Contents

List of Figures

Chapter 1

Introduction

1.1 Context of the internship

1.2 Objective and challenges

1.3 Proposed methodology

To be continue...

Chapter 2

State-of-Art

Text in images and video contains valuable information. Text appearing in images and video can provide useful semantic information and become a useful characteristic for many content-based image and video analysis tasks such as content-based image search, video information retrieval, digitize scanned document. Text analysis and recognition includes several sub-problems such as text detection, isolated character recognition, word recognition. They are researched individually ([2] [3] [11] [4] [22]) or jointly [13]. In these problem, the text detection modules is the most important part since it is shown to be critically affect the performance of text in image retrieval algorithms [4]. Although many approaches have been proposed, text detection in scene images (different from digital born one such as embedding text in images) is an open and challenging problem which has been receiving attentions. This is a difficult task due to high variation of scene text such as luminance, contrast, blur, distortion as well as variation of font type and size. Competitions has been held in order to evaluate state-of-art (Text Location Competition at ICDAR 2003, ICDAR 2005, ICDAR 2008 and ICDAR 2013). The leading approach of Task 2.1 (Text Localization in scene images) in ICDAR 2013 obtains a recall of 66.45% and precision of 88.47% [7] which poses significant challenges to state-of-the-art methods today. Related works can be divided into 3 catalogs: texture-based, connected-components and hybrids methods.

Texture-based methods [3] use a sliding windows to look for all possible texts in the image. It bases on hypotheses that text regions in images have distinct textural properties from non-text regions (gradient distribution, texture and structure) that can be identified with a learning machine technique. This approach may become costly in computation because they require calculation of features which describe texts properties of all possible sliding windows at different scale. These approach are also be limited by the learning data base which the learning machine uses. **FiXme Note: add more paper of this type**

FiXme Note!

The connected-components (CCs) methods [4] [22] group pixels, which have similar properties such as color, gray level or geometrical arrangement of edges, into connected regions. Detecting texts regardless image properties such as scales, orientation and font type is the advantage of this approach. It also provide a segmentation that can be useful in the OCR step .In the other hand, it might produce high amount of false positives. The CCs obtained may be filtered using texture properties to remove CCs that cannot be characters before grouping into words to reduce false positives rate.

The hybrid methods combine both approaches. Pan et al. [14] uses HOG features and a Waldboost cascade classifier to generate a text confidence map, based on which connected-components extraction is done by a local binarization. Liu et al [11], performs on color image, is also a hybrid approach since it use an edge on 3 channel to separate connected components. These candidates will be verified using Harr wavelet transform as texture characterization.

A typical text retrieval process includes 3 steps:

- Character retrieval: First, Text candidates must be highlighted and distinct from background. These candidate obtained by segmentation or by a characterization process, depends on the approach used (texture or CCs).
- Local classification: These candidates may be filtered out to remove non-text regions.
- Word grouping: Finally, character candidates will be grouped in to complete words.

We aim to make a reliable and fast text detectors, which can run on real time application such as detecting text in video. Many approaches **FiXme Note: cite** tried to detect all the texts candidates as possible then use a machine learning to remove the false positive before continue. This seams to be a redundant as the final purpose of text detection is to pass them into an OCR for recognition. This OCR will surely capable of verify if a candidate is truly a text.

FiXme Note!

FiXme Note!

To be continued... FiXme Note: authors use morphology

Chapter 3

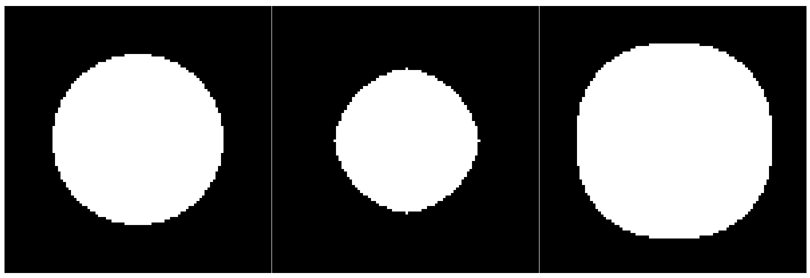
Theoretical Background

3.1 Mathematical morphology

Morphology come from Greek and means the "study of shape". Mathematical Morphology (MM) deals with describing shapes based on the set theory, integral geometry and lattice algebra. It is not just a theory but is now part of classical powerful image processing techniques. In the field of document image processing and analysis, in which the shape of objects contains prominent features, some morphological operators are regularly used. Mathematical morphology contains non-linear operator that can transform the shape of objects, thus allowing to extract them. Morphological operators can be classified as follow:

1. **Operators of binary images based on structuring element**. These operators describe the interaction of an image with a structuring element S. A binary image 1_F: $D \rightarrow \mathbb{B}$ is the indicator function of a set of point F which is a subset of and Euclidean space $\mathbb{R}^d$ or the integer grid $\mathbb{Z}^d$ of dimension d. A structuring element is a set S, usually centered in a limited definition domain. It acts like a filter for any transform applying to F. The size of S, usually small relative to the images, affects the strength of the transform and its sharp defines how the transform modify the shape of components in image. Many morphological operators has been defined, some of them are dual by set complementation. Two famous morphology operators, erosion and dilation, belong to this class. Erosion is defined as $\epsilon_S(F) = \{p | \forall s \in S, p + s \in F\}$ and dilation is defined as $\delta_S(F) = \{p + s | \forall p \in F, s \in S\}$ Their effects can be seen in 3.1b and 3.1c. They are dual operators with respect to complementation $\epsilon_S(F) = \mathcal{C}\delta_S\mathcal{C}F$. This duality property illustrates the fact that erosion and dilation do not process the objects and their background symmetrically: the erosion shrinks the objects but expands their background (and vice versa for the dilation).

