UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Extracting Speech Text from Comics

Pedro Daniel Clemente Rodrigues Inácio

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Abel João Padrão Gomes

Covilhã, Outubro de 2016

Agradecimentos

A Deus por me ter dado saúde e força para ultrapassar dificuldades.

Ao meu orientador, Prof. Dr. Abel Gomes, pelo suporte, conselhos e incentivos.

Aos meus pais, pelo amor, amizade, incentivo e apoio incondicional.

A todos vós, o meu grande BEM HAJA.

Resumo

No geral, tem sido desafiante encontrar soluções capazes de extrair correctamente distintos tipos de balões de texto a partir de qualquer tipo de banda desenhada, mas particularmente da mais complexa. O desafio provém do facto de que não existe na literatura um algoritmo capaz de lidar com quaisquer balões de texto sem fazer qualquer suposição em relação à profundidade de cor da imagem, orientação ou linguagem do texto. Pior ainda, é o facto de que a arte da banda desenhada evolui ao longo do tempo, o que faz com que exista um certo grau de imprevisibilidade associado aos livros. Isto significa que, um algoritmo pode funcionar bem para livros de banda desenhada lançados há vinte anos atrás, mas não tão bem para livros mais actuais, mesmo considerando que eles pertencem à mesma categoria ou série.

Com esta dissertação pretende-se apresentar uma possível solução para este problema, ao introduzir um algoritmo capaz de extrair balões de texto de páginas de banda desenhada. O algoritmo apresentado, aqui designado por CCD (components and corners detection), baseia-se no conceito de detecção de cantos para identificar trechos de texto dentro de componentes candidatos a balão. Assim sendo, depois de descartar um número significativo de regiões que não são consideradas balões de texto por um ou outro motivo, olhamos para a forma dos buracos das restantes regiões para verificar se ainda possuem um número significativo de cantos que seja capaz de fazer com que um candidato seja classificado como balão de texto.

Palavras-chave

Livros de Banda Desenhada

Extracção de Balões de Texto

Extracção de Texto

Detecção de Arestas

Detecção de Cantos

Análise de Componentes Ligados

Análise de Contornos

Abstract

Overall, it has been challenging to find solutions able to correctly extract distinct types of text balloons from any sort of comics, but in particulary from complex comic books. The challenge comes from the fact that there is no general extraction algorithm in the literature capable of handling any text balloons without making any assumption regarding color depth of the image, orientation or language of the text. Even worse, it is the fact that the comics art evolves over time, so that there is some degree of unpredictability associated to comics. This means that, an algorithm may work well for comic books released twenty years ago, but not so well for current comic books, even considering they belong to the same category or series.

With this dissertation it is intended to present a possible solution to this problem, by introducing an algorithm capable of extracting text balloons from comic book pages. The presented algorithm, here called CCD (components and corners detection), relies in the concept of corner detection to identify text snippets inside balloon candidates. So, after discarding a significant number of regions that are not considered as tentative text balloons for one reason or another, we look at the shape of the holes of the remaining regions to check if they still hold a significant number of corners capable to make a candidate be classified as text balloon.

Keywords

Comic Books

Text Balloon Extraction

Text Extraction

Edge Detection

Corner Detection

Connected Component Analysis

Contour Analysis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The research problem addressed in this dissertation has to do with the feasibility of applying digital image processing techniques to comic book images, in order to extract text balloons, without making any assumptions regarding color depth of the image, orientation, and language of the text. As shown throughout this dissertation, the main novelty of this work lies in the fact that text snippets are identified before their corresponding balloons (if any exists), not the other way round. As a consequence, it becomes possible to identify text snippets, no matter they are enclosed in balloons or not.

1.1 Motivation

Nowadays, with the information era transforming our ways of living, the world is becoming increasingly digital and interconnected. We are connected to our watches, computers, tabletops, cars, and more importantly to smartphones. To this it is not strange the fact that pictures studios are making significant investments in digital comics, taking advantage of their distribution over the Web and apps that allow comics on smartphones and other small devices. Thing is, while the number of the digital comics is increasing, the number of algorithms capable correctly retrieving key elements (e.g., speech balloons, captions, panels and characters) from comic book pages is still limited. In fact, there is no general algorithm to extract all of such elements at once.

Even information retrieval systems associated to digital comics are seemingly restricted to metadata (e.g., keyword search of comic book title, publisher, gender, and artist). Therefore, when someone tries to read a comic book with a mobile device, he/she needs to zoom in all text elements in order to follow and understand the flow of a story. This makes a reading experience very unpleasant and could lead readers to not choose a mobile device to have fun in reading comics.

In order to solve this problem, we propose to use digital image processing techniques to extract text balloons (e.g., speech balloons or captions) from comic books. This shall allow us zooming in of such balloons without zooming in of each book page as a whole.

1.2 Problem statement

Before proceeding any further, let us have a brief refresher about text balloons. A text balloon can be defined as an area owning text in some manner, so that the sequence of

balloons in a book tell much of a story, and it is important to mention that this is what makes them extremely important elements in comics.

The speech balloon on the left hand-side in Fig. 1.1 is a traditional text balloon because it owns white background and text in a black font type. Indeed, this is a common feature that we find on existing comic books, but nowadays there is also a trend to go beyond the traditional, e.g., the caption balloon on the right hand-side in Fig. 1.1 is not a traditional text balloon because it owns black background and text in a white font type. In addition, it is conveying the thought of a comic character.

Figure 1.1: Examples of text balloons.

The problem addressed in this dissertation has mainly to do with the extraction of text balloons from comics, no matter they own white background and black text or not. This means that such algorithm must handle balloons without making any assumption regarding color or font type. It happens that it is widely known that current algorithms are adequate for simple comics, but not for complex ones [CG15]. In fact, these algorithms do not offer guarantees in extracting balloons that do not follow the traditional feature of white background and black text.

So, the research work described in this dissertation mainly aims at exploring the leading idea of identifying text snippets in each comic book page, regardless of whether they are enclosed in balloons or not. Therefore, the problem statement that is central to this dissertation is as follows:

> *Is it possible to extract text balloons from comic book pages without making any assumptions regarding color depth of the image, orientation and language of the text?*

As will be shown throughout the dissertation, it is possible to extract text snippets and balloons from comic book pages, without making any assumptions regarding color depth of the image, orientation and language of the text, since we are capable of computing corners to detect the presence of text. Corners are features present in almost every text letter, and therefore, they can be used to filter out regions associated to text balloons.

1.3 Publications

The research work underlying this dissertation has also originated the following scientific article, which will be soon submitted to a journal for publication:

Extracting Speech Text from Comics

Pedro D. C. R. Inácio and Abel J. P. Gomes. 2016. Extracting Text Balloons from Comic Books using Corner Detection. *ACM Transactions on Graphics* (to be submitted), 2016.

Abstract. Extracting text balloons has been a challenge in digital comics, in particular for complex comic books. The challenge comes from the fact that there is no general extraction algorithm in the literature capable of handling any text balloons without making any assumption regarding color depth of the image, orientation or language of the text. Even worse, it is the fact that the comics art evolves over time, so that there is some degree of unpredictability associated to comics. This means that, an algorithm may work well for comic books released twenty years ago, but not so well for current comic books, even considering they belong to the same category or series. This paper proposes an algorithm capable of identifying text balloons in comic book pages. The proposed algorithm, here called CCD (components and corners detection), relies in the concept of corner detection to identify text snippets inside balloon candidates. So, after discarding a significant number of regions that are not considered as tentative text balloons for one reason or another, we look at the shape of the holes of the remaining regions to check if they still hold a significant number of corners capable to make a candidate be classified as text balloon.

1.4 Organization of the dissertation

This dissertation has been structured in a regular manner, including a literature review and a proposal of a new algorithm for extracting text snippets and balloons from digital comic books. More specifically, this dissertation is organized as follows:

- **Chapter 1**. This chapter makes an brief introduction to research work underlying the dissertation itself. In particular, one explains the motivation in carrying out the design and development of algorithms to extract text balloons from digital comics. In this respect, the problem statement put forward in this chapter plays a central role in all the way that has led to the writing of this dissertation.
- **Chapter 2**. This chapter reviews the algorithms for the extraction of text balloons from comics books using digital processing techniques.
- **Chapter 3**. This is the core chapter of the dissertation. It describes the proposed algorithm to extract text snippets and balloons from comic book pages, including testing results, as well as a brief comparison to other algorithms. As usual, the algorithm is based on digital processing techniques, but it has the advantage of being applicable to simple and complex comics.
- **Chapter 4**. In this chapter, we draw the most relevant conclusions about the accomplished research work, with a few open issues put forward for future work.

As a concluding remark, it is convenient to mention that this dissertation fits in the scope of computer graphics and image processing and analysis.

Chapter 2

Extracting text balloons: a literature review

This chapter presents a review of literature concerning the extraction of text balloons comic books, namely speech balloons and captions, although there are only a few studies in the literature of this emerging field of research, as is the case of digital comics.

2.1 Introduction

There was once a time that, if we wanted to read a comic book, we would have to purchase the paper book. Nowadays, with the information era transforming our ways of living, the world is becoming increasingly digital and interconnected. To this it is not strange the fact that such paper books started appearing also available in digital formats (i.e., e-books). In fact, pictures studios started by simply distributing digitized versions of the paper books. But, with the panoplia of mobile devices of varying sizes in the market, it was figured out that simply displaying comic book pages as a whole in distinct screen formats was not a good idea, because it could lead readers to not choose a mobile device to have fun in reading comics. Since then, the need to study and to accurately extract elements (e.g., panels, text balloons and comic characters) from comic books has become relevant.

Panel extraction has been mainly studied to ease the reading in a panel-to-panel sequence. But, *text balloons* end up playing a more important role than panels because they include text in the flow of a story for the sake of a better storytelling. Additionally, *comic character* (or persona) extraction is an important step towards a complete segmentation of comic book pages, and towards a complete understanding of how elements are used to compose and maintain a story; hence they have recently started being studied as well.

As widely known, most of the algorithms are adequate for simple comics, but not for complex ones [CG15]. Simple comics (see Fig. 2.3) involve newspaper comic strips and other simple comics with low level of art detail, which maintain a certain regularity in terms of layout. In turn, complex comic books (see Fig. 2.1) involve more complex level of art detail, in order to represent the beauty and the realism of scenery and storytelling, so that comic elements may overflow the limits or canons of the traditional art; for example, a balloon may extend over two comic pages, a page shows off itself empty (see Fig. 2.2), or a new type of balloon is introduced to enrich how flows the comic book.

Figure 2.1: A complex comic book page taken from [Ben15a].

