An Implementation: Laplacian Coordinates for Image Segmentation Method

Paola Ardon
ardonp@hotmail.com

Seperh Mohaimanian
Voidminded@gmail.com

Èric Pairet
ericpairet@gmail.com

Abstract— Nowadays, computer vision systems are becoming more important than ever. Such science is being used from continuous quality control in the industries to the most sophisticated robots, going through medical applications. Despite being used almost everywhere, it is a science under development. It faces many drawbacks, such as noise acquisition, hardware limitations, the well-known serial port bottleneck and long computational times, among others. Some of these drawbacks can be solved or, at least, improved by post-processing techniques. Under this category, many techniques can be considered. However, the most remarkable because of its high usage are the denoising and the segmentation methods. Specifically, the last term is going to be the core of the present paper.

Index Terms—Computer vision, image segmentation, object segmentation, project management, application software, software systems, software testing, open source software, benchmark testing.

1 Introduction

IMAGE segmentation is one of the main tools for computer vision nowadays. It is the process of partitioning a digital image into pixels, also commonly known as superpixels [1]. This technique is used in many fields, such as traffic control, object detection and one of the most important, medical imaging. All of them need a post imaging process to get valuable information, and this is what image segmentation offers; a simplification or exchange of the initial input image into information that can later be useful by the final application.

There are many image segmentation techniques. Among the former and most popular segmentation methods we have: thresholding, clustering and edge detection methods, as well as others. A more recent method that has gained popularity in the field of image segmentation and computer vision is seeded image segmentation. Seeded segmentation algorithms are based on treating the image as a weighted graph and minimize the energy function on the graph to produce segmented regions [2].

Laplacian Coordinates is one of the algorithms in this category proposed by Casaca, Nonato and Taubin in their paper *Laplacian Coordinates for Seeded Image Segmentation*. By using Laplacian Coordinates they assure a better-segmented image with a higher "quantitative quality" and better "visual results" by applying a simple mathematical approach that is easy to implement [3].

The scope of this paper is to implement, thoroughly evaluate and explore the Laplacian Coordinates algorithm proposed in the previous mentioned research paper. The knowledge required for understanding the main concept of Laplacian Coordinates is introduced in Section 2. Section 3 details the project organization for the implementation. The software is illustrated in Section 4. Further trials and improvements done over the developed application are explained in Section 5. The evaluation of the system is done by following standard experiments, these are reflected and analyzed in Section 6. Finally, some final remarks and future work are presented in Section 7.

2 Definitions and problem statement

This section explores the basis of the Laplacian Coordinates method. Understanding these concepts will lead to choosing the proper procedure for the implementation of the algorithm. By the end of the process, we will be able to analyze and validate the benefits and limitation of the Laplacian coordinates method.

2.1 Background

The Laplacian Coordinates method is based on the minimization of the functional energy on the seeds of an image [3]. The authors present their algorithm as one of the most efficient and outstanding methods among others in seeded image segmentation. Some of the algorithm advantages are:

- The minimization of the average distances among pixels which gives a better-segmented image. The Laplacian Coordinates approach minimizes the average distances while controlling assigning incorrect labels to pixels during the segmentation task.
- Consequently, similar pixels stay closer together creating clear boundaries among the different pixels in the neighborhood.
- Easy to use code. The simplicity of the algorithm can be summarized in four main steps: affinity graph building, definition of seeds, energy functional construction and assignment of labels.

Let us consider I the input image of size $m \times n$ with an amount of pixels denoted as I_i. Then, the relation kept between any two adjacent pixels P_i and P_j can be described by the weight w_{ij}. To compute this weight, we take into account the intensity property of the pixels, I_i and I_j. Equation 1 shows the weight equation in its simplest form.

$$w_{ij} = e^{-\frac{\beta||I_i - I_j||_\infty^2}{\sigma}}, \tag{1}$$

where, β is a tuning constant and σ standardizes the computed difference between intensities according to the maximum difference of I_i and I_j of all the edge.

At this point, the most important idea of the Laplacian Coordinates has already been explained. The interest of computing the weight between two pixels is to evaluate its similarity, which is the main concept for minimizing the energy between the pixels and the set of given seeds.