   Other interesting morphological filters can be formed using the differences of two or more operators, for example the morphological gradient and morphological Laplacian. As the erosion shrinks objects and the dilation grows them, examining the difference between original image and its erosion and dilation can gave us information of the edge of objects. The morphological gradient defined by $\nabla_S(F) = \delta_S(F) - \epsilon_S(F)$. This thick gradient appears on two sides of the acture edges and can be decomposed into two half gradient $\nabla_S(F) = \nabla_S^-(F) + \nabla_S^+(F)$ with $\nabla_S^-(F) = F - \epsilon_S(F)$ is the inner gradient adheres to insides of objects and $\nabla_S^+(F) = \delta_S(F) - F$ is the outer gradient, adheres to the outside of objects.


(a) Input F (b) $\epsilon_S(F)$ (c) $\delta_S(F)$

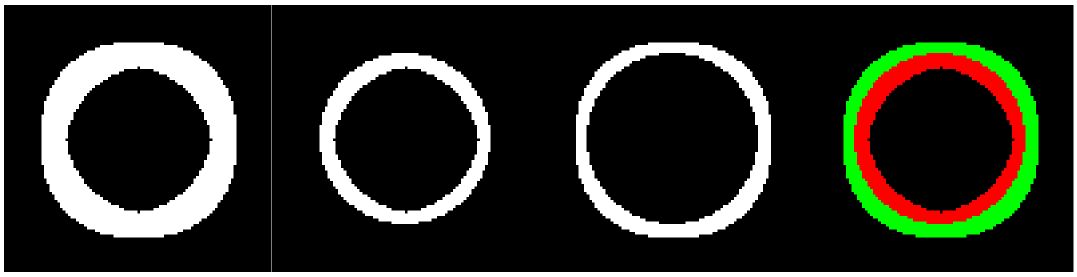

(d) $\nabla_S(F)$ (e) $\nabla_S^-(F)$ (f) $\nabla_S^+(F)$ (g) $\triangle_S\ (F)$

Figure 3.1: Binary Morphological Operations. The morphological Laplacian is colorized by Green (positive), red (negative) and black (zero)

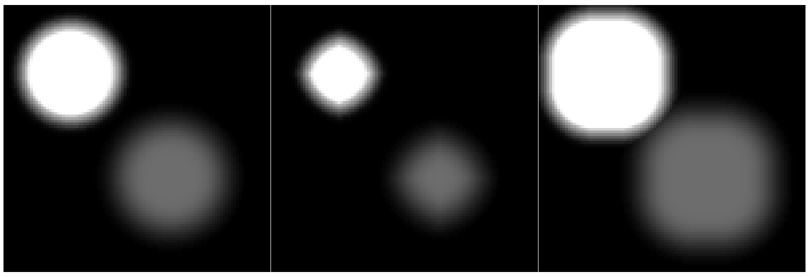
The morphological equivalent of the Laplacian is defined by $\triangle_S\ (F) = \nabla_S^+(F) - \nabla_S^-(F) = \delta_S(F) + \epsilon_S(F) - 2F$. There effects are shown in 3.1d, 3.1e, 3.1f and 3.1g

2. **Elementary operators of binary images on sets**. By replacing the structuring element B by the notion of neighborhood $\mathcal{N}$, morphological operators obtained will have elementary behavior (the slightest transform effect possible). For example, $\delta_{\mathcal{N}}(F) = \{p' \in \mathcal{N}(p) \cup p | p \in F\}$ is the elementary dilation.

3. **Extension from sets to functions** of the first two categories. Applying threshold decomposition principle, morphological operators can be extended to function i.e gray level images. $f : \mathcal{D} \rightarrow \mathbb{N}$. With a gray level t given, the subset of $\mathcal{D}$ obtained by shareholding f by t is denoted by $[f \geq t] = \{p \in \mathcal{D} | f(p) \geq t\}$. Any morphological set operator ϕ^{set} can be extended to define a morphological operator on gray level function ϕ using $\phi(f)(p) = max\{t | p \in \phi^{set}([f \geq t])\}$. There effect can be seen in 3.2.

4. **Connected operators**. Connected operators [15] are operator that work by merging elementary region called flat zones. It verify $\forall p, \forall p' \in \mathcal{N}(p), f(p') = f(p) \Longrightarrow \phi(f)(p') = \phi(f)(p)$. As a consequence, connecting operators cannot create new contours, nor modify their position. Therefore, they have a very good contour preservation properties. There are two commonly used classes of connected operators: filters by reconstruction and algebraic openings and closings [20]. Although they are well-know in the binary case, where only objects "marked" by another image are extracted, they can be defined for gray level images by shareholding the input image at different level t. **FiXme Note: example** FiXme Note!