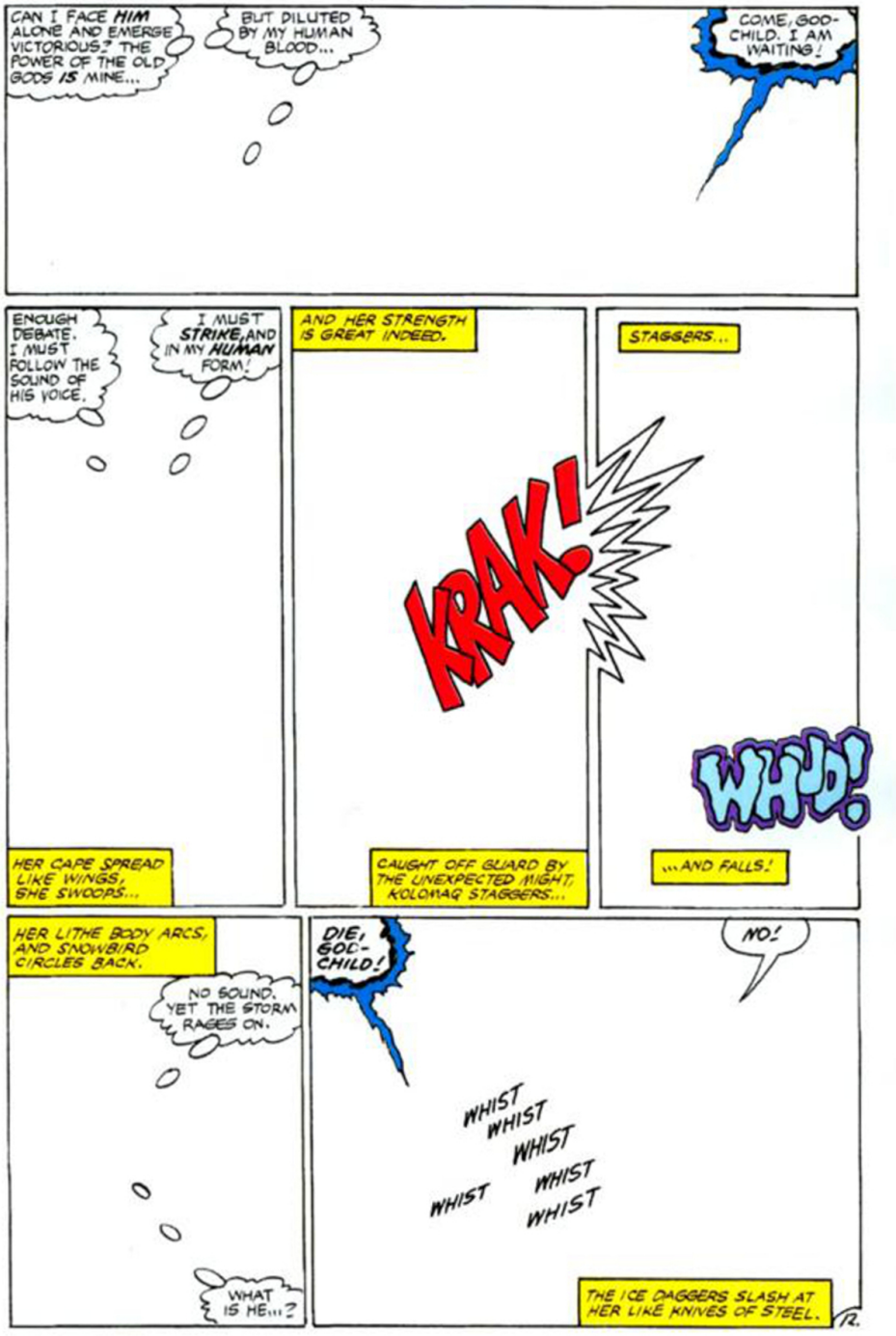


Figure 2.2: A pictureless comic book page taken from [Byr84].

Figure 2.3: A simple comic strip taken from [Dav].

2.2 Comics vocabulary

A comic book page is a graphic document composed of a set of elements to represent art an text information [Rig14]. Indeed, this is a common feature that we find on existing comic books. Therefore, and depending on the application goal we can have in mind, we may extract distinct comic elements from a comic book, namely: panels, text balloons, comic characters, among others.

A comic book page can be divided into a few categories of comic elements, which altogether are used to compose and maintain the flow of a story. So, for a better understanding of the vocabulary [Mac93, Mar07, Pie] used in this dissertation, some of those comic elements are described and illustrated below.

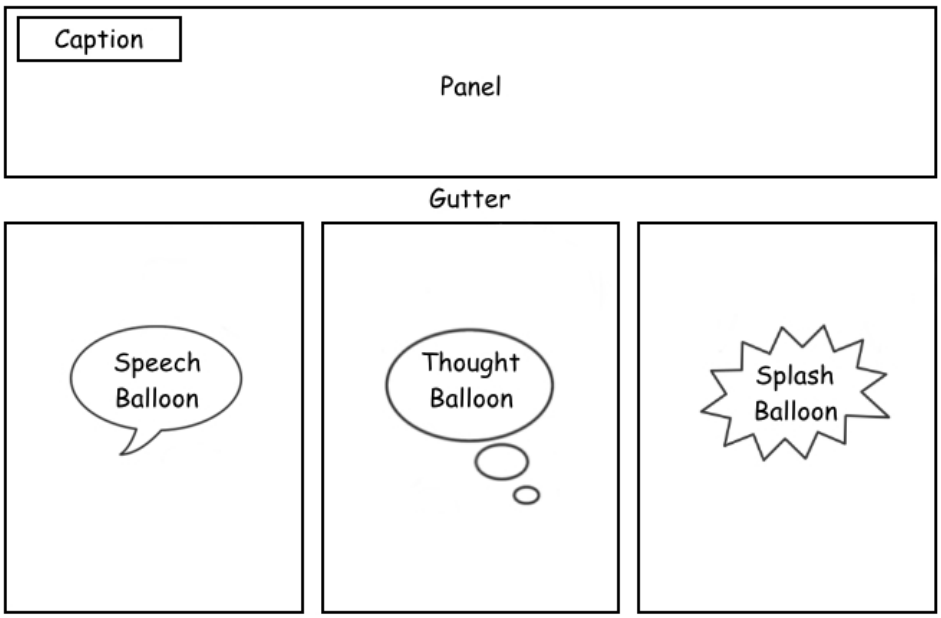


Figure 2.4: Comic elements of a comic book page.

Panel. This is a rectangular or a squared shaped element that contains a segment of the flow of a story. Nowadays, it is important to mention that a panel is allowed to be of any shape and any size, even between panels of the same page.

Gutter. This is a white element that encloses panels. Nowadays, it is important to mention that a gutter is allowed to be of another distinct color, even between pages of the same book.

Caption. This is a rectangular or a square shaped element that conveys narrative information, e.g., time and place. Nowadays, it is important to mention that a caption is allowed to convey thoughts of a character, which allows the reader to have an insight of the story, while, at the same time, it is kept away from its current flow.

Balloon. This is an element that is allowed to be of distict shapes, depending on the information that has to be conveyed (e.g., normal speech, thoughts or yelling). Still nowadays, the most common types of balloons are the following:

- *Speech Balloon*. This is a round shaped element, with a pointing tail, that conveys normal speech of characters. Nowadays, it is important to mention that multiple speech balloons are allowed to be joined if they belong to the same speech process of a comic character. In addition, if a comic character says two or more ideas separately, expressed in row, their balloons are allowed to be linked with thin connectors (see Fig. 2.1).
- *Thought Balloon*. This is a cloud-like shaped element, with a tail composed by small bubbles of decreasing size, that conveys thoughts of characters.
- *Splash Balloon*. This is a jagged shaped element, with a pointing tail, that conveys the effect of yelling of a character.

Traditionally, text balloons own white background and text in a black font type. Indeed, this is a common feature that we find on existing comic books, but nowadays there is also a trend to go beyond the traditional. Recall that a caption is a not a traditional text balloon, but in addition to its narrative role, nowadays its usage is allowed to convey thoughts of characters. Additionally, while there are sets of rules established by picture studios, e.g., DC [DC] or Marvel [Mar], in relation to font types, some editors are not familiar or not interested in them [Pie]; as a consequence, the text inside balloons can appear in any color or in any font type.

2.3 Extracting text balloons

Extracting text balloons has mostly relied so far on two categories of algorithms, namely: *connected component-based* and *edge-based*. Connected component-based algorithms try to identify and locate text balloons by relying on the assumption that traditionally

text balloons own white background, or in addition that they own a similar background to the one of the page. In turn, edge-based algorithms try to identify and locate text balloons by relying in an edge map and a text area detector.

2.3.1 Connected component-based algorithms

Guo et al. [GKSH06] have introduced an algorithm, called borders' connected components deletion, that relies on the assumption that text balloons are always connected to panels, but not text. After the binarization of a page, the authors suggest to use a connected component analysis to find the components that are connected to panels, and then to replace the pixels of such components to white color. To continue the processing, the remaining components are validated in conformity with height and width computed ratios.

Ho et al. [hBO12] have introduced an algorithm that relies on the assumption that text ballons own a bright background. The authors suggest to convert a page to the HSV color space, where bright regions own low values in terms of saturation and high values in terms of value. To continue the processing, the remaining regions are validated in conformity with an area computed ratio. This algorithm is in line with the one introduced by Arai et al. [AT11], which relies on the assumption that text balloons own white background, but in turn, the authors suggest to use a graylevel between 224 and 250 as a threshold to make a first selection of regions. To continue the processing, such regions are validated in conformity with height, width and area computed ratios.

Rigaud et al. [RTBO13b] have introduced an algorithm that relies on the assumption that text balloons own a similar background to the one of the page. The authors suggest to convert a page into its grayscale representation, to perform an inversion relatively to a threshold of binarization and to use a connected component analysis to find the bounding boxes of each one. The threshold is determined from the median value of the pixels of the border of the image, which are used as the representation of the background of the page. If it is figured out that the threshold is much closer to a black graylevel than a white graylevel, the image inversion is performed and the process starts again in order to always get a white background at the end of this step. To continue the processing, the connected components are validated in conformity with their height by a K-means algorithm, which classifies them into three categoried, namely: panels (i.e., the highest components), text (i.e., the most numerous) or noise (i.e., few pixels height).

Ranjini et al. [RS13] have introduced an algorithm that relies on the assumption that text balloons own white background. The authors suggest to perform a binarization of the page, namely by using the channel of blues of the RGB color model, relatively to a threshold that is determined from the average value of the pixels of that channel, to remove noise using median filter and to use a connected component analysis. To continue the processing, the connected components are validated in conformity with

an area computed ratio.

Correia et al. [CG15] have introduced an algorithm that relies on the assumption that text balloons own a bright background. The authors suggest to convert a page into its grayscale representation, to find the boundaries of objects using Sobel edge-based algorithm and to use a fill to find connected components. To continue the processing, the connected components are validated in conformity with height, width and area computed ratios, and afterwards in conformity with the average value of its luminance histogram.

2.3.2 Edge-based algorithms

Rigaud et al. [RTBO13a] have introduced an algorithm that relies on the assumption that sometimes the contour of a text balloon is not completely drawn, but overall, the location of the text turns out to be a good clue to guess where the balloon is. The authors suggest to handle closed and opened balloons together by using active contours, i.e., snakes. To locate text, it was introduced an algorithm that relies on the assumption that text balloon background is identical for all balloons inside the same page [RKdW+13]. To continue the processing, text lines are grouped into text areas, being the convex hull of each one used to initialize a snake. Then, the suggestion is to try to fit a closed contour around such text areas by attempting to minimize the energy associated to a snake, which is defined as the sum of three terms, namely: internal energy, external energy and text energy. Briefly, the internal energy aims to control deformations made to a snake, the external energy aims to control the fitting by using an edge map, and finally the text energy aims to give the snake a push towards to the place where the balloon possibly is.

2.3.3 Brief analysis

Connected component-based algorithms rely on the assumption that traditionally text balloons own white background, or in addition that they own a similar background to the one of the page. Thing is, they fail when a book page appears with text balloons that own distinct background colors between them, and in addition, with a background distinct to the one of the page (see Fig. 2.5). In turn, edge-based algorithms rely on the assumption that sometimes the contour of a text balloon is not completely drawn. Interestingly, they use an edge map that controls the fitting of the contour. Thing is, they fail when a page appears, e.g., with text balloons linked with thin connectors (see Fig. 2.1) because the contour seems that it does not consider such connectors.