Considering the weights of a pixel P_i with respect to all its 8 neighbors that share an edge with pixel P_j, the weighted valency d_i of the pixel P_i is computed as indicated in Eq. 2.

$$d_i = \sum_{j \in N_i} w_{ij}, \tag{2}$$

where, N_i is the set of 8 neighbours P_j of the pixel P_i.

These two previous computations of W_{ij} and D_{ij} lead us to compute the Laplacian matrix as $L = D - W$. This matrix characterizes the relation between all the pixels of the image I. In other words, the matrix L represents the graph of the image I.

In addition to all of these characterizations of the input image I, the Laplacian Coordinates method takes into account the external set of given seeds, which can be considered from the foreground or the background, x_f and x_b respectively. Once the seeds have been stipulated, the energy minimization is formulated as in Eq. 3.

$$\begin{aligned} E(x) =& k_1 \sum_{i \in B} ||x_i - x_B||_2^2 + k_2 \sum_{i \in F} ||x_i - x_F||_2^2 \\ &+ k_3 \sum_{i \in V} ||d_i x_i - \sum_{j \in N(i)} w_{ij} x_{ij}||_2^2, \end{aligned} \tag{3}$$

where B and F are the set of seeds belonging to the background and foreground respectively, V is the set of nodes belonging to P_i and the variables k_i for $i = 1, 2$ and 3 are constants.

According to Eq. 3, x_i wants to be defined as part of the foreground or the background. Therefore, once this function is ready to be minimized and considering the proposed approximations in the paper, Eq. 3 becomes the simple and nice system shown in Eq. 4.

$$(I_s + L^2)x = b, \tag{4}$$

where, I_s is a diagonal matrix that contains 1 if its element $I_{s_{ii}}$ belongs to either the foreground or the background, and b is a vector of size $m \times n$ that is set to a value or another according to the belonging of its element b_i to the foreground or the background.

Once we get x, which is of size $m \times n$, it has to be thresholded and reshaped in order to get a binary mask. Finally, to obtain the segmented seeded region as foreground, this mask has to be applied onto the original image I.

The previous brief explanation contains all the key points to be considered for the implementation scope of this paper. Additionally, from the described equations, it is easily seen that the method can be implemented in two ways [3]:

- Simple segmentation, which comprises only two regions, one set of seeds representing background and another set representing foreground region.

- Multi-region segmentation, for this application many regions, can be segmented in the image by using multiple sets of seeds.

Considering the possible applications, the flow of the implementation and its structure are described in Section 2.2.

2.2 Implementation Approach

As any image segmentation method, the Laplacian coordinates works with a set of pixels that can be better described in matrix form. For the scope of this project, we are to implement the algorithm on a user-friendly application developed in C++ programming language. The code is written using Qt Framework in conjunction with template libraries such as OpenCV [4] and Eigen [5] to help the management and process of images and matrices respectively.

It is not enough to develop an application that carries the implemented algorithm. The main purpose is to build it in a way that is efficient and highly structured. This is achieved by the combination of predefined libraries and by using the advantages of C++ programming structure such as friendship and inheritance among functions or simple inline functions that improve the execution time of a task.

Since the basic algorithm structure could be used for further segmentation purposes, we need to assure every function that composes the base of the implementation works properly. This testing/debugging task is performed by the employment of Google Test tool (GTest). GTest is a testing framework library that facilitates on the debugging stage of the code. Since it is a very thorough testing process, it helps to avoid surprises such as "bugs" or "glitches" when putting the different functions together.

Moreover, as this code will not only be for a personal usage but also available for third parties, a proper comment structure is needed. Therefore, we made use of Doxygen tool to help in the code documentation and flow diagram structure. One of the greatest advantages of this tool is that it creates a user manual of the code in case the user needs to "walk" through the implementation.

With these tools in hand, our approach was first to perform a basic implementation of the algorithm described in Section 2.1. This implied the construction of a GUI (Graphical User Interface) that allows the user to interact with the source code. Once the basic segmentation code was stable, we tried to go further than the proposed in the research paper. However, one of the difficulties remains on keeping the documentation updated according to the changes. Therefore the importance of having a proper project management organization.

3 Project Management

At the very beginning and taking advantage of the preliminary study of the paper, we programmed a schedule in order to achieve the completeness of the project. The fixed project deadlines take into account the ones established by the course schedule. Figure 1 shows the Gantt chart for the organization of the project.