Mathematical morphology is a powerful tool for image processing. It found application in numerous field: geoscience and remote sensing, materials science, biological and medical imaging, industrial applications, identification and security control, document processing and image coding [19]. These tools provide a strong impression, especially class of morphological operations that do not alter the contours of objects


(a) Input F (b) $\epsilon_S(F)$ (c) $\delta_S(F)$

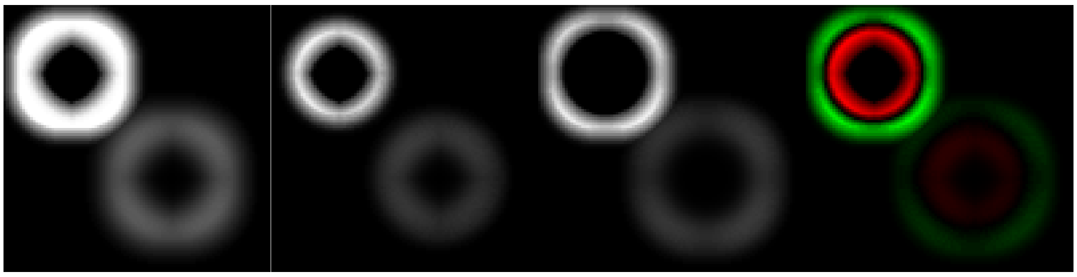

(d) $\nabla_S(F)$ (e) $\nabla_S^-(F)$ (f) $\nabla_S^+(F)$ (g) $\triangle_S\,(F)$

Figure 3.2: Morphological Operations on set, using a square 10x10 pixels as the structuring element. (The laplacian is colorized by Green (positive), red (negative) and black (zero))

3.2 Digital topology and Self-Duality

3.2.1 Basic notions of digital topology

A 2D discrete image I is a function $\mathbb{Z}^2 \rightarrow E$ with E is a set. I is a digital binary image when $E = 0, 1$ and it is said a gray scale image when $E = 0, \dots, k$. A point of I is a couple $p = (x, y) \in \mathbb{Z}^2$. Two point $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$ are:

- 4-adjacent if and only if $d_4(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2| = 1$
- 8-adjacent if and only if $d_8(p_1, p_2) = max(|x_1 - x_2|, |y_1 - y_2|) = 1$

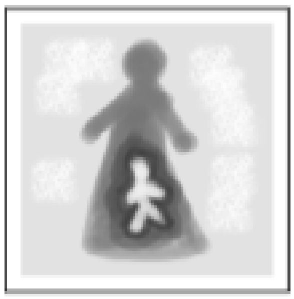
The n-neighborhood $N_n(p)$ of a point p is the set of all the points n-adjacent to p, with n = 4 or n = 8. **FiXme Note: continue**

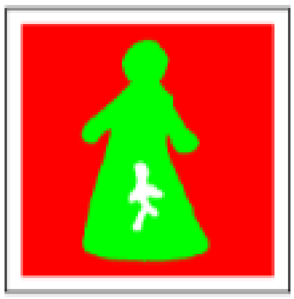
FiXme Note!

3.2.2 Self-Duality

In the case of text detection in gray scale image, a well known problem encountered is to treat bright texts over dark background and dark texts over bright background. Common workaround which processes both the original and it complement will increase computation cost. Consequently, we interest in the class of self-dual operator.

A self-dual operator processes the image contents regardless its contrast [5]. In general, operators which are not self-dual do not treat bright objects over dark background and dark objects over bright background similarly which is often an undesirable feature. It is most important when no assumption about the contrast between object and background can be made. As any part of the images could be the subject, we want an operation which behavior


(a)


(b)

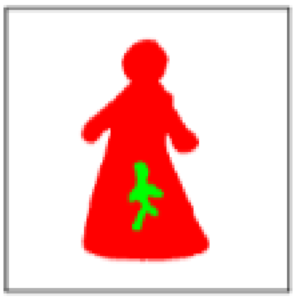

(c)

Figure 3.3: From the original gray level image 3.3a, the subject can be either the mother 3.3b (dark over bright) or the baby 3.3c (bright over dark)

the same way regardless the contrast between the subject and its background to obtain a unique representation.

As we can seen in 3.3, where green and red respectively represent the background and foreground. From the grayscale image 3.3a If we focus in the mother, then the outer zone will be the background as in 3.3b. Furthermore, we can chose the baby as the subject and therefore, the woman will become the background 3.3c.

One of the major draw backs of digital topology is the connectivity paradox. As said in [8], if the same connectivity is used for both the object and background, the Jordan curve theorem is not respected. In order to solve this problem, a pair of connectivity is used. For a image defined in the regular cubical grid, the digital topology must be describe by a "Jordan pair" of connectivity (c_α, c_β) . One connectivity is used for the dark pixels and the other one for the bright pixels. **FiXme Note: connectivity image** FiXme Note!

For example, if the image **??** is describe by the pair (c_4, c_8) (the 4-connectivity for bright pixels and 8-connectivity for dark pixels) we will obtain 1 object. On the same image, using (c_8, c_4) will gave us 2 separate objects. This arbitrary choice affect the topology. In effect, self-dual operation must use different connectivity for the complementary to work in the same way as demonstrated in **??** **??**: the complement image must use the (c_8, c_4) connectivity to gave the same presentation of the original image which use (c_4, c_8). As no assumption is made about the gray level of background and object, the choice of connectivity pair is better avoided. Another approach draws our attention to : the well-composed images. Latecki et al. [9] present this image class which enjoys very useful topological and geometric properties, especially the fact that there is only one connectivity relation on points of the image.

3.3 Well-composed Images

As defined in [9], a 2D set S is weakly well-composed if any 8-component of S is a 4 component. S is well-composed if both S and its complement $\bar{S}$ are weakly well-composed. It can easily be defined using the notion of "critical configurations" which are ▚ and ▞ : S is weakly well-composed if these configuration do not appear.

The notion of well-composednessed has also been extended to gray-level images. A gray-level image μ is well-composed if any set $[\mu \geq \lambda]$ is well-composed. The extended version of "critical configuration" is that every block $\begin{array}{|c|c|}\hline \text{a} & \text{d} \\ \hline \text{c} & \text{b} \\ \hline\end{array}$ must verify interval(a,b) $\cap$ interval(c,d) $\neq \varnothing$, where interval(a,b) = [min(a,b),max(a,b)].