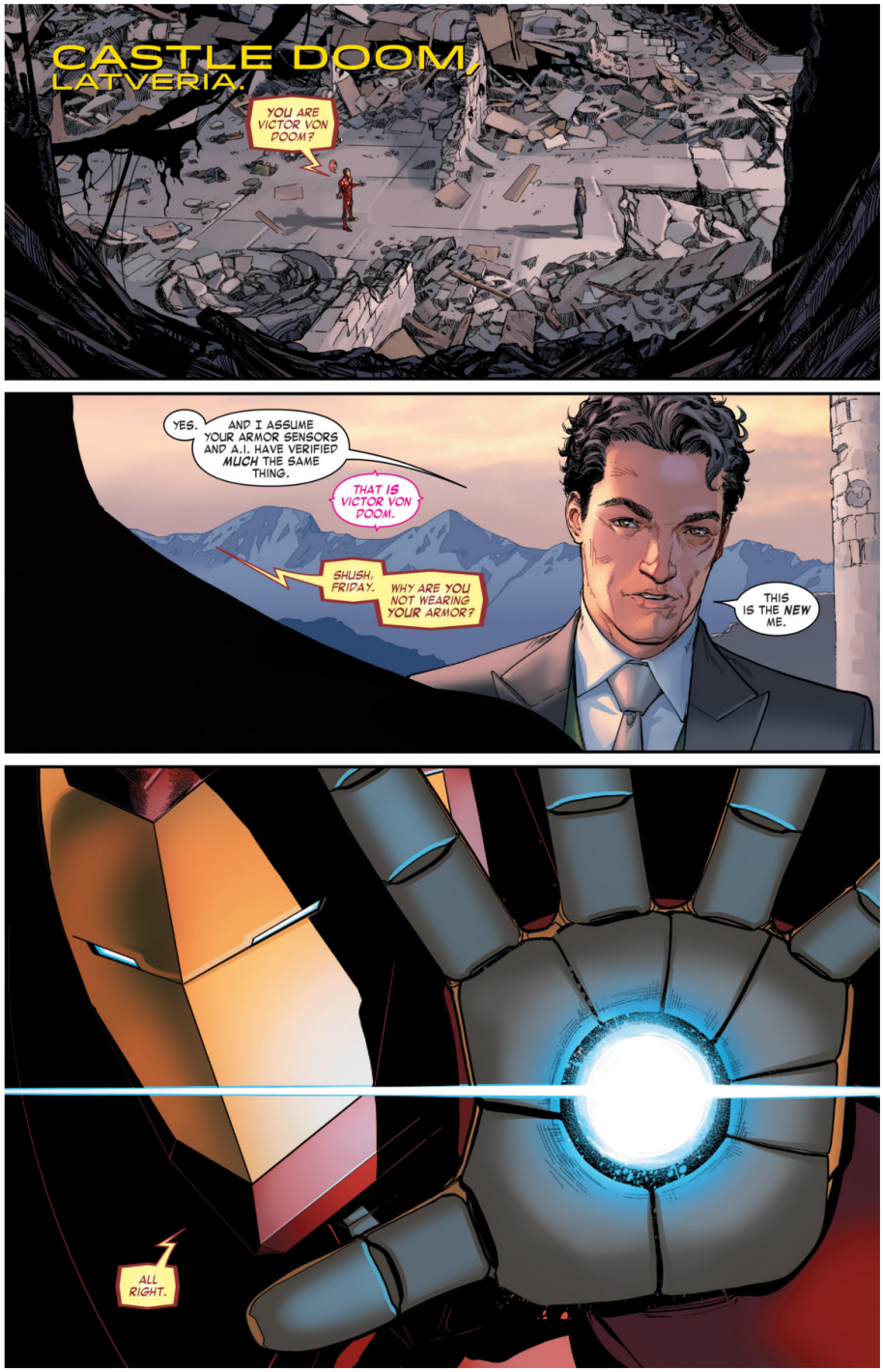


Figure 2.5: An Iron Man comic book page taken from [Ben15b] showing text balloons that own distinct background colors between them.

2.4 Concluding remarks

This chapter has reviewed the relevant literature about digital comic books. In the next chapter, we will introduce our algorithm that aims to perform the extraction of text balloons from comic books. The only assumption made by the algorithm is that comic book pages are composed by a set of closed regions, including balloons. This new text balloon extraction algorithm can be used for both simple and complex comics, and without making any assumptions regarding color depth of the image, orientation and language of the text.

Chapter 3

Extracting text balloons

This chapter describes a novel image processing-based algorithm to extract text balloons from comic book pages. The leading idea of the algorithm relies in the concept of corner detection to identify text snippets inside balloon candidates. Recall that the corner detector used is a technique introduced by Shi and Tomasi [ST94] in the general context of digital image processing and analysis, but not in digital comics. This new text balloon extraction algorithm can be used for both simple and complex comics.

3.1 Introduction

As widely known, current text balloon extraction algorithms are adequate for simple comics, but not for complex ones [CG15]. In fact, these algorithms do not offer guarantees in extracting balloons that do not follow the traditional feature of white background and black text. However, there is a current trend to overcome constraints of this sort, as it is the case of the algorithms introduced by Rigaud et al. [RTBO13b], assuming that text balloons own a similar background to the one of the page, or by Ho et al. [hBO12] and Correia et al. [CG15], assuming that text balloons own a bright background.

Our algorithm described in this chapter is in line with recent trends. More specifically, our algorithm starts from the simple assumption that a comic book page is composed of a set of closed regions. No assumptions regarding color depth of the image, orientation and language of the text are taken into consideration.

The algorithm belongs to the category of connected component-based techniques, but edge-based techniques are also used to better accentuate contours and to compute corners. Recall that corners are features present in almost every text letter, Therefore, they will be used to filter out regions associated to text balloons. Such an algorithm is here called CCD (components and corners detection) algorithm.

3.2 CCD algorithm: overview

In general terms, the CCD algorithm consists of two main steps:

- *Finding of connected components*. This step aims to identify and locate all the closed regions (or connected components) of a comic book page.

- *Finding of tentative text balloons*. This step aims to identify and locate only the regions that are associated to text balloons, in conformity with a few criteria regarding size, area and content of each region. The detected corners play an important role in this step.

Therefore, each step discards a number of regions, i.e., those that are not considered as tentative text balloons for one reason or another.

3.3 Finding Connected Components

The connected components (or closed regions) of each comic book page are extracted according to the following algorithm:

1. Convert comic book page to grayscale.
2. Find boundaries (i.e., edges) of objects using Sobel's edge-based algorithm.
3. Find corners of objects using Shi-Tomasi algorithm.
4. Find the negative of the Sobelized grayscale image.
5. Extract connected components using a labeling algorithm.

3.3.1 Conversion of each comic book page to grayscale

To convert each comic book page into a grayscale representation, we have to determine the luminance of each RGB pixel, which can be seen as a measure of brightness when adjusted for the human vision. The human eye is much more sensible to brightness variations rather than color variations due to presence of a bigger number of rods (about 75 to 150 milions) than cones (6 to 7 milions) [GW07].

The YCbCr color space has turned out to be one of the most popular color spaces. In addition, it has the advantage of taking into consideration the fact that the human eye is much more sensible to brightness rather than color in the grayscale conversion, which has much to do with how the human eye perceives object contours in images. This color space splits the RGB information also into three channels, Y, Cb and Cr, so that Y represents the original image in grayscale (see Fig. 3.1), while Cb and Cr channels represent the chrominance of blues and reds. As usual, the grayscale image can be determined by computing the value of Y for each pixel as follows:

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{3.1}$$

Figure 3.1: Batman [MD07] book page image (left); and its representation in grayscale (right).

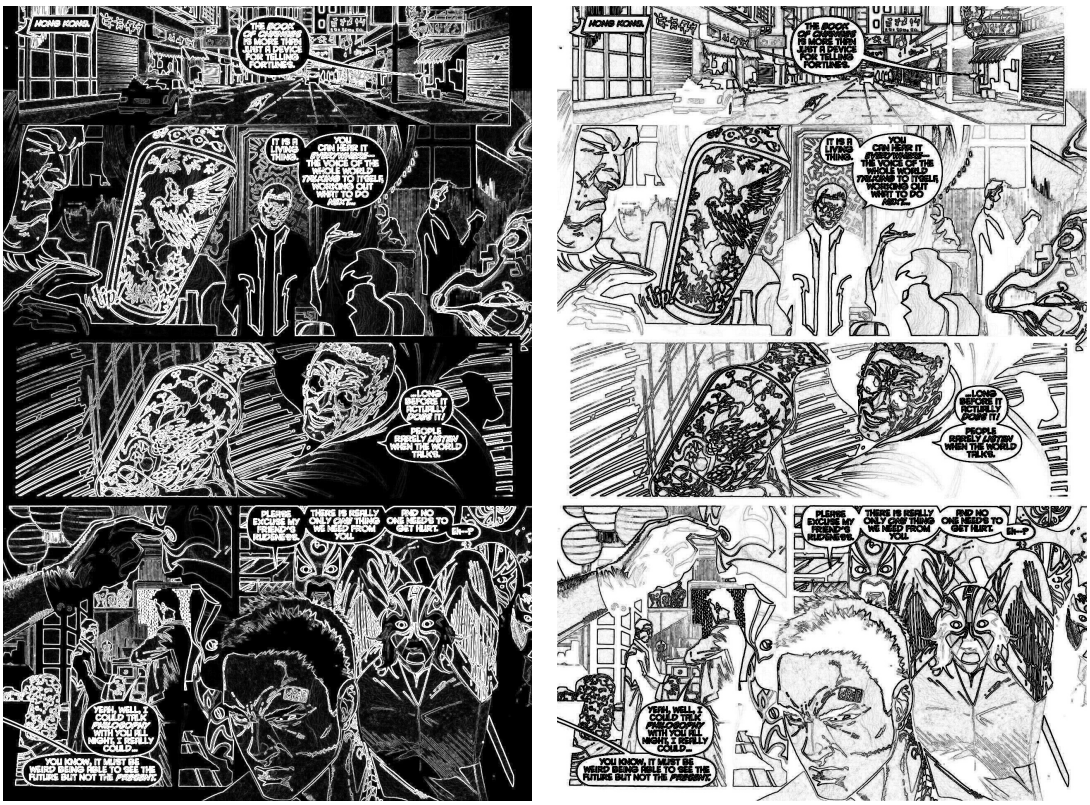


Figure 3.2: Sobelized comic book page image (left); and its negative representation (right).

By the way, if it is known beforehand that the comic book page is already in grayscale, the conversion given by Eq. (3.1) can be skipped.

3.3.2 Find boundaries of objects using Sobel's edge-based algorithm

Edge-based algorithms try to identify boundaries of objects in a given image by locating its discontinuities, which turns out to be nothing more than abrupt changes in pixel intensities inside a small neighborhood. In fact, such changes occur on boundaries of objects.

As known, there are a number of edge-based algorithms available in the literature [MA09], but we are aware that any choice has to have in mind a few factors, e.g., edge orientation and response to distinct types of intensity changes. Among those algorithms, we chose the Sobel operator because of its simplicity to detect edges and their orientations, as well as its adequacy to accentuate object contours (see Sobelized image on the left-hand side of Fig. 3.2).

The Sobel edge detector algorithm essentially computes the intensity gradient in the x and y directions, i.e., the intensity variations in the neighborhood of each pixel in each one of those two directions. For that purpose, Sobel operator convolves two 3×3 kernels with a grayscale image, one kernel for x direction and another for y direction. These x and y direction kernels are applied to the 3×3 neighborhood of each pixel, and are as follows:

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.2}$$

The gradient components (i.e., discrete partial derivatives) at a given pixel (x, y) are given by the following expressions:

$$G_x = [I(x-1, y+1) + 2 \times I(x, y+1) + I(x+1, y+1)] \tag{3.3}$$
$$-[I(x-1, y-1) + 2 \times I(x, y-1) + I(x+1, y-1)] \tag{3.4}$$

and

$$G_y = [I(x+1, y-1) + 2 \times I(x+1, y) + I(x+1, y+1)] \tag{3.5}$$
$$-[I(x-1, y-1) + 2 \times I(x-1, y) + I(x-1, y+1)] \tag{3.6}$$

where I stands for the intensity of a given pixel.