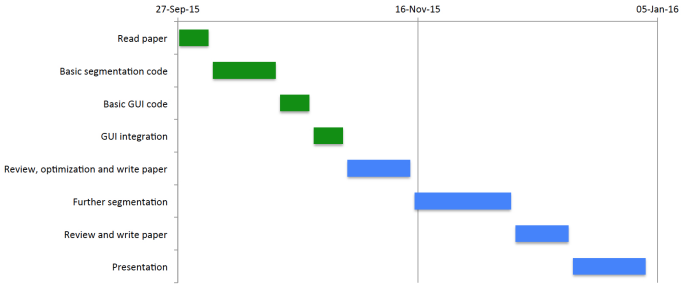

Fig. 1: Gantt chart about the expected timing for the realization of the project.

This Gantt chart sketches the main tasks we stated to be done by the end of the semester. Additionally, Table 1 shows the detailed information on dates and duration of each of the tasks of the project.

TABLE 1: Expected timing for the realization of the project.

Task name	Start	End	Duration (days)
Read paper	28/09/15	04/10/15	7
Basic segmentation code	05/10/15	18/10/15	14
Basic GUI code	19/10/15	25/10/15	7
GUI integration	26/11/15	01/11/15	7
Review, optimization and write paper	02/11/15	15/11/15	14
Further segmentation	16/11/15	06/12/15	21
Review and write paper	07/12/15	18/12/15	12
Presentation	19/12/15	03/01/16	16

As can be seen, we structured our project in 9 big tasks, all of them sequentially timed. In general, the tasks are sorted from the basics to the more complicated steps. The first stage it is the understanding of the reference paper hand the initial basic implementation. It is worth noticing that in our schedule we reserved two weeks for doing the first draft of the documentation of the project.

Alongside the original defined goals for the implementation, some extra tools were asked to be integrated to the code. This variant integration tools made us difficult to stick to the original planned schedule. As a result, the original schedule was modified to the one represented in Table 2. It is important to notice that due to our margin of error consideration in the design of the first schedule all original 9 tasks were successfully accomplished.

TABLE 2: Final scheduling of the project.

Task name	Start	End	Duration (days)
Read paper	28/09/15	08/10/15	11
Basic segmentation code	09/10/15	18/10/15	10
Basic GUI code	19/10/15	25/10/15	7
GUI integration	26/11/15	03/11/15	9
Review, optimization and write paper	04/11/15	10/11/15	7
Further segmentation	11/11/15	27/12/15	47
Review and write paper	28/12/15	10/01/16	14
Presentation	28/01/16	10/01/16	14

As it can be observed, the main differences between the initially expected timing represented in Table 1 and the final one represented in Table 2 are the followings:

- We spent more time on understanding the Laplacian Coordinates method. However, this was definitely a worthy task since we could implement the first simple segmentation method faster.
- The first review of the project, which includes optimization of the code and initial draft of its documentation, took much less than expected. This as a consequence of the continuous improvement that is done while programming.
- The weakest point of our project management has been on further segmentation. First, we analyzed the mathematical fact described in Section 5.1 and performed the multi-region segmentation explained in Section 5.2. However, our idea was to implement real-time segmentation, where we spent so much time without any successful result.
- In order to compensate for the extra time spent in the previous task, the final review of the project and its final documentation were performed in parallel.

For the sake of the organization of the project, as a team, we employed online project management tools. On one hand, we used Trello for organizing tasks and deadlines. Trello also presents a good platform for brainstorming and discussion as well as sharing important links.

All the code of the project was shared online using a GitHub repository. Github not only allows all the members of the group to follow the implementation but also to have access to the different versions of the code. The repository not only stored the implemented code but also other important reference files.

Finally, in order to allow all the members of the group to edit the documentation of the project, we used Overleaf, which is an online LaTeX editor. Despite its limitations, this tool has been really useful when writing group documents.

4 System Implementation

Following the previously explained algorithm and the scheduled tasks in Section 3, we implemented the system application with mainly three parts:

- Segmentation code
- GUI
- GTest

The developed application is designed to properly work on Windows, Linux and OS X Operating Systems (OS). The compatibility of the system is designed in the CIS.pro file of the project folder. Some Eigen, OpenCV and GTest libraries are required for the proper functionality of he code. Detailed instructions about the installation process can be found in the GitHub repository */ericpairet/Software-Engineering-Project*.