Latecki has shown in [10] that a topology preserved digitization process must results in a well-composed image. An image is not *apriori* well-composed but its attribute has motivated de development of algorithms to make an image well-composed. There are currently 2 approach to get a well-composed image from the original image: by changing it pixel value (with the possibility of alter the image's topology) or by a well-composed interpolation.

3.4 Connected filter

Motivated by families of filters by reconstruction, the notion of filter by reconstruction is introduced in [15] [17]. It work by simplifier the topographic map of images. The most important feature of this type of operator is preservation of contours [16]: they does not create new contours nor shift them.

3.4.1 Morphological Tree-Based Image Representation

One strategy to define connected operators replies on a hierarchical region-based representation of the input image, in another word, a tree. The filtering step is done by pruning that tree and the output image is compute by reconstructing the pruned tree. Tree of shapes is a contrast-invariant image representation which is defined in [12].

A Couple of Dual Trees: given an image μ in nD: $\mathbb{Z}^n \to \mathbb{Z}$, the lower cuts of μ are defined by $[\mu < \lambda] = \{x \in X | \mu(x) < \lambda\}$. The set of all connected components of all these cuts is $\mathcal{T}_<(\mu) = \{\Gamma \in \mathcal{CC}([\mu < \lambda])\}$ (with $\mathbb{CC}$ denote the operator that takes a set and gives its set of connected component. As these sets verify $\forall\Gamma$ and $\Gamma' \neq \lambda$, $\Gamma \subset \Gamma'$ or $\Gamma \cap \Gamma' = \emptyset$, they can be arranged into a tree, called the *min-tree* (because it if form using lower cuts, and the leaves will be minimum of image). We can also consider the upper cuts $[\mu \geq \lambda] = \{x \in X | \mu(x) \geq \lambda\}$, the element of $\mathcal{T}_\geq(\mu) = \{\Gamma \in \mathcal{CC}([\mu \geq \lambda])\}$ will be arranged in to the *max-tree*. The *max-tree* and *min-tree* are dual by complementation, the max-tree of μ is the min-tree of μ, therefore operations define on them are dual.

A Self-Dual Tree:We use two set $\mathcal{T}_<(\mu)$ and $\mathcal{T}_\geq(\mu)$ with their holes filled to define two other sets $\mathcal{S}_<(\mu)$ and $\mathcal{S}_<(\mu)$. We denote *Sat* as the cavity-fill-in operator, these new sets are defined as: $\mathcal{S}_<(\mu) = \{Sat(\Gamma); \Gamma \in \mathcal{T}_<(\mu)\}$ and $\mathcal{S}_\geq(\mu) = \{Sat(\Gamma); \Gamma \in \mathcal{T}_\geq(\mu)\}$. The tree of shape is defined as: $\mathfrak{S}(\mu) = \mathcal{S}_<(\mu) \bigcup \mathcal{S}_\geq(\mu)$. We have also $\Gamma \subset \Gamma'$ or $\Gamma \cap \Gamma' = \emptyset$ with $\Gamma, \Gamma' \in \mathcal{S}$. There is a quasi-linear algorithm to compute the tree of shape, which works also in the case of nD presented in [6]. This tree is *self-dual* because many self dual operation can be defined in this tree *¡reading article¿*.

The min and max tree are easily compute with a few line of code [1] and the tree of shape can be obtain thanks to [6]. The tree encoding is very compact in term of memory: the parenthood relationship between pixels is an image having the same size of the input.

To be continue... (consider add the attribute and definition of Sat)

Chapter 4

Approach

4.1 Hypothesis

Texts included in images contain a rich information source. They are difficult to be detected due to many factors as presented in 2 as their various sizes, luminance, complex milieu but interesting text can be narrow down by some criteria:

1. **Size:**

   Text can appear in variety of sizes in images. But interesting text should be readable and therefore there should be a minimal size of text. Esually, the larger font size, the more important is the text. Text which is very small cannot be recognized easily by OCR engines, nor human eyes anyway. However, a upper bound on character sizes is uninvited.

2. **Color and Intensity:**

   A readable scene text must at least contrast to their background. Each character tends to have a perceptually uniform color and intensity. Generally, characters of the same word will be in the same background.

3. **Alignment and inter-character distance:**

   Generally, important text in images appear mainly in horizontal direction. Scene text direction may suffer by different distortions that changes alignment but they are generally in a near horizontal direction.

   Characters in the same word should have small variation in height. The uniform distance between characters in the same word are not grantee due to distortions but the inter-character can be limited by a function of its height and width.

Many author working with CCs approach used a verify step by a learning machine. This is not really interesting as non-text component can lately rejected by an OCR engine. We try to increase the recall rate at the cost of the precision rate.

4.2 The Tree of Shapes of the Laplacian

Based on these hypothesis, region which is contrast to their background make a good candidate to be text. Many authors worked with the same hypothesis, using MSER [13] [22] [18] or edges detectors [11] [23] to extract components. For this task, we found that the Laplacian is excellent for extracting components having different contrast as the sign of Laplacian output already provides us an segmentation of the image. More over, the Laplacian treats the dark over light and light over dark in the same way which allow us to apply the method only once. The zero-crossing of the laplacian are closed-contours and mark the boundary of object. In reconstructing a tree from these components by their inclusion, the background - objects relations shown are useful for characters grouping and elimination of false positive characters detected.

To compute a representation tree, mathematical morphology requires at first an order function of pixel values. In the case of color images, since there is not a natural order of color, the definition of is important and affect the performance. A classical workaround is to sort using their luminance by converting color image to gray-level image.

