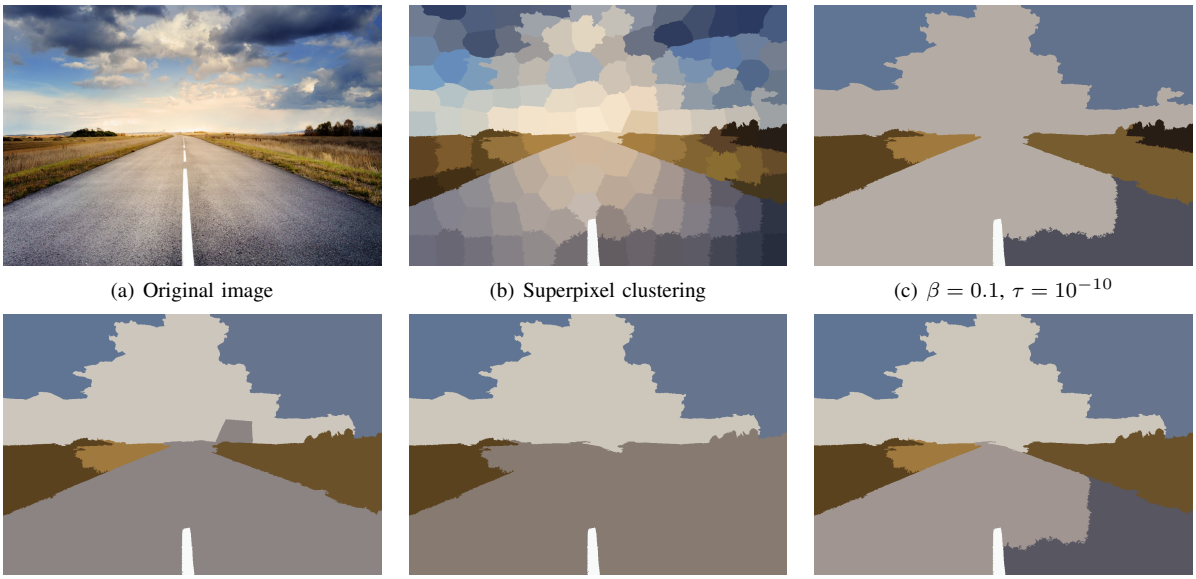
Experiments are run by performing a normalized cut segmentation on graphs generated from images with varying parameters α and β. Since the objective function of (7) only contains two terms, we can evaluate the impact of these on the results by fixing one of the parameters and varying the other. For simplicity we therefore set $\alpha = 1$ for all experiments.

We will run our experiments on two different images: one of a bouquet of flowers with regions clearly distinguishable by color (see Fig. 2), and one of a road, which could represent a real-world application like the one described in the introduction [6]. In this second case, we ideally want to be able to distinguish the road from its surroundings.

Another important parameter to take into account is the threshold value τ for the Ncut-algorithm, as this notably impacts the nature of the segmentations. For the segmentation of the first image we fix τ to a suitable value (one that we deem yields reasonable segmentations for our parameter choices) in order to examine how a varied β generally impacts absolute values of normalized cuts. For the image of the road, we

(a) Superpixel clustering (b) $\beta = 1$ (c) $\beta = 0.1$ (d) $\beta = 0.01$ (e) $\beta = 0.001$

Fig. 3. Results from normalized cut segmentation of the image seen in Fig. 2 based on learned graphs with varying parameter β. For all segmentations $\tau = 3 \cdot 10^{-14}$ and $\alpha = 1$.



(a) Original image (b) Superpixel clustering (c) $\beta = 0.1$, $\tau = 10^{-10}$

(d) $\beta = 0.01$, $\tau = 2 \cdot 10^{-9}$ (e) $\beta = 0.001$, $\tau = 2 \cdot 10^{-15}$ (f) $\beta = 0.001$, $\tau = 3 \cdot 10^{-15}$

Fig. 4. Results from normalized cut segmentation of image of a road based on learned graphs with varying parameters. For all results, $\alpha = 1$. Original image and its clustering into superpixels are shown in (a) and (b), while (c)-(f) showcase the segmentations.

instead vary τ with each different input so as to achieve the best segmentation possible.

Finally, we would like to compare how our method of graph construction measures up to the classic one proposed by Shi and Malik. We do so by assigning weights between nodes as described in (6), but excluding the second case-term and restricting edges as done in section III-B. Segmentations of each method are then compared. Here too the scikit-image library is utilized for obtaining all of our results [15]. The code used can be found in appendix A.

Results from segmentation of the first image based on our proposed graph construction are shown in Fig. 3 along with the image of superpixels. Each segment takes on the average color of the superpixels contained within it. Largely, perceptually significant areas of the original picture are captured. We note that for a fixed threshold τ, a decreasing β leads to finer segmentations, to the degree that the one resulting from $\beta = 0.001$ is essentially meaningless.

Moving on to Fig. 4, results are shown from segmentations of the second image. Here too we see that overall features of the image could be captured, even though color discrepancies between regions were less defined. As mentioned before, an ideal segmentation of this image would be one that could identify the road as an individual object, so it is by this standard we evaluate the results.

For $\beta = 0.1$, the algorithm identifies the left and middle part of the road to be more similar to the sky than to the right part of the road. $\beta = 0.01$ gives the best result of the values we tested for, successfully identifying the entire road with the exception of the white stripe in the middle of it. For $\beta = 0.001$, it was difficult to find a proper τ to give an optimal partitioning of the image. Either the lower right part of the road is seen as a separate object, or virtually the entire lower part of the image is.