The gradient components are then combined in order to compute the gradient magnitude as follows:

$$G = |G_x| + |G_y| \tag{3.7}$$

Recall that Sobel operator has the effect of thickening edges. This is, if an edge is one pixel thick, it becomes thicker, because each kernel detects two consecutive abrupt changes during the processing. This behaviour is adequate to guarantee that the boundary of each object is well defined. Besides, in the presence of eventual small gaps on the boundary of a given object, such gaps will be narrowed or even eliminated.

3.3.3 Find corners of objects using Shi-Tomasi algorithm

Corner detection algorithms identify corners of objects in a given image by locating its discontinuities, i.e., abrupt changes in pixel intensities inside a small neighborhood, but in all directions. In fact, such changes occur on boundaries of objects. Moreover, and more importantly, they are features present in almost every text letter, as illustrated in Fig. 3.3. Therefore, they can be used to identify and locate regions that are associated to text balloons.

Figure 3.3: A text letter alongside its corners marked as circles of radius 3 for a better visualization.

As known, there are a number of corner detection algorithms available in the literature [HS88, ST94], but we are aware that any choice has to have in mind a few factors, e.g., cornerness scoring function and thresholding criteria. Among those algorithms, we chose the one introduced by Shi and Tomasi [ST94], because of its simplicity. In fact, to create a scoring map with the same size (resolution) as the original page image, we have only to determine the lowest eigenvalue of the covariance matrix $M_{i,j}$ at each pixel (i, j), which is given by the following expression:

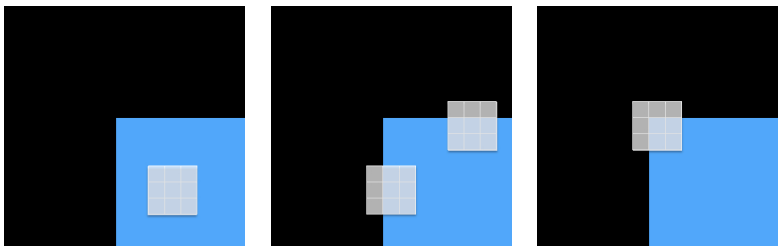
$$M_{i,j} = \begin{bmatrix} \sum G_x G_x & \sum G_x G_y \\ \sum G_x G_y & \sum G_y G_y \end{bmatrix} \tag{3.8}$$

where G_x and G_y stand for the discrete gradient components in x and y directions, respectively, considering that we are using a 8-neighborhood (i.e., a 3x3 mask) around each pixel. Recall that these gradient components have already been calculated before to identify boundaries (i.e., edges) of objects using Sobel edge-based algorithm. The matrix $M = M_{i,j}$ is then used to compute the scoring value of the pixel (i, j), which

turns out to be nothing more than its lowest eigenvalue. Recall that the eigenvalues of a matrix are given by the roots of its characteristic polynomial as follows:

$$p(\lambda) = det(M - \lambda I) \tag{3.9}$$

where I stands for the 2×2 identity matrix.



(a) (b) (c)

Figure 3.4: A pixel and its 8-neighborhood in grey: (a) a flat pixel; (b) a pixel alongside an edge; and (c) a pixel at a corner.

Based on the eigenvalues of M, the following can occur:

1. When λ_1 and λ_2 are small (approximately zero), that means that a pixel is flat, i.e., a pixel without a change in all directions (see Fig. 3.4(a)).
2. When λ_1 is large and λ_2 is small (approximately zero), or vice versa, that means that a pixel is an edge, i.e., a pixel without a change along the edge direction (see Fig. 3.4(b)).
3. When λ_1 and λ_2 are large, that means that a pixel is a corner, i.e., a pixel with changes in all directions (see Fig. 3.4(c)).

To assure that only strong corners are retained inside the scoring map, a non-maxima suppression is applied in order to retain local maxima in a 3×3 neighborhood. Then, a threshold τ is computed by applying the weight $w = 0.21$ to the corner that has the highest score inside the scoring map S as follows:

$$\tau = w \,.\, \max[S(i,j)] \tag{3.10}$$

where w was obtained empirically after a series of experiments in order to reduce the number of false positives, i.e., incorrectly identified and located text balloons. Finally, a separate image, called *corners* image is created, which has the same size (resolution) as the original page image, where all the corners of the scoring map below the threshold are marked as zero (or black) pixels, and the ones above are marked as white pixels.

3.3.4 Find the negative of the Sobelized grayscale image

The negative of the Sobelized grayscale image of each comic book page is important to determine the size and the area of each object in such image (see image on the

right-hand side of Fig. 3.2).

To compute the negative image, use the following expression:

$$N(i,j) = 255 - I(i,j), \quad \forall i,j \tag{3.11}$$

3.3.5 Extraction of connected components using a labeling algorithm

Connected component labeling algorithms try to identify and locate connected regions composed of a set of adjacent pixels. Each connected component (or region) turns out to be composed of a set of non-zero pixels delimited by a set of zero (or black) pixels. Each component corresponds to the interior of each image object, while its contour or boundary is given by such a set of black pixels. We extract the components of the negative image, as the one shown in the middle of Fig. 3.5, by identifying sets of connected non-zero pixels; its distinct regions appear in distinct colors on the right hand-side of Fig. 3.5.

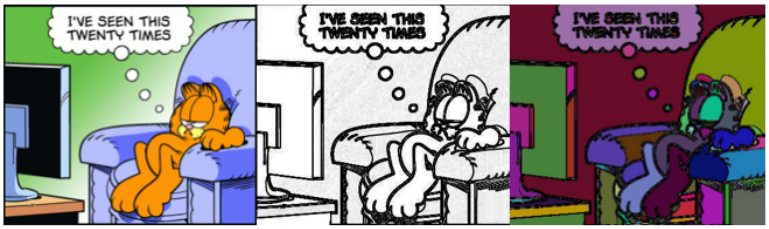


Figure 3.5: A panel of a comic strip taken from from [Dav] (left); its negative (middle); and it connected components labeled with distinct colors (right).

Our algorithm follows the two-pass introduced by Wu et al. [WOS09]. Usually, two-pass connected-component extraction algorithms consist of three phases: *scanning*, *analysis*, and *relabeling*. The first pass consists of two intertwined phases, scanning and labelling, while the second pass makes the analysis and the relabeling of pixels in order to correctly extract the connected components. Therefore, our algorithm consists of the following phases:

Scanning and labeling phase. The scanning is performed using a *forward* strategy, i.e, from left to right and top to bottom. This step aims to identify non-zero pixels (or non-black pixels) in the negative image of each comic book page, labeling them temporarily in a separate image, called labeling image L, which has the same size (resolution) as the original page image. In addition, the temporary containment relationships between pixel labels and component labels are recorded in a separated data structure, called disjoint-set data structure (or union-find data structure or merge-find set).

This data structure keeps track of the set of pixels of an image divided into a number of disjoint (non-overlapping) components. This data structure was implemented as an array of labels P, and only keeps track of distinct labels (a single label for each

temporary component) and their corresponding parent labels (a single label for each temporary parent component). Note that this data structure does not contain pixels, but only distinct temporary components.

Algorithm 1 findRoot

Input: Array P and a label i
Output: Root label

$root \leftarrow i$
while $P[root] < root$ **do**
    $root \leftarrow P[root]$
end

return root

Algorithm 2 setRoot

Input / Output: Array P
Input: Label i and root label

while $P[i] < i$ **do**
    $j \leftarrow P[i]$
    $P[i] \leftarrow root$
    $i \leftarrow j$
end

$P[i] \leftarrow root$

In order to merge and group pixels of each component, we need two operations: *find* and *union*. The *find* operation (see Alg. 1) allow us to determine the parent component of a given pixel, using for that component labels (i.e., indices) and their parent labels in the union-find data structure. The *union* operation (see Alg. 2) allow us to merge two temporary components into a single one.

Therefore, iterating over the pixels of the negative image, and considering only the left and top pixels neighboring each pixel, the labeling of each non-black pixel in the labeling image is performed using the following rules:

1. If both neighbors do not exist, a new temporary singleton component label is generated for the pixel, which means that the array of labels P also increases by one. It is important to mention that such array always starts with size 1, i.e., the label '0' is used to denoted the black pixels of the negative image.

2. If only one of the neighbors exists, one has to assign its label to the pixel being processed.

3. If both neighbors exist, that means that we are eventually in the presence of a merging pixel. In this case, we determine the root parent of each neighboring pixel by calling the *find* operation (see Alg. 1), assigning then their lowest root parent label to the pixel being processed in L. If the root parent labels of both neighboring pixels are distinct, we call the *union* operation (see Alg. 2) to merge their components into one. The merge is performed by assigning the lowest parent label of both neighboring pixels to their highest parent label in P.

This labeling process of each non-black pixel is illustrated in Figs. 3.6-3.11).

Label	0
Parent label	0

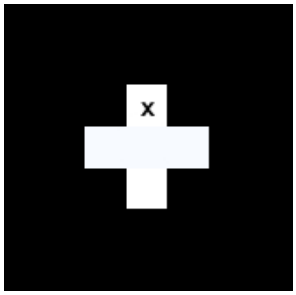


Figure 3.6: Original state of the array P before labeling any component (left); the cross indicates the pixel of L that will be labeled using the 1st labeling rule (right).

Label	0	1
Parent label	0	1

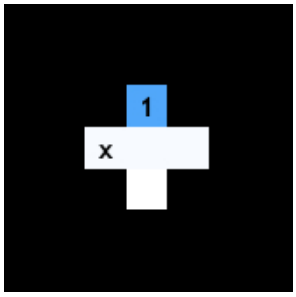


Figure 3.7: Current state of the array P after labeling of the previous crossed pixel with '1', i.e., $P[1] = 1$ (left); the cross indicates the pixel of L that will be labeled using the 1st labeling rule (right).

Label	0	1	2
Parent label	0	1	2

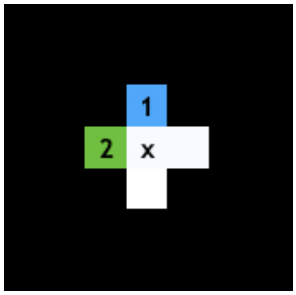


Figure 3.8: Current state of the array P after labeling of the previous crossed pixel with '2', i.e., $P[2] = 2$ (left); the cross indicates the pixel of L that will be labeled using the 3rd labeling rule (right).

Label	0	1	2
Parent label	0	1	1

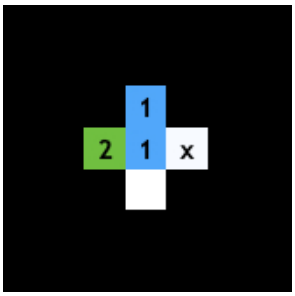


Figure 3.9: Current state of the array P after labeling of the previous crossed pixel with '1', so that no new label is added to P, but $P[2] = 1$ (left); the cross indicates the pixel of L that will be labeled using the 2nd labeling rule (right).

Label	0	1	2
Parent label	0	1	1

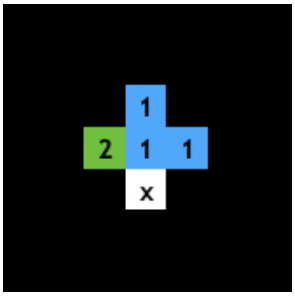


Figure 3.10: Current state of the array P after labeling of the previous crossed pixel with '1', so that no new label is added to P (left); the cross indicates the pixel of L that will be labeled using the 2nd labeling rule (right).

Label	0	1	2
Parent label	0	1	1

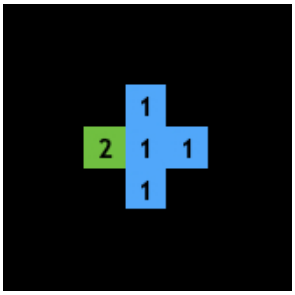


Figure 3.11: Current state of the array P after labeling of the previous crossed pixel with '1', so that no new label is added to P (left); the scanning and labeling process terminates (right).