In addition to the three main coding parts of the system, we integrated CMake platform in order to properly compile the code. The following

subsections are an overview of the content of each component. Further details about the structure of the code can be found in *http://voidminded.com/sbis/*.

4.1 Segmentation Code

The code in charge of the image segmentation is basically the application of the described equations in Section 2.1.

For implementation purposes, the value of σ is set to 0.1 by the authors recommendation. The tuning value of β can be chosen by the user through the GUI. Otherwise, the tuning default value is set to 0.005 due to the good results in the output segmented image after experimentation. Additionally, the background pixels value x_b is set to 1 and the foreground pixels value x_f is set to 0.

4.2 Graphical User Interface

The purpose of the user interface design is to be as user-friendly as possible. In addition to its practical layout, it gives the user valuable information on the segmentation process. This is done by printing the status of the input image in text form in the GUI while going through segmentation code. Some of the features in the user interface are:

- Tab generation: it generates individual tabs in the monitor application in order to display the selected input image and the segmented output regions. This allows the user to have a clear view of the results.
- Beta slider: it allows the user to select a value between 0 and 1 for the tuning value of β. This range was selected after intensive experimentation, where the result of the segmentation with a $\beta > 1$ were not satisfactory.
- Size of coloring pen: it allows the user to select the thickness of the seeds to be painted on the input image.
- Seed selection: it allows the user to select as many segmentation regions as desired as well as the different color of the seeds.
- Browsing image button: it allows the user to select the input image to be segmented.
- Execute segmentation button: sends the input image and the selected seeds to the segmentation code to be processed.
- Clearing data button: in case the user wants to segment another image, he/she is able to delete the previous data such as the seeds so that the new image can be processed.

These tools allow the user to perform a segmentation in four basic steps. First, once executed the program performed in C++, the main window of the application shown in Fig. 2 appears on the screen.

Fig. 2: Main appearance of the application

The next step consists on uploading a picture. Therefore, the "Browse" button has to be pressed. After selecting the image to segment on the pop-up files browser, such image is shown in the tab called "Original". The aspect of the application at this point looks like the Fig. 3.

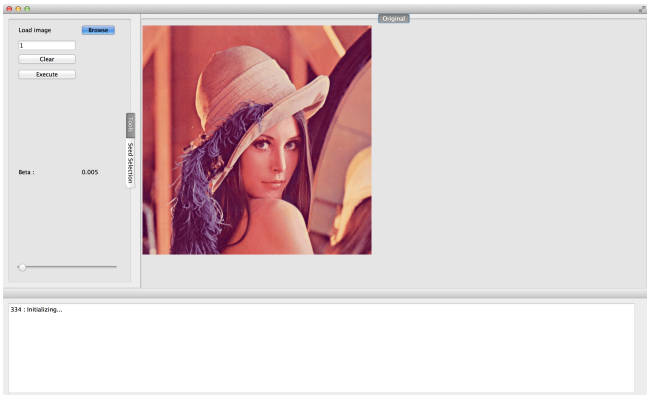

Fig. 3: Imported image to segment

At this point, the user has to select the region to segment. For this finality, in the "Seed selection" tab the user is provided with the required tools. By pressing once the "Add seed!" button, the seeds for the background can be specified. After that, at any time that this last button is pressed, it adds the possibility to add a new set of seeds as a foreground.

The color of the seeds can be selected within a large range of colors. The aim of this feature is to let the user select a color that will stand up enough over the region. To locate the seeds, the user only has to move the pointer above the image. After this procedure, the application looks like Fig. 4.