Our approach base on the Laplacian morphology on the gray-level image. By using laplacian morphology, we can distinguish objects through the contour detected. A transition between positive - negative zone and vice versa mark a contour. Pixels having same sign belong to same region. Null pixels corespondent to homogeneous regions and therefore belong to the region they are included in. To maintain the duality, null pixels on the zeros crossing will belongs to the outer region. The size of objects conserved is depend on the windows using. Remind that the Laplacian morphology is calculate by $\Delta_{\square}(f) = \delta_{\square} + \varepsilon_{\square} - 2f$ as defined in [21]. As the sign of Laplacian carries the information needed so the structure of the tree of shapes of the Laplacian (ToSoL) is:

- A zone negative or positive will form a node.
- The root node is either positive or negative, defined by the median of pixels on the bother.
- A zero point included in another node will belongs to that node.
- A note included in another node is a descendant of the former node.

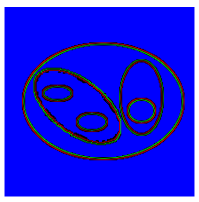
The ToSoL is the representation of region having similar luminance and their inclusion relationship. This structure is more compact than the tree of shape presented in [6] because the information require to compute is simpler (sign of Laplacian instead of the value of each pixel). Thus, we can present the tree as a table of parents instead an image of same size as the original image. **FiXme Note: treat back on white and white on black**

FiXme
Note!

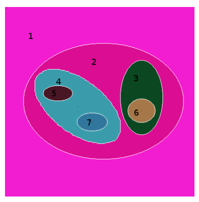
That ToSoL will be pruned using different condition in order to eliminate components which is impossible to be character. The hierarchical structure also allows us to group candidate in the same back ground (having same parent).

The propose approach will follow this procedure: First, the input image will be convert in to gray level image. Then the morphological Laplacian will be calculated. The morphological gradient will also be calculate to determine if a zero-crossing of Laplacian is strong enough to be kept. The well-composed interpolation will be applied on these results. After that, a tree of shape will be calculated from the well-composed interpolated Laplacian: each node will be

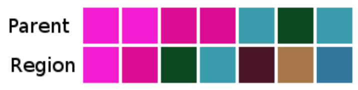
(a) Original image



(b) Laplacian red is negative green is posifive and blue is zero



(c) segmentation resultat



(d) The Tree will be present as a table

Figure 4.1: Example of Tree of Shape of Laplacian in segmentation of images.

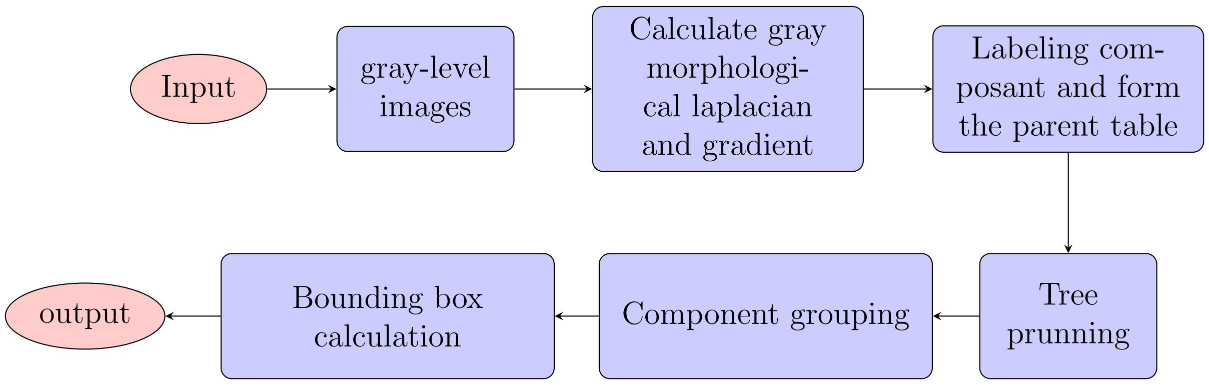

Input
gray-level images
Calculate gray morphological laplacian and gradient
Labeling composant and form the parent table
Tree prunning
Component grouping
Bounding box calculation
output

Figure 4.2: The processing chain of our approach

labeled with a same value, the parent relation will be given by a table of parent which shows us the parent of each label. Not all zeros crossing will be the beginning of a new node, new node only be started if the average gradient on the outer contour is stronger than a threshold. Then, components which have possible shape and similar size will be grouped together to form word candidates which will remove stand alone candidate which is rarely a case of a text. A bounding box will be calculated for each word candidate. The false positives will be eliminated during the procedures en pruning the tree by shape, size, how strong the region is in term of Laplacian...

seem fine

Chapter 5

Implementation

This chapter will clarify the implementation of the approach present in 4.

5.1 Gray-level conversion and well-composed interpolation

First of all, the color input images will be transformed into gray level image using its luminance information. Most digital images using the RGB (or sRGB) which use the ITU-R BT.709 primaries. The coefficients defined to calculate the relative luminace from linear RGB components is: $Y = 0.2126R + 0.7152G + 0.0722B$. The formula reflects the spectral sensitivity of human visual perception of brightness: green light contributes the most to the intensity perceived by humans, and blue light the least. Other steps will work on this image.

At first, morphological Laplacian is calculated using a relatively large size structuring element to reduce influences of small components. A morphological gradient will also be calculated to provide information for pruning the tree *a priori*. These output images will be doubled the resolution using a well-composed Interpolation **FiXme Note: add more detail** FiXme Note!