Analysis and relabeling phase. After the scanning and labeling process, the component shown in Fig. 3.11 has two distinct temporary labels, '1' and '2', but with the same root parent label '1', which makes them equivalent. Therefore, we need to perform two operations. First, we generate the final parent label of each component, analyzing the equivalences between labels in P, by calling the *flatten* operation (see Alg. 3). Second, we relabel all the pixels in the labelling image L, so that all the pixels can have the final label of its parent, and as follows:

$$L(i,j) \leftarrow P[L(i,j)], \quad \forall i,j \tag{3.12}$$

Algorithm 3 flatten

Input / Output: Array P
Input: Size of array P

$k \leftarrow 1$
for $i \leftarrow 1$ to $size - 1$ **do**
  if $P[i] < i$ **then**
    $P[i] \leftarrow P[P[i]]$
  else
    $P[i] \leftarrow k$
    $k \leftarrow k + 1$
  end
end

Label	0	1	2
Parent label	0	1	1

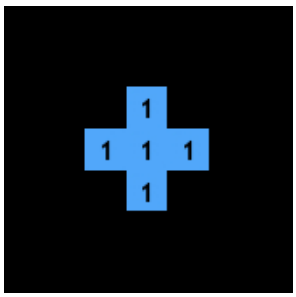


Figure 3.12: Current state of the array after the *analysis* (left); and all the pixels relabeled by the *relabeling* (right).

3.4 Finding text balloons

After determining connected components (or closed regions) of a given comic book page, we have to discard those that are not text balloons. It is clear that we risk to discard some components that are text balloons (i.e., false negatives), but we follow a strategy to reduce false negatives to a minimum as much as possible.

The text balloons of each comic book page are extracted from the set of components according to the following algorithm:

1. Discard of non-text balloons.
2. Extraction of boundaries (i.e., contours) of objects using Suzuki-Abe algorithm.
3. Shift corners.
4. Filter out text balloons.

3.4.1 Discarding non-text balloons

By analyzing some comic book pages, e.g., Fig. 3.1, we see that the majority of its regions are very small, e.g., art details, or too large, e.g., gutters, to be associated to text balloons. Therefore, they can be discarded.

Region \ Criteria	*Too small*	*Too large*
Region width	<1.5%	>50%
Region height	<1.5%	>50%
Region area	<0.01%	>20%

Table 3.1: Criteria for discard non-text balloon components.

The elimination criteria of regions take into consideration the size and area of each region relative to the size and area of the page (see Table 3.1). Therefore, if the length of each region's bounding box is less than 1,5% or is greater than 50% relative to the page width, then such region is discarded. Similarly, if if the height of each region's bounding box is less than 1,5% or is greater than 50% relative to the page height, then such region is also discarded. Finally, if each region's area (total pixels of the region) is less than 0,01% or is greater than 20% relative to the page area, then such region is discarded too.

3.4.2 Extraction of boundaries of objects using Suzuki-Abe algorithm

At this stage, we assume that components corresponding to non-text balloons have been already thrown away. In order to extract boundaries of objects in the image with the remaining putative text balloons, we make use of a boundary-following algorithm. This sort of algorithm tries to identify and locate contours of objects composed of a set of adjacent pixels along their boundaries. Each boundary turns out to be composed of a set of non-zero pixels delimited by a set of zero (or black) pixels on its right-hand side or on its left-hand side, depending on the type of a boundary.

Our algorithm follows the introduced by Suzuki and Abe [SA85], consisting in two scanning steps: *labeling* and *following*. Both scanning steps are performed using a forward strategy, i.e. from left to right and top to bottom.

Labeling step. This scanning step aims to identify non-zero pixels (or non-black pixels) and zero pixels (or black pixels) in the image, labeling them temporarily as '1' and '0' in a separate image, called labeling image L, which has the same size (resolution) as the original page image.

Following step. This scanning step aims to correctly extract each object's contour. The leading idea is to follow the boundary of each object in a pixel-after-pixel manner, relabeling pixels belonging to same contour with a distinct label in L. In addition, all

the pixels that belong to a boundary are also recorded in a separated array of arrays, called *contours*.

There are two types of boundaries, namely: outer boundaries and inner boundaries (see Fig. 3.13). A pixel is classified as a starting point of an outer boundary if it is labeled as '1' and its left-hand neighbor labeled as '0' (Fig. 3.13(left)). In turn, it is classified as a starting point of an inner boundary if it is labeled with any positive label and its right-hand neighbor labeled as '0' (Fig. 3.13(right)). In addition, if a pixel satisfies both criteria, then it is classified as a starting point of an outer boundary.

Figure 3.13: Conditions of starting point for an outer boundary (left); and an inner boundary (right).

This phase also keeps track of the number of new boundaries found and of the label of the last boundary found in the current row, using for that two integer identifiers, here called NBD and LNBD. Every time a new row starts being scanned, LNBD is always set to '1' because of a special hole boundary composed by the frame of the image. Overall, LNBD can be keeping track of the label of an outer boundary or an hole boundary, but interestingly that boundary will be the parent of NBD or a sister of NBD. Therefore, by knowing the type of LNBD and NBD it will be possible to establish hierarchical relationships between all the boundaries, and this simply by determining the parent of each one as follows:

NBD \ LNBD	*Outer boundary*	*Hole boundary*
Outer boundary	Parent of LNBD	LNBD
Hole boundary	LNBD	Parent of LNBD

Table 3.2: Criteria for determine the parent of a boundary [SA85].

The determination of the parent label of a boundary occurs in conformity with a few criteria regarding its own type and the type of the last boundary found in the current row (see Table 3.2). Therefore, if LNBD and NBD are keeping track of boundaries of the same type, then it means that the parent of NBD will be the parent of LNBD. Otherwise, if LNBD and NBD are keeping track of boundaries of distinct types, then it means that the parent of NBD will be LNBD instead. In addition, the parent of each boundary is recorded in a separated array of labels, called *hierarchy*.

Overall, the algorithm of this phase is as follows:

Algorithm 4 findContours

(1) Select one of the following:
  (a) If it is found a boundary starting point of an outer boundary (see image on the left-hand side of Fig. 3.13), increment NBD, and $(i_2, j_2) \leftarrow (i, j-1)$
  (b) If it is found a boundary starting point of an inner boundary (see image on the right-hand side of Fig. 3.13), increment NBD, $(i_2, j_2) \leftarrow (i, j+1)$, and LNBD $\leftarrow L(i,j)$ in case $L(i,j) > 1$
  (c) Otherwise, go to (4)

(2) According to the type of LNBD and NBD, determine the parent boundary of NBD (see Table 3.2)

(3) From the starting point, follow the detected boundary:
  (3.1) Starting from (i_2, j_2), look around clockwise in the neighborhood of (i, j) for the first '1' pixel, called (i_1, j_1). If none can be found, then $L(i,j) \leftarrow$ -NBD and go to (4)
  (3.2) $(i_2, j_2) \leftarrow (i_1, j_1)$ and $(i_3, j_3) \leftarrow (i, j)$
  (3.3) Starting from the next pixel of (i_2, j_2) in counterclockwise order, look around counterclockwise in the neighborhood of (i_3, j_3) for the first '1' pixel, called (i_4, j_4)
  (3.4) Change the value of $L(i_3, j_3)$ as follows:
    (a) If $(i_3, j_3 + 1)$ is a '0' pixel examined in (3.3), then $L(i_3, j_3) \leftarrow$ -NBD
    (b) If $(i_3, j_3 + 1)$ is not a '0' pixel examined in (3.3), and $L(i_3, j_3) = 1$, then $L(i_3, j_3) \leftarrow$ NBD
    (c) Otherwise, do not change $L(i_3, j_3)$
  (3.5) If (i_4, j_4) = (i, j) and (i_3, j_3) = (i_1, j_1), it means that the following is coming back to the starting point, then go to (4); otherwise, (i_2, j_2) = (i_3, j_3) then (i_3, j_3) = (i_4, j_4), and go back to (3.3)

(4) If $L(i,j) \neq 1$, then LNBD $\leftarrow |L(i,j)|$, and resume the scanning from the pixel $(i, j+1)$, until it is found a new boundary starting point

3.4.3 Shift corners

Before filter out the text balloons, the corners need to be placed on positions of contours. Recall that corners were determined from partial gradients used to identify boundaries (i.e., edges) of objects using Sobel edge-based algorithm. In turn, contours were extracted from the remaining components after the discarding of non-text balloons, but these components were extracted by identifying the interior area of each object in the negative image.

The pixels of each contour were used to create a separated image, called *contours of connected components*. By analyzing the position of corners againt pixels of contours, we see that corners are shifted between 3 to 5 pixels of a contour. Therefore, to correct this small shift, a scanning in the *corners* image is performed using a forward strategy, i.e., from left to right and top to bottom. This step aims to shift a corner to the position of its closest neighboring contour pixel, and this by finding it inside a 11×11 window in the *contours of connected components*. If none can be found, then the corner is discarded.

3.4.4 Filter out text balloons

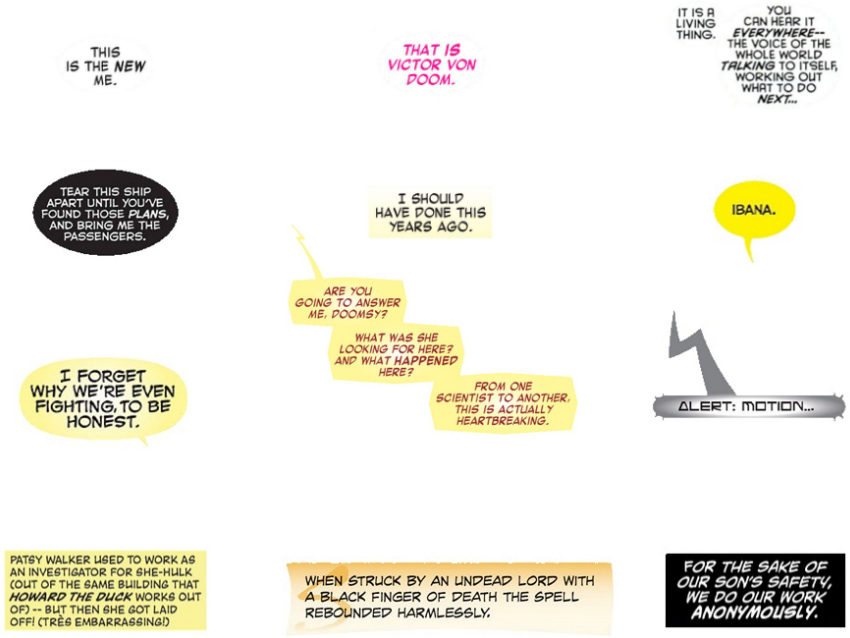


Figure 3.14: Balloons filtered out by our algorithm.

Text balloons are closed regions with holes left by text snippets. So, the filtering occurs in conformity with a few criteria regarding the content of the holes of each region. For that purpose, we use the *hierarchical* relationships between the *contours* in order to analyze parents (i.e., regions) along their children (i.e., holes). In addition, the pixels of the children contours are used to check if they match with the position of any corner in the *corners* image, and this is to find if they still hold a significant number of corners capable to make a candidate be classified as a text balloon. Therefore, a candidate contour is classified as a text balloon only if it satisfies the following criteria:

1. A contour is parent of at least one child contour. In addition, children contours can not be parents of any other contour.