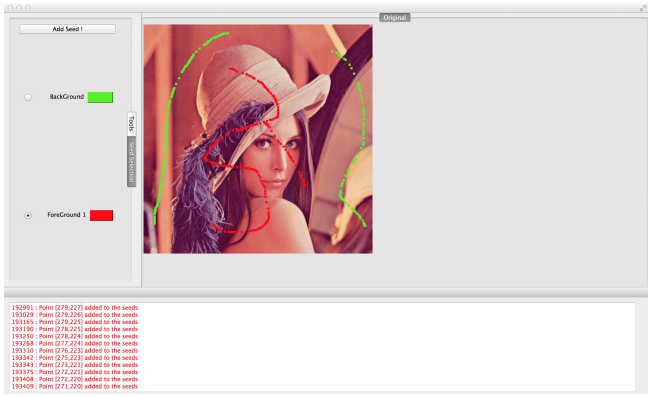

Fig. 4: Specification of the seeds

Finally, the user only has to press the button "Execute" found in the "Tools" tab in order to obtain the segmented image. During the computation, information about the state of the segmentation is reported in the terminal located in the lower part of the application. The output image is shown in a new superior tab, as can be seen in Fig. 5.er

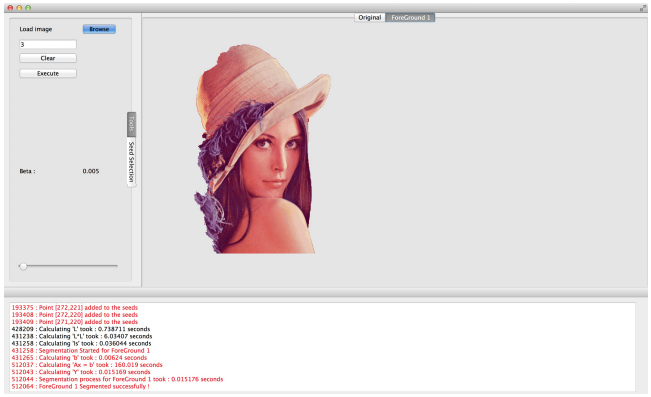

Fig. 5: Resulting segmentation

Based on the previously explained code for segmentation and GUI, the GTest and the CMake code were developed.

4.3 GTest

Google test is a unit testing library specially designed for C++ programming language [6]. It is a very useful tool that helps to achieve a thorough debugging process to avoid inconsistencies on the code.

In our project, GTest is used to test the 5 functions that compose the segmentation code. Specifically, when performing a GTest, the system reads a predefined image and segments it. Meanwhile, GTest checks if the outputs of all the functions are like a predefined hash value. If not, the GTest reports the test as failed.

Actually, what is saved for performing the GTest are the expected matrices at the output of each function. Therefore, the expected hash value is computed from there, avoiding to check all the elements in a matrix. All these files are included as part of the code of the application. Even though the input image is hardcoded, it can be changed for another one. Consequently, the files saving the expected matrices have to be changed correspondingly.

4.4 CMake

The implemented code was developed using *qmake* as the system compiler. However, in order to improve the architecture of the implementation, the code also could be built using *cmake*. CMake is a cross-platform that allows the building of a directory tree outside the source tree [7]. This warranties that if a directory is removed its source files remain untouched. The objective of CMake integration is to learn the benefits of using a platform able to recognize the compilers needed for a given type of source and avoid the pain to run a code manually.

Once we implemented the useful tools of CMake and GTest on our base code, we proceeded with the implementation of extra features.

5 Additional Features on Implementation

The current performance of the system achieves the desired quality levels. Nonetheless, additional improvements were made on the application. Enhancements on the code such as methods to compare math calculations efficiency vs accuracy and the implementation of multiple-region segmentation were done in the application.

5.1 Trade-Off - Speed vs Accuracy

While we were checking the maths behind the implementation of Laplacian Coordinates for Seeded Image Segmentation [3], we realized the insignificant numerical difference between the matrix L described in Section 2.1 and its squared version L^2 used to solve Eq. 4. Although this difference could be considered negligible, it makes a huge difference on the computation time when solving the system slightly affecting the segmentation.

Considering the results, we propose a small different approach to the classical method of the Laplacian Coordinates. This new approach is much faster and at the same time gives a solution is not far from being equal to the obtained by the classical

approach. Consequently, a trade-off between speed and accuracy should be considered before segmenting an image.

The matrix L is characterized for being symmetric and really sparse [3]. Actually, as sketched in Fig. 6, the numbers that differ from zero make 3 diagonal lines. Each one of these diagonals has a symbolic width of 3. Therefore, almost each column of the matrix L has only 9 values different from zero.

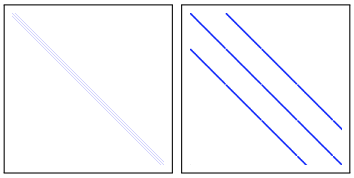

Fig. 6: Representation of the sparsity of L. a) Full L. b) Enlarged patch of L.