The topology of the output images are not well-composed and therefor does not provide the self-dual **FiXme Note: why we have to interpolate**. Our method will double the resolution of the image by a well-composed interpolation. This interpolation will add temporary points: FiXme Note!

As proved in [5], if added points is calculate by the median of points around them will leads to a self-dual plain map.

a	$(a+b)/2$	b
$(a+c)/2$	$median(a,b,c,d)$	$(b+d)/2$
c	$(c+d)/2$	d

Table 5.1: Well-composed interpolation

5.2 Objects labeling

5.2.1 Labeling by front propagation

From the well-composed interposed laplacian, we will label all connected components by front propagation. A connected component contain a set of connected pixels that has same sign of Laplacian and zeros that included in that region. As the zeros on the contours is always include in the outer region, we treat connected component in a dual-way i.e this process applied to complement image will gives the same result. Thanks for the well-composed interpolation, we can use only one type of connectivity. We chose the the 4-connectivity to reduce number of neighbors needed to be checked . As there are only 2 type of component (positive and negative one), a region is always surrounded by one other region. With the labeled image, we will present the tree structure with a parent table which tell the parent of each label. The parent of the root will be itself.

Because of large number of component extracted by the Laplacian, we will try to prune the tree at the same time of labeling. Whenever we found a new component, some criteria will be check first to decide if a new label will be spread or continue using the label of that component's parent. These criteria includes average gradient of points on the outer contour and the perimeter of that contour.

The labeling algorithm is based on front propagation algorithm. At first, an all zeros images is create with the same size like the interpolated Laplacian. Then the output images is read from left to right, top to bottom. When we read point which is not labeled (i.e maintain the null value), a new component is detected and a label will be propagate from that point until all connected points which have same laplacian sign are labeled. If it is not the first point (root component), we will follow points on its boundary to calculate the average gradient and perimeter of this contour and decide if a new label will be use. This contour following algorithm will be presented in 5.2.2. If the component is small or its gradient is week, we will use its parent's label to mark this component. The label will be propagated in checking its 4 neighbors. If a neighbor is inside the image, still not labeled and having same sign of the region, it will be labeled and the label will continue propagate from that point. The sign agreement can be checked simply by a xor operation between sign of region and sign of that point on the Laplacian. During the labeling process, other information can also be gather such as the bounding box, area, average gray level of each component.

As text must be large enough, if it is not, even the human eye can not recognize them, nor the OCR. Size of a component will be verified by its perimeter and bounding box size. A component is considered noteworthy if:

- $Bounding\_Box\_height > 5 pixels$ and $Bounding\_Box\_width > 5 pixels$
- $\dfrac{Bounding\_Box\_height}{Bounding\_Box\_width} > 5$
- $\dfrac{Bounding\_Box\_width}{Bounding\_Box\_height} > 10$

The pseudo code is presented in Algorithm 1

Algorithm 2 Labeling process to construct the tree

```
1: procedure LABELING(IMAGE,LAPLACIAN,GRADIENT)
2:     $output \leftarrow$ zeroes
3:     $level \leftarrow 1$
4:     $boundingBox \leftarrow image.Domain()$
5:     for all $p$ do
6:         if $output(p)$ then continue
7:         if p is the first point then
8:             $current \leftarrow level$
9:             $tree.parent.push\_back(current)$
10:            $tree.color.push\_back(0)$
11:            $tree.area.push\_back(0)$
12:        else
13:            $followEdge(p, grad, perimetre, boundingBox)$
14:            if $grad > gradThresHold$ and $notSmall(perimetre,boundingBox)$ then
15:                $level \leftarrow level + 1$
16:                $current \leftarrow level$
17:                $tree.parent.push\_back(current)$
18:                $tree.color.push\_back(0)$
19:                $tree.area.push\_back(0)$
20:                $tree.boundingBoxes.push\_back(boundingBox)$
21:                $newRegionFlag \leftarrow true$
22:            else
23:                $current \leftarrow output$
24:                $newRegionFlag \leftarrow false$
25:        $output(p) \leftarrow current.$
26:        $sign\_flag \leftarrow laplacian(p) > 0.$
27:        $Queue.push(p).$
28:        while $Queue$ is not empty do
29:            p ← Queue.pop()
30:            $tree.color[current - 1] \leftarrow tree.color[current - 1] + image(p)$
31:            $tree.area[current - 1] \leftarrow tree.area[current - 1] + 1$
32:            for all $neighbor\ of\ p$ do
33:                if n $inside$ image and output(n) = 0 and sign_flag = laplacian(n)>0 then
34:                    $output(n) = current$
35:                    $Queue.push(n)$
36:    for $i = 0; i < level; i++$ do
37:        tree.color[i]=tree.color[i]/tree.area[i];
38:    return tree, output
```