2. If a contour is parent of exactly one child, such child holds at least 4 corners.

3. If a contour is parent of more than one child, at least half of them holds at least 2 corners.

The contours classified as text balloons are used to create a separated image, called *output*, where each one is drawn with a distinct color, and being the white color reserved for the surrounding area of the contours that is determined afterwards. We know that the pixel $(0, 0)$ belongs to a corner of a special boundary composed by the frame of the image. Therefore, the surrounding area of the contours is determined by applying a standard flood fill from such pixel in the *output* image. Then, to get a *mask* of the filled balloons we have only to determine its negative.

The text balloons shown in Fig. 3.14 are regions that were filtered out by our algorithm.

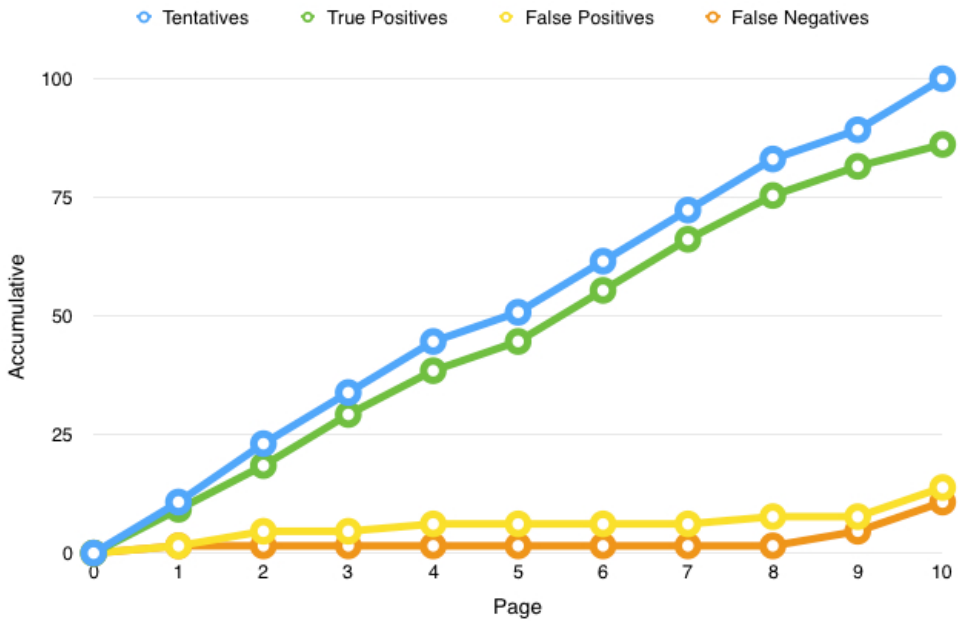
3.5 Results

Our algorithm was designed in C++, along with the version 3.1 of openCV. The results here shown (see Fig. 3.15 - 3.26) were computed by running the algorithm on a MacBook Pro running MacOS X El Capitan operating system, powered by an 2.53 GHz Intel Core 2 Duo processor, with 4GB RAM DDR3, one NVIDIA GeForce 9400M graphics card, and on a dataset with some degree of unpredictability. Such dataset is composed by 120 comic book pages of distinct resolutions (between 1080×1660 to 3975×3056), previously selected from 12 comic books of 2 distinct publishers, where only the first 10 pages containing the flow of the story were used.

The number of text balloons was seen and counted before processing a comic book page. The here called 1st components is the number of extracted components by identifying sets of connected non-zero pixels in the negative image, and the 2nd components is the number of the remaining components after the discarding of non-text balloons. Therefore, the tentatives is the number of detections produced by the algorithm in the 2nd components. A true positive is a correctly identified and located text balloon as a whole, while a false positive is an incorrectly identified and located text balloon. A false negative is a text balloon that was missed by the algorithm. In addition, the time taken to process each comic book page, after reading it, was also measured in seconds.

Based on the above definitions, and to measure the performance of the algorithm relatively to the process of finding text balloons for each book, two measures were used, namely: precision and recall [OD08]. The precision measures the percentage of detections produced by the algorithm that are true text balloons of a comic book. In turn, the recall measures the percentage of true text balloons that are retrieved of a comic book.

$$Precision = \frac{TP}{TP + FP} \times 100 \qquad Recall = \frac{TP}{TP + FN} \times 100 \tag{3.13}$$



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	7	6	7	6	4	7	7	6	6	7
1st Components	18544	15008	13813	20106	10826	12588	25420	22974	16823	10081
2nd Components	161	99	79	73	82	96	283	166	54	67
Tentatives	7	8	7	7	4	7	7	7	4	7
True Positives	6	6	7	6	4	7	7	6	4	3
False Positives	1	2	0	1	0	0	0	1	0	4
False Negatives	1	0	0	0	0	0	0	0	2	4
Time (seconds)	1,1	1,1	1,2	2,0	1,0	1,1	1,2	1,2	1,8	1,1

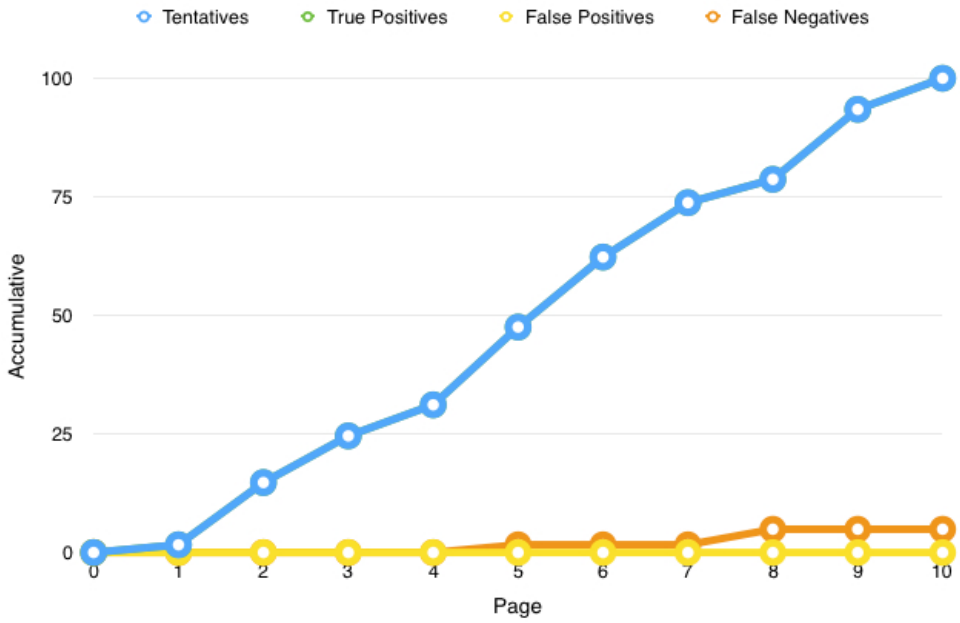
Overall Performance	
Precision	Recall
86,15%	88,89%

Figure 3.15: Results for Batman #670 [MD07].

	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	2	5	5	6	8	4	4	4	8	6
1st Components	24095	13541	11032	6828	24619	14737	25490	13956	10384	7882
2nd Components	172	33	37	46	118	58	138	60	41	40
Tentatives	5	6	6	6	8	4	4	3	9	5
True Positives	2	5	5	6	8	4	4	2	8	5
False Positives	3	1	1	0	0	0	0	1	1	0
False Negatives	0	0	0	0	0	0	0	2	0	1
Time (seconds)	1,3	2,0	3,5	1,2	1,5	1,2	1,3	1,1	1,1	1,1

Overall Performance	
Precision	Recall
87,50%	94,23%

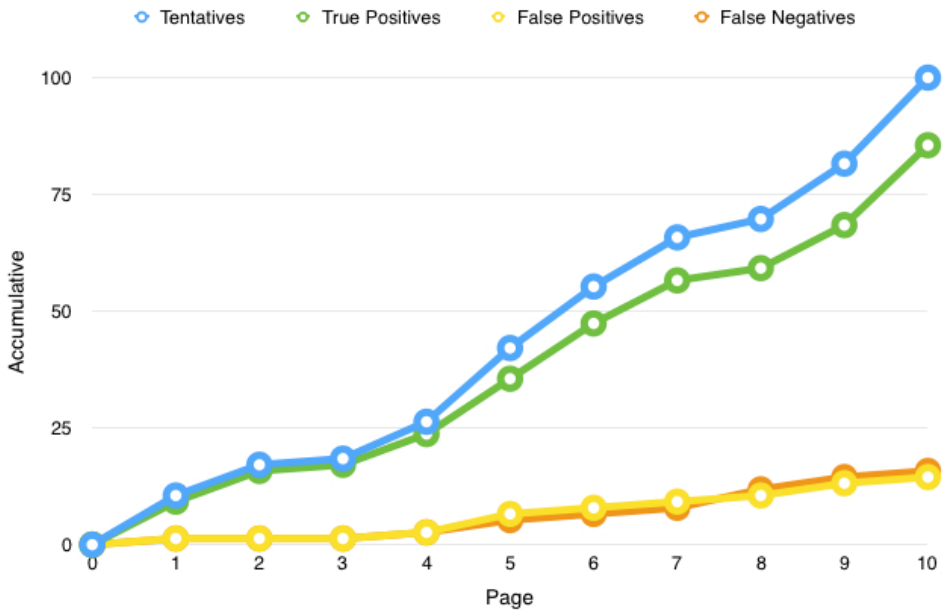
Figure 3.16: Results for Batman #671 [MD08a].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	1	8	6	4	11	9	7	5	9	4
1st Components	3478	7013	13653	6092	14486	7915	4578	2968	4052	2546
2nd Components	37	49	101	24	84	22	43	20	21	31
Tentatives	1	8	6	4	10	9	7	3	9	4
True Positives	1	8	6	4	10	9	7	3	9	4
False Positives	0	0	0	0	0	0	0	0	0	0
False Negatives	0	0	0	0	1	0	0	2	0	0
Time (seconds)	1,1	1,0	1,1	2,1	1,1	1,0	1,0	1,1	1,0	1,1

Overall Performance	
Precision	Recall
100%	95,31%

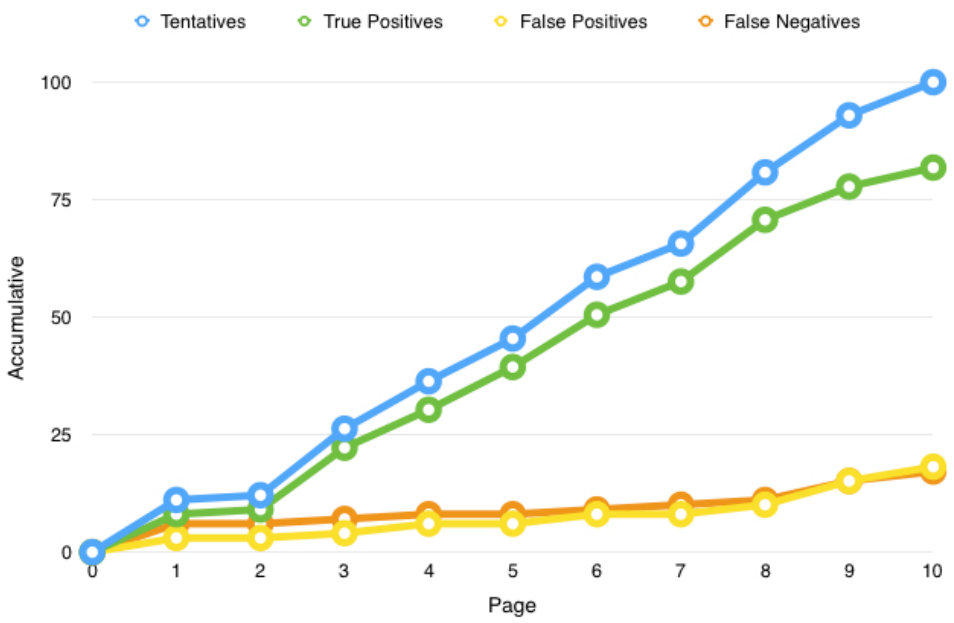
Figure 3.17: Results for Batman #672 [MD08b].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	8	5	1	6	11	10	8	5	9	14
1st Components	12112	17195	12144	29075	18818	20891	16095	14030	11914	15045
2nd Components	74	115	31	230	183	122	130	83	105	148
Tentatives	8	5	1	6	12	10	8	3	9	14
True Positives	7	5	1	5	9	9	7	2	7	13
False Positives	1	0	0	1	3	1	1	1	2	1
False Negatives	1	0	0	1	2	1	1	3	2	1
Time (seconds)	0,9	0,9	0,8	1,5	1,7	0,9	1,1	0,8	1,6	2,1

