Extraction of ancient map contents using trees of connected components

Jordan Drapeau$^{\alpha}$, Thierry Géraud$^{\gamma}$, Mickaël Coustaty$^{\alpha}$, Joseph Chazalon$^{\alpha,\gamma}$
Jean-Christophe Burie$^{\alpha}$, Véronique Eglin$^{\beta}$ and Stéphane Bres$^{\beta}$

***Abstract*—Ancient maps are an historical and cultural heritage widely recognized as a very important source of information, but exploiting such maps is complicated. In this project, we consider the Linguistic Atlas of France (ALF), built between 1902 and 1910. This cartographical heritage produces first-rate data for dialectological researches. In this paper, we focus on the separation of the content in layers for facilitating the extraction, the analysis, the visualization and the diffusion of the data contained in these ancient linguistic atlases.**

***Index Terms*—Mathematical morphology, Trees, Connected components, Linguistic Atlas, Layers of information, Filters**

I. Introduction

The Linguistic Atlas of France (ALF) is a collection of maps in paper format. It comprises 35 booklets, bringing together in 12 volumes, 1920 geolinguistic maps presenting an instantaneous picture of the dialect situation of France at the end of the 19th century. It can be defined as a first-generation atlas publishing raw data and constituting a corpus of more than one million of reliable lexical data, homogeneously transcribed, using the Rousselot-Gilliéron phonetic alphabet.

The ALF maps are mainly composed of three kinds of elements: names of departments (always surrounded by a rectangle), survey point numbers (identification of a city where a survey has been done), and words in phonetics (pronunciation of the word written in Rousselot-Gilliéron phonetic alphabet). Let us note that each map gathers the different pronunciations of a given word into a single map. An illustration of these components is given in Figure 1.

This atlas is of prime interest for the researchers in dialectology as it allows to understand how the French language has evolved over the last century. This work takes place in the context of the ECLATS projet, a French national research project[1] which aims at automatically extracting this information and generating maps with selected elements (currently, this process is done manually and it takes weeks to build a single map). More specifically, the aim of this paper is to separate each kind of information into layers in order to prepare data for subsequent analysis.

Based on our analysis of the map, it clearly appears that all the interesting pieces of information can be seen as dark connected components on a light background (initially white but degraded by time and manipulations). Filtering the connected components is a difficult process, but the sub-field of mathematical morphology based on trees of connected components offers some interesting strategies to extract the desired layers of information [2], [5].

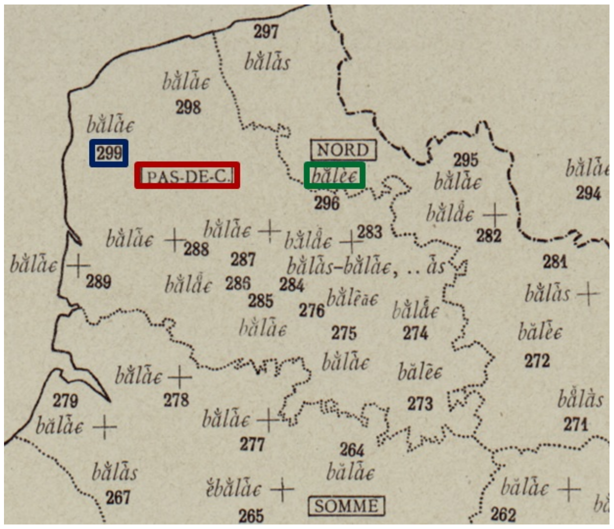

Fig. 1: A name of department in red, a survey point number in blue, and a word in phonetics in green.

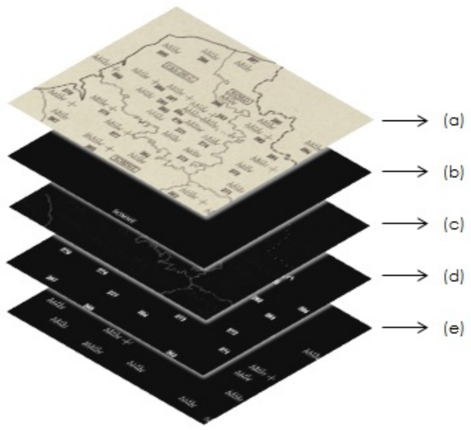

Fig. 2: (a) Map; (b) Names of departments; (c) Borders; (d) Survey point numbers; (e) Words in phonetics.

II. Tree of connected components

A. Definition

A connected component is a set of pixels which are connected to each other. It can own a group of properties, which can be its area, height, width, location, shape, etc. When considering all the threshold sets of a grey-level

$^{\alpha}$Laboratoire L3i, University of La Rochelle, 17000 La Rochelle, France `firstname.lastname@univ-lr.fr`
$^{\beta}$Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France `firstname.lastname@insa-lyon.fr`
$^{\gamma}$EPITA Research and Development Laboratory (LRDE), France `firstname.lastname@lrde.epita.fr`

[1]This work is supported by the French National Research Agency under the grant number ANR-15-CE38-0002

3	3	1	4	2
4	1	2	3	1

C	D	H	A	F
B	I	G	E	J

Fig. 3: Grey-level image (left), and pixels annotated with letters (right).

image, their connected components can be arranged into a *component tree* thanks to the inclusion relationship. Several trees exist, namely the min-tree (with lower thresholds only), the max-tree (with upper thresholds only), and the tree of shapes [1].