Moreover, the main diagonal of the matrix L contains large values in comparison to the rest of diagonals. This is due to the computation of L, which is equal to the subtraction of the matrix W on the matrix D [3].

When computing L^2, the number of symbolic diagonal lines and its width increased to 5, obtaining the drawing of the matrix L^2 shown in Fig. 7. Now, the values in the main diagonal of L^2 are still greater than the values in the not-new diagonals. Furthermore, the values of the new diagonals are tiny in comparison to the rest.

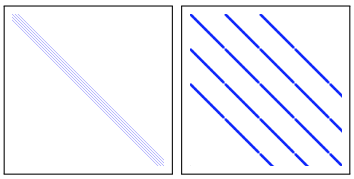

Fig. 7: Representation of the sparsity of L^2. a) Full L^2. b) Enlarged patch of L^2.

This observation suggested us that omitting the use of L^2 when solving the system represented in Eq. 4 should not be of great impact for the final output. Consequently, by changing to L would not only avoid a reduction of sparsity by a factor of 2.78, but it will also achieve an important reduction of the usage of memory and execution time. A proper analysis has been reported in Section 6 in order to evaluate this idea.

5.2 Multiple-Region Segmentation

The original scope of the project was to implement the basic segmentation proposed in the paper, which is based in two regions: one set of seeds for the background and another one for the foreground. However, in order to improve the user's experience and expand our knowledge on the segmentation method we also applied multiple-regions segmentation. On the research paper, in order to achieve multiple segmentation they treat Eq. 4 as an extension of solving N linear systems. Then, the original setting becomes Eq. 5

$$(I_s + L^2)x^{(j)} = b^{(j)}. \quad (5)$$

Note that there is not need to compute the matrices I_s and L for each iteration, as they are characterized by the existence of some seeds and the image, respectively. However, the vector b has to be redefined for each segmentation j with the proper label $K_j \in K = K_1, K_2, ... K_N$. Following this approach, on the implemented code we treat every new selected region by the user as a new foreground and the rest as background, not a matter of how many N new regions are chosen. As briefly introduced before in the GUI explanation, the user is able to select a new region to segment by pressing "Add Seed!" button.

The results of multiple-region segmentation are shown in Section 6 where among the segmentation results other qualitative and quantitative values are analyzed.

6 Results and Evaluation

In this section, we provide the obtained results with all the approaches and further improvements discussed in this paper. The results and their analysis are going to be presented in the following manner:

- First, the classical approach described in Casaca et al. will be compared with the proposed one in Section 5.1.
- Secondly, the multiple-region segmentation results explained in Section 5.2 will be shown in a detailed manner.

All these results have been obtained by setting the tuning parameters σ and β to fixed values. On one hand, the value of σ was set to 0.1 as proposed in the research paper. On the other hand, the value of β was experimentally found to perform well in a wide range of cases when it was set at 0.005.

The database "Grabcut" has been used in order to perform a fair comparison between approaches [8]. This database provides some sample images alongside with their ground-truth and marked foreground and background regions (seeded maps). For all the images, the ones classified as bunch, ceramic, grave, llama, person and sheep have been used. All of them represent the most common difficulties when performing segmentation: noncontinuous and thin regions; and shadows and intensity similarity between foreground and background regions.

Figure 8 shows the results obtained when segmenting the proposed images with the standardized seeds, the results of applying the two methods reviewed in this paper, and the provided ground-truth segmentation.