Overall Performance	
Precision	Recall
85,53%	84,42%

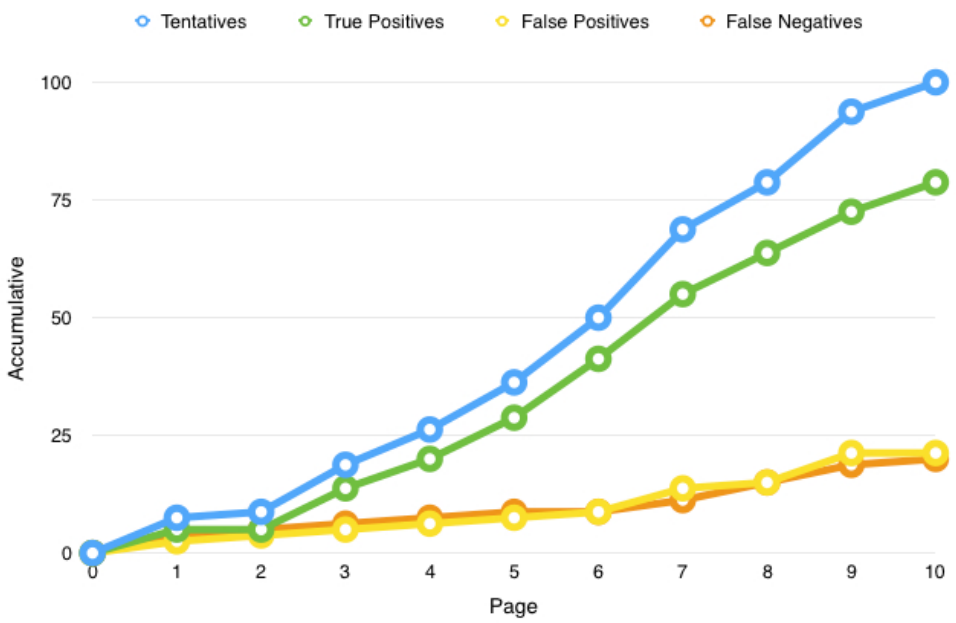
Figure 3.18: Results for Superman #3 [Jur15a].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	14	1	14	9	9	12	8	14	11	6
1st Components	16165	11413	12493	21318	12800	12157	8400	22874	23518	21817
2nd Components	180	110	114	145	139	119	68	203	257	159
Tentatives	11	1	14	10	9	13	7	15	12	7
True Positives	8	1	13	8	9	11	7	13	7	4
False Positives	3	0	1	2	0	2	0	2	5	3
False Negatives	6	0	1	1	0	1	1	1	4	2
Time (seconds)	2,5	0,8	1,3	1,6	1,6	1,2	1,4	1,6	1,6	1,2

Overall Performance	
Precision	Recall
81,82%	82,65%

Figure 3.19: Results for Superman #4 [Jur15b].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	7	1	8	6	8	10	13	10	10	6
1st Components	11819	16178	19240	27069	19229	17326	16626	11241	19444	21277
2nd Components	87	124	167	120	184	206	205	79	149	244
Tentatives	6	1	8	6	8	11	15	8	12	5
True Positives	4	0	7	5	7	10	11	7	7	5
False Positives	2	1	1	1	1	1	4	1	5	0
False Negatives	3	1	1	1	1	0	2	3	3	1
Time (seconds)	0,8	1,2	2,6	1,3	1,6	1,3	1,5	0,9	1,1	1,3

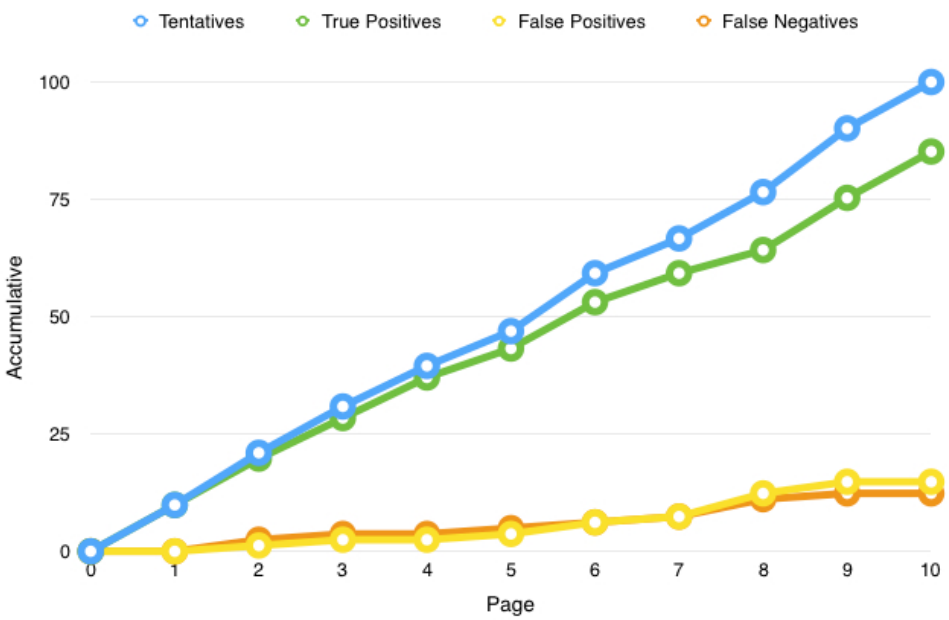
Overall Performance	
Precision	Recall
78,75%	79,75%

Figure 3.20: Results for Superman #5 [Jur15c].

	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	9	6	8	10	11	10	8	8	6	6
1st Components	5580	4742	8684	5170	9066	7598	7052	6878	7151	4376
2nd Components	69	104	112	91	93	114	122	103	159	83
Tentatives	9	7	8	10	11	9	9	8	6	7
True Positives	8	5	6	10	11	8	7	7	6	6
False Positives	1	2	2	0	0	1	2	1	0	1
False Negatives	1	1	2	0	0	2	1	1	0	0
Time (seconds)	0,9	0,9	0,9	1,0	0,9	0,9	1,2	0,9	1,0	1,0

Overall Performance	
Precision	Recall
88,10%	90,24%

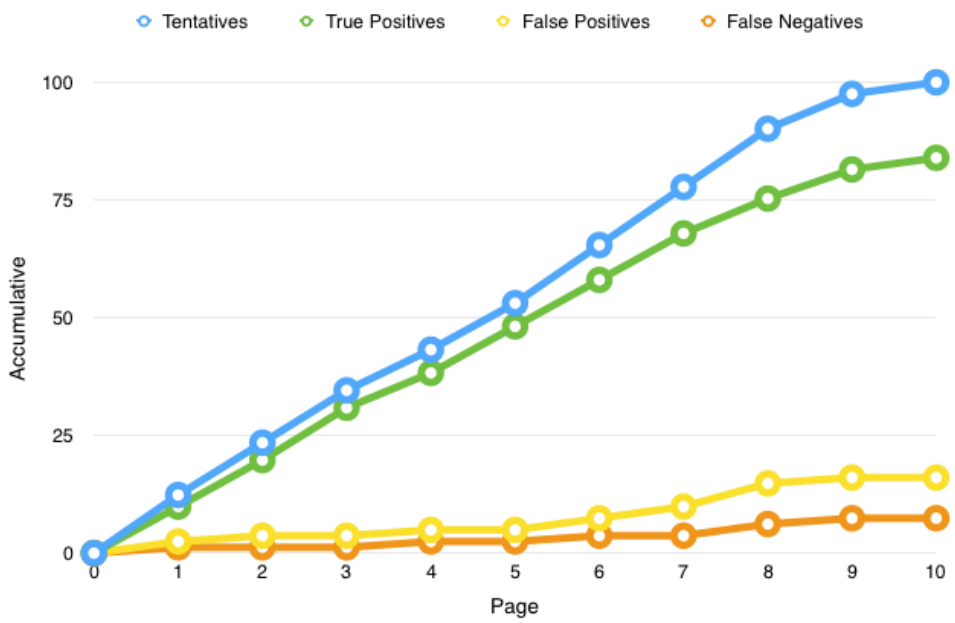
Figure 3.21: Results for Spiderman #2 [WZ16a].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	8	10	8	7	6	9	6	7	10	8
1st Components	10626	9871	10837	8797	6065	10713	12430	15939	16570	7968
2nd Components	92	188	179	144	97	108	130	226	194	96
Tentatives	8	9	8	7	6	10	6	8	11	8
True Positives	8	8	7	7	5	8	5	4	9	8
False Positives	0	1	1	0	1	2	1	4	2	0
False Negatives	0	2	1	0	1	1	1	3	1	0
Time (seconds)	2,4	1,9	1,0	1,6	0,8	1,1	1,4	2,1	2,1	2,1

Overall Performance	
Precision	Recall
85,19%	87,34%

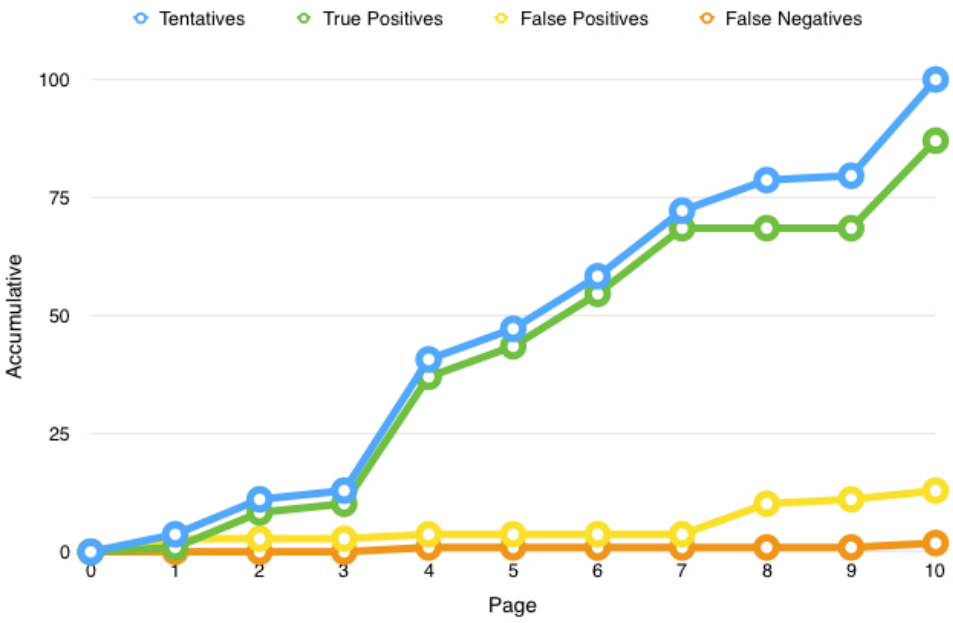
Figure 3.22: Results for Spiderman #3 [WZ16b].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	9	8	9	7	8	9	8	8	6	2
1st Components	3591	4746	4801	3922	3547	5802	4660	7280	7570	4537
2nd Components	29	46	46	76	60	64	60	77	137	74
Tentatives	10	9	9	7	8	10	10	10	6	2
True Positives	8	8	9	6	8	8	8	6	5	2
False Positives	2	1	0	1	0	2	2	4	1	0
False Negatives	1	0	0	1	0	1	0	2	1	0
Time (seconds)	0,9	0,9	1,0	0,8	0,9	1,1	1,2	1,1	1,6	0,9