The comparison to the classic method for graph construction can be seen in Fig. 5. For the image of the flowers we found that almost identical results could be obtained using both methods with a correct calibration of τ. More interesting were the results for the image of the road. It turns out that the construction proposed by Shi and Malik has the same problem as was presented for our learned graph with $\beta = 0.1$, namely that it deems the middle part of the road to be more similar to

(a) $\beta = 0.1$, $\tau = 3 \cdot 10^{-14}$ (b) $\tau = 2 \cdot 10^{-18}$ (c) $\beta = 0.01$, $\tau = 2 \cdot 10^{-9}$ (d) $\tau = 4 \cdot 10^{-4}$

Fig. 5. Segmentation of images using (a), (c): a learned Laplacian with $\alpha = 1$, and (b), (d): the classic method from (6) with $\sigma_I = 255$.

the sky than to the right part. In this example, the new method for graph construction seems to have a possible advantage with the correct choice of parameters. It should however be mentioned that generating a Laplacian by solving (7) as by our implementation can take up to several minutes, whereas calculating weights through (6) is done in less than a second.

V. Discussion

In discussing our findings, we have decided to mainly focus on three aspects. These are the quality of the results, the time complexity for the method used and what one might want to focus on in future studies related to the area.

A. Quality of Segmentation

Defining the quality of segmentation is in itself an area that would require its own study. The segmentations in this work were done using arbitrary choices of τ, selected so that the results looked good in our opinion. This is not a guarantee of correctness for any particular application. Further, it holds that multiple selections of segmentation fineness can yield reasonable results. In Fig. 2 it is possible to see each individual flower as a segment, but also each petal within each flower. Within Fig. 4 it is possible to choose as we did to segment the road as its own segment, but you could as well want each lane. Whichever is more correct is up to the application, and it is not certain that this method is suitable for all levels of segmentation.

B. Time complexity

Maybe the most evident disadvantage of this approach to image segmentation is that it, at least with our implementation, is very slow. In applications where you need to process images in real time, such as in the example with a self-driving car photographing the road, this could be a crucial problem. Even for applications where fast segmentations not are essential, users may still prefer other methods out of convenience.

An important reason to why our implementation has this issue is that it uses CVX to solve a rather large problem, even with the down-scaling done by the SLIC algorithm. CVX is made for solving multiple types of convex optimization problems, and for this reason may not exploit all properties of a specific one. Because of the ”black box” nature of the program, it is hard to discuss an exact time complexity without extensive knowledge of the source code. The authors of CVX, however, explicitly express in the user guide that it is not meant to be used for large-scale problems [16], although we out of convenience have disregarded this advice in our project.

With this in mind, we judge that (7) could be solved significantly faster. As alluded to, an important part of this is the choice of a suitable solving method. Since the problem can be cast as quadratic one in $\frac{N(N-1)}{2}$ variables, N being the number of vertices, developing an ADMM solver as suggested by Dong et al. would likely be a good approach to improving performance. Exploiting the fact that we only make use of variables corresponding to edges between adjacent superpixels (section III-B), the number of variables that need to be solved for could be decreased from the current $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$.

C. Future Work

For anyone who wants to further improve on the findings of this project, we think a good first step would be to find a faster method for solving (7) by developing a specific algorithm and trying to limit the size of the problem. One should note that this would not only be useful for faster image segmentation, but in the entire field of data based graph learning when one is interested in mapping large networks.

It would also be of interest to evaluate the results of segmentations where the input signal is noisy and compare to established methods. Such research could have implications for applications where images are recorded in real time and interpreted directly as signals.

Finally, to the more general topic of image segmentation, it could be useful to investigate what constitutes a desirable result in specific situations and to develop a more rigorous framework for determining this than what is used in this project. For many applications, doing so may be essential to benefit from any existing technique for segmenting images.

VI. Conclusions

In this project, we have presented an approach for constructing graph representations of images using methods from the field of GSP. We then proceeded to segment the images based on yielded graphs using the classic normalized cut algorithm.

The resulting segmentations show promise in terms of quality, although the process of generating graph structures

was significantly slower than the standard method of direct assignment. We do however see great potential for mitigating this inconvenience, and hope our results may inspire further investigation into the areas of graph learning and image segmentation.

APPENDIX A

Code used for image segmentation.

ACKNOWLEDGMENT

The authors would like to thank Magnus Jansson for supervision through out the project, Joakim Lilliesköld for valuable input during the planning of it and Anna Herland for help with technicalities related to the writing of the report.

REFERENCES

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[3] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.

[4] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[5] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 484–499, 2017.

[6] A. Guin, "Terrain classification to find drivable surfaces using deep neural networks: Semantic segmentation for unstructured roads combined with the use of gabor filters to determine drivable regions trained on a small dataset," Master Thesis Report, KTH, Stockholm, Skolan för elektroteknik och datavetenskap (EECS), Robotik, perception och lärande, RPL, 2018.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[8] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[9] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[10] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[12] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[13] C. Zhao, "Image segmentation based on fast normalized cut," *The Open Cybernetics & Systemics Journal*, vol. 9, no. 1, 2015.

[14] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[15] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: https://doi.org/10.7717/peerj.453

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Cvx users' guide - introduction. Stanford University, Palo Alto, CA, USA. Accessed: (2020, Apr). [Online]. Available: http://cvxr.com/cvx/doc/intro.html#what-cvx-is-not