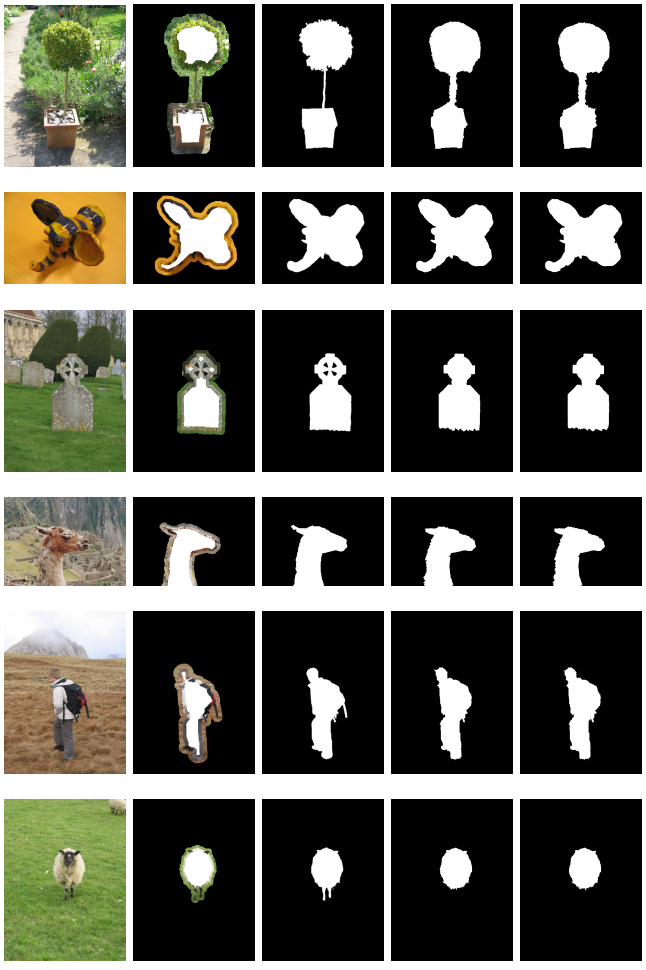

Fig. 8: From left to right: original image, the tri-map seeds (black is background, white is foreground and the rest is the unknown region), ground-truth, obtained result considering L^2 and obtained result considering L.

By simply observing the images, some conclusions about the segmentation performance can be done. However, this does not give a quantitative evaluation of the methods. To achieve this quantitative data a proper benchmarking has been performed. This procedure evaluates the accuracy of our results comparing the ground-truth region with the same region quality metrics used in the research paper [3]. These metrics can be described as it follows:

- **Rand Index (RI):** counts the pixels pairs that share the same label in the ground-truth and the own segmented image. This value is then divided by the total number of pixels in the image, obtaining the RI quality metric. The higher the value the better.
- **Global Consistency Error (GCE):** quantifies how much a segmentation can be viewed as a modification of the other. The lower the value the better.
- **Variation of Information (VoI):** measures the difference between the ground-truth and the own segmentation in terms of relative entropy. Values close to 0 are better.

To compute these metrics, the code provided by the Berkeley Image Segmentation Benchmark Database [9] has been used. Table 3 shows the quantitative and qualitative values of the algorithm performance while working with L vs L^2.

TABLE 3: Region quality metrics of the six studied images. Evaluation done considering L^2 and L.

Image	Approach	RI	GCE	VoI
Bush	L^2	**0.9441**	**0.0534**	**0.4010**
	L	0.9310	0.0665	0.4662
Ceramic	L^2	**0.9817**	**0.0117**	**0.1677**
	L	0.9796	0.0137	0.1837
Grave	L^2	**0.9840**	**0.0128**	**0.1432**
	L	0.9820	0.0144	0.1558
Llama	L^2	**0.9747**	**0.0240**	**0.1781**
	L	0.9729	0.0255	0.1857
Person	L^2	0.9864	0.0097	0.1172
	L	**0.9868**	**0.0094**	**0.1151**
Sheep	L^2	**0.9900**	**0.0072**	**0.0852**
	L	0.9899	0.0073	0.0857

Analyzing Table 3 data, we can see that the results of the Laplacian Coordinates method are

satisfactory. Considering the quality metrics of the other methods, we can confirm that such method outperforms other segmentation methods. These results also suggest that the implementation of the Casaca et al. algorithm has been done correctly.

Now, comparing the traditional approach and the one proposed in Section 5.1, the results shown on Table 3 clearly state that the traditional method defeat the performance of our modification. As an example, in 5 out of 6 trials the traditional method performed better.

Although its resulting differences are almost meaningless, what makes a big difference is the computation time. Figure 9 represents the computational time needed to segment an image according to its number of pixels.