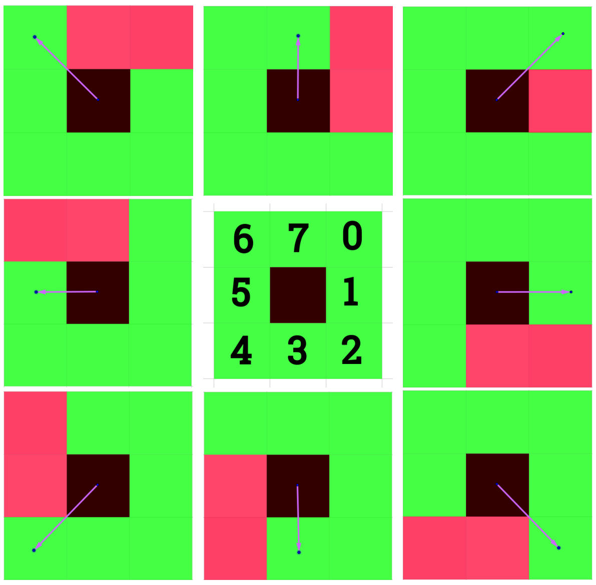


Figure 5.1: Possible positions of next point

5.2.2 Average gradient on contour calculation

We prune the tree during the labeling process to reduce the calculation cost of reading the image multiple time. By merging nodes which has low contrast with the upper node which means having low gradient in the contour the remain tree only maintain high contrast component. By calculate average gradient of these points, we can decide which node to remove. There are many approach to obtain the contour, for example we can use a dilation follow by a erosion using a element cross as structure element to obtain the contour. But the calculation of erosion and dilation costs. Because the topology is simplified by the well-composed interpolation, which made every component is surround by only one other component, we can apply a simple contour following method to follow its outer contour.

As after each label propagation, component which is not yet labeled will remain null and surround by another label. For every new component detected except the first one (root), we will first follow the outer contour of the object to calculate its average gradient. As we read the image from left to right, top to bottom, first point of new region is always the left top most point of that component.

The algorithm will try to follow the component in clockwise direction so the object will always on the left of each point of the contour. We initialize the starting point is the one on the left of first point of component. There are 8 possible direction of next point to advance (direction are numbered as in 5.1). The next point must be the one which is labeled and has a non labeled point on its left. The searching order is a semicircle clockwise from last direction plus a semicircle counter clockwise from last-direction (for example if the last direction is 0, we will check these directions in this order: 0 1 2 3 7 6 5 4). We continue this until reach the starting point. The length and gradient will be gathered during the process.

Algorithm 4 Contour following process

```
1: $iter \leftarrow [0, 1, 1, 1, 4, 7, 7, 7]$
2: procedure FOLLOWCONTOUR(P)
3:     $np = p$
4:     $count \leftarrow 0$
5:     repeat
6:         for $i = 0; i < 8; i++$ do
7:             $direction \leftarrow (direction + iter[i])$ mod 8
8:             if $output(np + direction)! = 0$ and $output(np + direction + 1) = 0$ then
9:                 $np \leftarrow np + direction$
10:                $grad \leftarrow gradientImage(np)$
11:                $count \leftarrow count + 1$
12:    until $np == p$
13:    $grad \leftarrow grad/count$
14:    $perimetre \leftarrow count$
15:    return (grad,perimeter)
```

5.3 Component grouping and bounding box calculation

All the component extracted by the Laplacian will be treated as a character candidate. Of cause they contain a lots of false positive. The labeling process has remove some small or components .After having the tree structure, we will filtered out components which does not match these criteria:

- $Bounding\_Box\_height > 5pixels$ and $Bounding\_Box\_width > 5pixels$
- $\dfrac{Bounding\_Box\_height}{Bounding\_Box\_width} > 5$
- $\dfrac{Bounding\_Box\_width}{Bounding\_Box\_height} > 10$
- $Filling = \dfrac{real\_area}{Bounding\_Box\_height * Bounding\_Box\_width} > 0.1$

The first three criteria has met since these nodes has been filtered out during the construction of the tree with a single loop. Nodes do not meet the forth criteria will be flagged so they will be transparent during the grouping process. After that, word candidate will be created by grouping remain nodes which have same parent (share the same background) and have relatively small horizontal distance. This value chose is the maximum of height and width of each components. In reading backward the tree table (from leaves to root), we will try to group component with their neighbors. We look at its left and right to find brother node which is in ranges and isn't grouped to another word. For a component found, a double check is done to make sure the other component is in range of this one. This component is grouped in to current word and new component will be search from this one.

We only interest in word which has more than 2 characters. Only components with similar height is grouped together because characters in the same word do not vary more than 2 of smallest character. We use the generalized Jaccard similarity:

$$J(\mathbf{x},\mathbf{y}) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$$

In applying to the height of 2 components, it becomes simple. Components are only grouped if the similarity is greater than 0.5.

$$J(\mathbf{height1},\mathbf{height2}) = \frac{(min(height1, height2)}{min(height1, height2)} < 0.5.$$

We use different strategy to find the neighbors of components. At first, only one searching pointer is use and it situates in the center of that component. We also use 3 different searching pointer, one in the middle, one at 10% of height from the top and another at 10% of height from the bottom. The pseudo code is present in Algorithm 5 for the case of only one searching pointer in the center..

Algorithm 5 Tree prunning and Word candidate grouping

```
1: procedure GetNeighbors(Component a,component b)
2:     return $min(a.height, b.height)/max(a.height, b.height) < 0.5$
3: procedure GetNeighbors(Start point $p$,searching direction dp)
4:     $current \leftarrow image(p)$
5:     if direction = left then
6:         move $p$ left thisComponent.width/2
7:     else
8:         move $p$ right thisComponent.width/2
9:     for $i = 0; i < height; i++$ do
10:        $p = p + dp$
11:        if image does not contain $p$ then continue
12:        if $image(p) = current$ then continue
13:        if $image(p) \neq 1$ and $parent(thisComponent) = parent(componentAt(p))$ and
           $isSimilar(thisComponent, componentAt(p))$ then
14:            return image(p)
15:        else
16:            $current = image(p)$
17:    return 0


18: function JoinWord(label,box)
19:     $found = 0$
20:     $deja_vu(label) = true$
21:     if $NeighborRight(label)$ and $label = NeighborLeft(NeighborRight(label))$ and not
        $deja_vu(NeighborRight(label))$ then
22:         $box.extendToMatch(NeighborRight(label))$
23:         $found \leftarrow found + 1 + JoinWord(NeighborRight(label), box, oldNode)$
24:     if $NeighborLeft(label)$ and $label = NeighborRight(NeighborLeft(label))$ and not
        $deja_vu(NeighborLeft(label))$ then
25:         $box.extendToMatch(NeighborRight(label))$
26:         $found \leftarrow found + 1 + JoinWord(NeighborRight(label), box, oldNode)$
27:     return found


28: procedure GetWordCandidate(a,b)
29:     for all label do
30:         if MatchCharacterCondiditon(label) then
31:             deja_vu(label) = true
32:         else
33:             deja_vu(label) = false
34:     for all label except root do
35:         NeighborLeft(label) = GetNeighbors(label.center, leftPointer)
36:         NeighborRight(label) = GetNeighbors(label.center, rightPointer)
37:     for all label backward except root do
38:         new empty box
39:         if joinRegions(label,box) then WordCandidate.push_ back(box)
```