B. Building a tree

A component tree can be computed directly on the original gray-scale image or, to be more robust to defects, to the result of some filtering process (for instance, some thin objects can be re-connected beforehand, so that the components of threshold sets are better formed).

To better understand how a component tree is built, we use a simple example to illustrate the process. Let us take an image of 5×2 pixels having grey levels in the range $[1, 4]$ and, to make the explanations easier, let us name each pixel of the image by a letter going from A to J (see Figure 3).

For each pixel having a grey level $\lambda \geq 4$, two distinct connected components are obtained, which are $\{\mathrm{A}\}$ and $\{\mathrm{B}\}$. Note that their surrounding pixels belong to some other connected components. In the next step, we consider all pixels having a grey level $\lambda \geq 3$. We then obtain two connected components, which are $\{\mathrm{A}, \mathrm{E}\}$ and $\{\mathrm{B}, \mathrm{C}, \mathrm{D}\}$. These two new components include the two first ones. In the following steps, we consider the components obtained with the thresholds $\lambda \geq 2$ and $\lambda \geq 1$. Finally the pixels of the image can be arranged into a rooted tree, shown in Figure 4, where the arrows map a *parenthood* relationship. Some particular pixels, depicted within circles, are the representatives of the connected components: a component is a sub-tree rooted at a given representative pixel. For instance, the pixel D represents the component $\{\mathrm{B}, \mathrm{C}, \mathrm{D}\}$, obtained at threshold $\lambda = 3$.

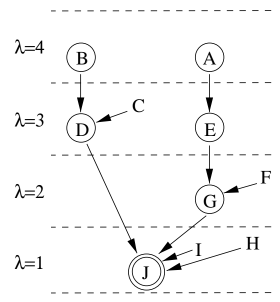

Fig. 4: Max-tree of the image of Figure 3.

The full algorithm, depicted in Figure 5, takes only a few lines of code and is very easy to implement. In an equivalent manner, the min-tree of an image can be obtained going from the lowest to the highest grey level values. Last, let us mention that computing some attributes related to connected components, and processing such trees to filter out or identify some particular connected components, are also very easy [6]. More details about these tree structures and their implementation are given in [1].

```
FIND-ROOT($x$)
1  if $zpar(x) = x$ then return $x$
2     else { $zpar(x) \leftarrow$ FIND-ROOT($zpar(x)$) ; return $zpar(x)$ }

COMPUTE-TREE($f$)
1  for each $p$, $zpar(p) \leftarrow undef$
2  $R \leftarrow$ REVERSE-SORT($f$)    // maps $\mathcal{R}$ into an array
3  for each $p \in R$ in direct order
4     $parent(p) \leftarrow p$ ; $zpar(p) \leftarrow p$
5     for each $n \in \mathcal{N}(p)$ such as $zpar(n) \neq undef$
6        $r \leftarrow$ FIND-ROOT($n$)
7        if $r \neq p$ then { $parent(r) \leftarrow p$ ; $zpar(r) \leftarrow p$ }
8  DEALLOCATE($zpar$)
9  return $pair(R, parent)$    // a ``parent'' function

CANONICALIZE-TREE($parent, f$)
1  for each $p \in R$ in reverse order
2     $q \leftarrow parent(p)$
3     if $f(parent(q)) = f(q)$ then $parent(p) \leftarrow parent(q)$
4  return $parent$    // a ``canonical'' parent function
```

Fig. 5: Code of the algorithm for creating a component tree.

III. Extraction by layers of information

Our aim was to extract by layers the informations on the maps, as we can see in Figure 2. Based on the trees of connected components we have extracted, some components can be isolated by identifying their features. So, we browse the created trees by filtering out the connected components that do not correspond to the required profile.

To make our algorithm robust, the results obtained for a given layer are removed before processing the next layer. Indeed, ignoring already identified components helps to reduce errors while extracting new components.

IV. Evaluation

In this work, we used an open source image processing library to build the trees of connected components [3]. In order to assess our work, we annotated 7 maps, which correspond to the different kinds of templates that can be found in the corpus. We run our system on these 7 maps, and using the LabelMe tool [4], we manually annotated the maps to build the ground truth. The first part of the evaluation is quite encouraging, and an analysis of our preliminary results will be discussed during the workshop.

References

[1] E. Carlinet and T. Géraud. A comparative review of component tree computation algorithms. *IEEE Transactions on Image Processing*, 23(9):3885–3895, 2014.

[2] G. Lazzara, T. Géraud, and R. Levillain. Planting, growing, and pruning trees: Connected filters applied to document image analysis. In *IAPR Intl. Wksh. on Document Analysis Systems (DAS)*, pages 36–40, 2014.

[3] G. Lazzara, R. Levillain, T. Géraud, Y. Jacquelet, J. Marquegnies, and A. Crepin-Leblond. The SCRIBO module of the Olena platform: A free software framework for document image analysis. In *Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pages 252–258, 2011.

[4] A. Torralba, B. C. Russell, and J. Yuen. LabelMe: Online image annotation and applications. *Proceedings of the IEEE*, 98(8):1467–1484, 2010.

[5] Y. Xu, E. Carlinet, T. Géraud, and L. Najman. Hierarchical segmentation using tree-based shape spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):457–469, 2017.

[6] Y. Xu et al. Efficient computation of attributes and saliency maps on tree-based image representations. In *Intl. Symp. on Mathematical Morphology (ISMM)*, pages 693–704, 2015.