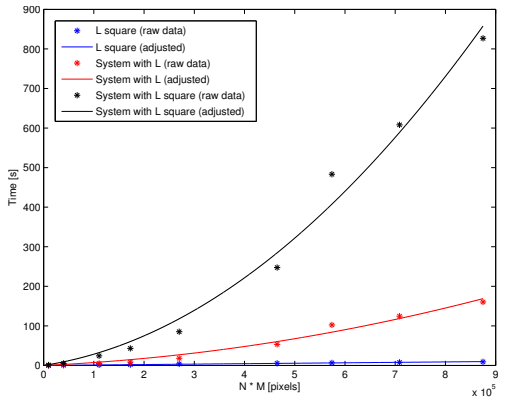

Fig. 9: Time spent on computing L^2, on solving the system with L and on solving the system with L^2 regarding the number of pixels in the image.

By evaluating the obtained results between both strategies and their computational time we have a trade-off between speed and accuracy. While the differences between the results are meaningless the method proposed in the Section 5.1 spends 5 times less than the traditional approach.

Further, we checked the ability of our implementation of performing multi-region segmentation. The qualitative and quantitative evaluation of the segmentation is the same as the one obtained with simple segmentation. The purpose of showing the multi-region segmentation is to demonstrate the code's ability to perform this task. In order to display the additional improvement Fig. 10 a). Since it has the necessary qualities to show the task. Then, as illustrated in Fig. 10 b), by selecting one set of background seeds and different sets of foreground seeds, explicitly 5, a multi-region segmentation is performed.

(a) (b)

Fig. 10: On the left, original image. On the right, manually selected seeds for multiple-region segmentation.

Using the previous setup, the system computes one mask for each set of foreground seeds, which can be seen in the first row of Fig. 11. After applying these masks to the original image, the system returns to the user one output segmented image per selected foreground. Such results can be seen in the second row of Fig. 11.

Fig. 11: From top to bottom: obtained masks and resulting segmented images.

This tool gives the user the option of segmenting more than one region at the time. Therefore, it could be suitable for some applications such as multi-tracking objects.

7 Final Remarks and Future Work

The Laplacian Coordinates method has been implemented obtaining an elegant, stable and easy-to-use program.

An important part of the work presented in this paper resides in the project management that has made possible the implementation of the project in an easy way. The avoidance of problems due to a bad organization has allowed all the members of the group to focus on their assigned tasks.

The system has been developed in C++ programming language and it is available online as an open source, allowing third party users to have access to the code. In order to improve the installation of the system a CMake file has been added

to the application. Also, with the purpose of ensure that the project basics are working properly even with future changes, the GTest modules have been integrated to the different functions of the code.

Besides the basic implementation, a further study of the method and improvements have been done. First, we proposed a slightly different mathematical approach that performs almost as good as the classical method but it reduces almost 3 times the amount of needed memory and 5 times the computation time. Secondly, the implementation of multiple-region segmentation has been included in the developed system.

A quantitative and qualitative analysis of the following methods have been reported: basic method, the method proposed in this paper, and multi-region segmentation method.

We tried to do further improvements over the basic implementation. Specifically, to dramatically reduce the computational time of the system in order to be able to perform real-time segmentation.

Some of the future works comprises the use of symmetric properties on Eigen matrices and the integration of Cuda with the developed application. It is worth noticing that we started these tasks but due to their complexity and time constraint of the project we were not able to successfully complete them.

An interesting additional future work will be a deeper study on the mathematical approach proposed in this paper. For the scope of this project, we only showed the results and benefits of modifying the classical method. However, a theoretical approach might be needed to consolidate its usage.

REFERENCES

[1] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice-Hall, ISBN 0-13-030796-3, New Jersey, 2001.

[2] A diffusion approach to seeded image segmentation. *Computer Vision and Pattern Recognition (CVPR),*.

[3] Nonato LG Casaca W. and Taubin G. Laplacian coordinates for seeded image segmentation. *In Computer Vision and Pattern Recognition (CVPR), 2014. IEEE conference*.

[4] G. Bradski. *Dr. Dobb's Journal of Software Tools*.

[5] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[6] Google Test. 2010. [Online; accessed 09-January-2016].

[7] CMake. 2015. [Online; accessed 09-January-2016].

[8] V. Kolmogorov C. Rother and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. (SIGGRAPH 04).*

[9] D. Tal D. Martin, C. Fowlkes and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *In Proc. 8th Int. Conf. Computer Vision (ICCV).*