Bibliography

[1] Christophe Berger, Thierry Géraud, Roland Levillain, Nicolas Widynski, Anthony Baillard, and Emmanuel Bertin. Effective component tree computation with application to pattern recognition in astronomical imaging. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 4, pages 41–44, San Antonio, TX, USA, September 2007.

[2] Teófilo E. De Campos, Bodla Rakesh Babu, and Manik Varma. Character recognition in natural images. 2010.

[3] Xiangrong Chen and A.L. Yuille. Detecting and reading text in natural scenes. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–366–II–373 Vol.2, June 2004.

[4] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2963–2970, June 2010.

[5] Thierry Géraud, Edwin Carlinet, and Sébastien Crozet. Self-duality and digital topology: Links between the morphological tree of shapes and well-composed gray-level images. In J.A. Benediktsson, J. Chanussot, L. Najman, and H. Talbot, editors, *Mathematical Morphology and Its Application to Signal and Image Processing – Proceedings of the 12th International Symposium on Mathematical Morphology (ISMM)*, volume 9082 of *Lecture Notes in Computer Science Series*, pages 573–584, Reykjavik, Iceland, 2015. Springer.

[6] Thierry Géraud, Edwin Carlinet, Sébastien Crozet, and Laurent Najman. A quasi-linear algorithm to compute the tree of shapes of n-D images. In C.L. Luengo Hendriks, G. Borgefors, and R. Strand, editors, *Mathematical Morphology and Its Application to Signal and Image Processing – Proceedings of the 11th International Symposium on Mathematical Morphology (ISMM)*, volume 7883 of *Lecture Notes in Computer Science Series*, pages 98–110, Uppsala, Sweden, 2013. Springer.

[7] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Gomez i Bigorda, S. Robles Mestre, J. Mas, D. Fernandez Mota, J. Almazan Almazan, and L.-P. de las Heras. Icdar 2013 robust reading competition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1484–1493, Aug 2013.

[8] T. Y. Kong and A. Rosenfeld. Digital topology: Introduction and survey. 37:357–393, Dec. 1989.

[9] Longin Latecki, Ulrich Eckhardt, and Azriel Rosenfeld. Well-composed sets. *Computer Vision and Image Understanding*, 61:70–83, 1995.

[10] LonginJan Latecki, Christopher Conrad, and Ari Gross. Preserving topology by a digitization process. *Journal of Mathematical Imaging and Vision*, 8(2):131–159, 1998.

[11] Yangxing Liu, Satoshi Goto, and Takeshi Ikenaga. A contour-based robust algorithm for text detection in color images. *IEICE - Trans. Inf. Syst.*, E89-D(3):1221–1230, March 2006.

[12] P. Monasse and F. Guichard. Fast computation of a contrast-invariant image representation. *IEEE Transactions on Image Processing*, 9(5):860–872, May 2000.

[13] Lukas Neumann and Jiri Matas. Real-time scene text localization and recognition. In *Proc. CVPR*, pages 3538–3545, 2012. calculate running time.

[14] Yi-Feng Pan, Xinwen Hou, and Cheng-Lin Liu. Text localization in natural scene images based on conditional random field. In *Proc. ICDAR*, pages 6–10, 2009.

[15] P. Salembier and J. Serra. Flat zones filtering, connected operators, and filters by reconstruction. *IEEE Transactions on Image Processing*, 4:1153–1160, 1995.

[16] P. Salembier and M.H.F. Wilkinson. Connected operators. *Signal Processing Magazine, IEEE*, 26(6):136–157, November 2009.

[17] Jean C. Serra and Philippe Salembier. Connected operators and pyramids. *Proc. SPIE*, pages 65–76, 1993.

[18] Cunzhao Shi, Chunheng Wang, Baihua Xiao, Yang Zhang, and Song Gao. Scene text detection using graph model built upon maximally stable extremal regions. *Pattern Recognition Letters*, 34(2):107–116, January 2013.

[19] Pierre Soille. *Morphological Image Analysis: Principles and Applications.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2 edition, 2003.

[20] L. Vincent. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, Apr 1993.

[21] Lucas J. Van Vliet, Ian T. Young, and Guus L. Beckers. An edge detection model based on non-linear laplace filtering, 1988.

[22] Kaizhu Huang Xu-Cheng Yin, Xuwang Yin and Hong-Wei Hao. Robust text detection in natural scene images. *Pattern Analysis and Machine Intelligence*, 36(5):970–983, 2013.

[23] Jianqiang Yan, Jie Li, and Xinbo Gao. Chinese text location under complex background using gabor filter and svm. *Neurocomputing*, 74(17):2998–3008, October 2011.