Overall Performance	
Precision	Recall
83,95%	91,89%

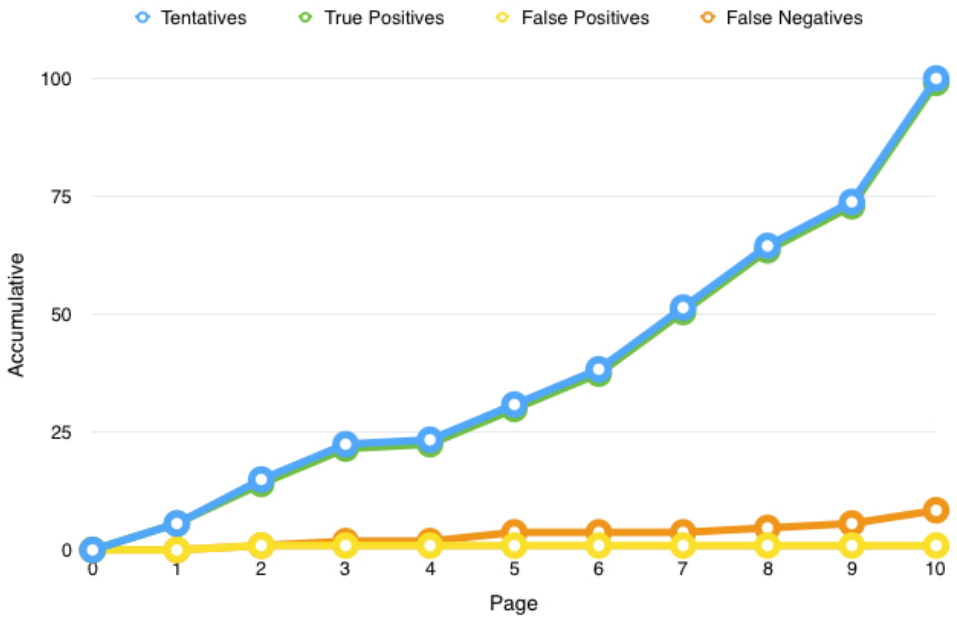
Figure 3.23: Results for Spiderman #4 [WZ16c].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	1	8	2	30	7	12	15	0	0	21
1st Components	16033	8926	9968	72723	4635	20506	37681	45950	54525	52786
2nd Components	66	59	29	232	43	84	151	288	251	109
Tentatives	4	8	2	30	7	12	15	7	1	22
True Positives	1	8	2	29	7	12	15	0	0	20
False Positives	3	0	0	1	0	0	0	7	1	2
False Negatives	0	0	0	1	0	0	0	0	0	1
Time (seconds)	2,3	3,0	2,3	92,0	2,4	5,8	6,1	3,8	6,9	5,0

Overall Performance	
Precision	Recall
87,04%	97,92%

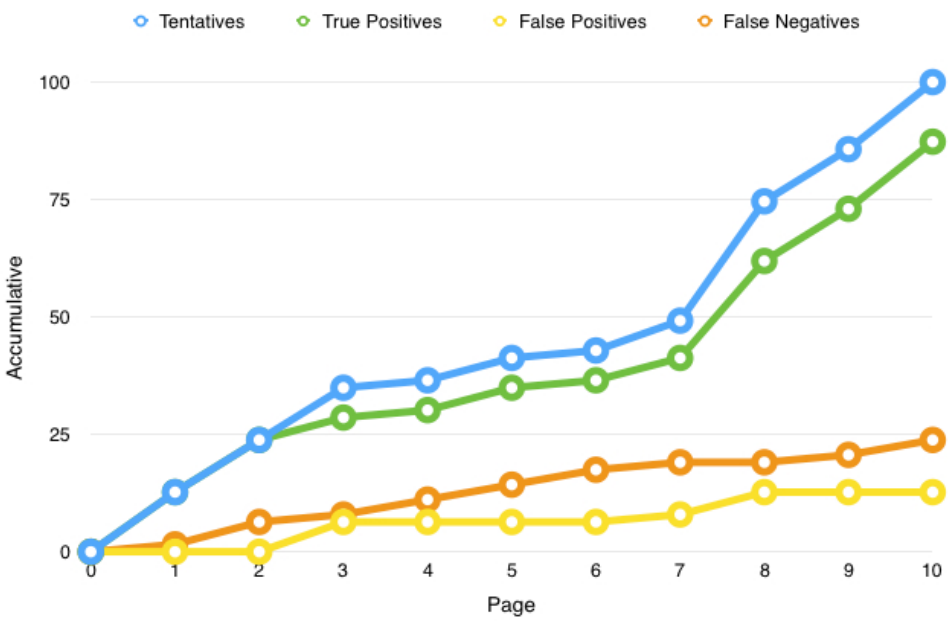
Figure 3.24: Results for Iron Man #1 [Ben15a].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	6	10	9	1	10	8	14	15	11	31
1st Components	31810	49278	28901	12976	23295	26725	11975	21307	17831	56859
2nd Components	113	103	113	46	57	50	37	46	38	110
Tentatives	6	10	8	1	8	8	14	14	10	28
True Positives	6	9	8	1	8	8	14	14	10	28
False Positives	0	1	0	0	0	0	0	0	0	0
False Negatives	0	1	1	0	2	0	0	1	1	3
Time (seconds)	2,5	71,7	5,4	3,8	13,3	2,6	2,7	11,4	2,3	5,3

Overall Performance	
Precision	Recall
99,10%	92,17%

Figure 3.25: Results for Iron Man #2 [Ben15b].



	Performance per Page									
	1	2	3	4	5	6	7	8	9	10
Text Balloons	9	10	4	3	5	3	4	13	8	11
1st Components	16329	52893	38936	25828	21552	13335	23373	26223	26658	15996
2nd Components	44	95	247	122	86	44	72	67	93	69
Tentatives	8	7	7	1	3	1	4	16	7	9
True Positives	8	7	3	1	3	1	3	13	7	9
False Positives	0	0	4	0	0	0	1	3	0	0
False Negatives	1	3	1	2	2	2	1	0	1	2
Time (seconds)	2,5	65,4	3,3	2,6	2,8	2,3	7,0	2,3	2,7	2,4

Overall Performance	
Precision	Recall
87,30%	78,57%

Figure 3.26: Results for Iron Man #3 [Ben15c].

3.6 Discussion

The results (see Fig. 3.15 - 3.26) show that our algorithm detects false positives in a few cases. Interestingly, most of such false positives are, e.g., text balloons misshapen or text balloons with thin connectors broken (see Fig. 3.27). These problems occur due to the thickening behaviour of Sobel, and due to the extraction of connected components in the negative image. Therefore, any text balloon with snippets close to its contour is misshapen, or in the worst case scenario, missed by the algorithm. In addition, any text balloons linked with a thin connector are broken, i.e., the component associated to the connected balloons splits into two or more components.

Overall, and considering the degree of unpredictability associated to comics, the total number of true positives is very high relative to the total number of false positives and false negatives. In addition, the precision and the recall also are very high for every comic book.

Figure 3.27: Examples of false positives produced by our algorithm.

A few opened balloons were also found during the processing of the dataset of book pages, but they were missed by the algorithm. In fact, our algorithm handles text balloons as long as it is possible to assume that they are delimited by closed contours. In addition, text snippets must have a certain distance from the contour of the balloon.

As a final remark, it is important to mention the fact that other authors have addressed the feasibility of applying digital image processing techniques to comic book pages, in order to extract balloons. However, and in a quality measurement standard, our algorithm surpasses most of them by not relying on some assumptions that are not always true, namely: text balloons connected to panels [GKSH06], text balloons own bright background [AT11, hBO12, RS13, CG15], text balloon background is similar to the one of the page [RTBO13b], or text balloon background is identical for all balloons inside the same page [RKdW+13].

3.7 Concluding remarks

This chapter has described the proposed algorithm to extract text balloons from comic book pages. As shown by the results, the number of correcly located and identified baloons as a whole is high. However, it is important to mention that an effort still has to be done to improve its limitations. Therefore, in the next chapter we will conclude the dissertation, though a few open issues have been outlined for possible future work.

Chapter 4

Conclusions

This dissertation is about extracting text balloons from comic books, which is an important research topic in digital comics. In our opinion, the main contribution of this work remains in an automatic balloon extraction algorithm using digital image processing techniques. Consequently, we end up extending the extraction to a level that goes beyond the traditional feature of white background and black text, and without making any assumptions regarding color depth of the image, orientation and language of the text.

4.1 Research context

Digital comics and extraction of text balloons using image processing techniques has been a research topic in MediaLab (Computer Graphics and Multimedia Laboratory), Department of Computer Science and Engineering, Universidade da Beira Interior, Portugal, for the last few years. In this context, the work carried out in this dissertation started after getting a better understanding of how much challenging can be to extract text balloons from comic books. In fact, current algorithms are seemingly adequate for simple comics, but not so well for complex ones [CG15].

Recall that, the challenge comes from the fact that there is no general extraction algorithm in the literature capable of handling any text balloons without making any assumption regarding color depth of the image, orientation or language of the text. Even worse, it is the fact that the comics art evolves over time, so that there is some degree of unpredictability associated to comics. This means that, an algorithm may work well for comic books released twenty years ago, but not so well for current comic books, even considering they belong to the same category or series.

4.2 Research questions

Taking into consideration the research work that has led to the writing of the present dissertation, we are in a position to respond to the following research questions:

Is it feasible to use corner detection to extract text balloons?

In Chapter 3, we show that the answer to this question is positive. It is clear that detecting the presence of text can also be a real challenge. Essentially, we are using

the principle of the existence of corners in almost every text letter. So, after discarding a significant number of regions that are not considered as tentative text balloons for one reason or another, we look at the shape of the holes of the remaining regions to check if they still hold a significant number of corners capable to make a candidate be classified as text balloon.

Summing up, and recalling the problem statement mentioned in Chapter 1:

Is it possible to extract text balloons from comic book pages without making any assumptions regarding color depth of the image, orientation and language of the text?

We are able to say that the research work described in this dissertation responds positively to such problem statement. Furthermore, corners revelead themselves adequate to extract text balloons without making any assumptions regarding color depth of the image, orientation and language of the text. These are important advantages of the proposed algorithm, and at this precise moment, recall that such particularities are not offered in any other balloon extraction algorithm.

Interestingly, the corner detection can be also used to directly extract text snippets in comics, i.e., the extraction of text snippets can be accomplished without considering that they are enclosed by balloons.

4.3 Algorithm limitations

During the study and design of the algorithm, a few limitations were identified, which end up opening a window for future work, namely:

- *Sensitivity to links between balloons*. The algorithm extracts components by identifying sets of connected non-zero pixels in the negative of a Sobelized image. In fact, when a balloon appears linked by a very thin connector, the edge thickening behaviour of the Sobel operator may lead to a gap between the region interiors of connected balloons, i.e., the component associated to the connected balloons splits into two components.

- *Sensitivity to continuity of balloon contour*. The algorithm assumes that balloons are regions delimitated by closed contours, but after the analysis of a few comic book pages, we can see that sometimes such contours are not completely closed. Any opened balloon present in a comic book page certainly results in inefficiencies at the postprocessing, since they are not handled, in any manner, by the algorithm.

4.4 Future work

In the near future, our goal is to improve the algorithm in order to overcome the limitations mentioned in the previous section.

In respect to sensitivity to links between balloons, we intend to come up with an algorithm that first will label the boundary of each object, and then it will fill its interior. This will result in an improvement to locate and identify each object. However, it is clear that this will only work for text balloons delimitated by closed contours on each comic book page.

In respect to sensitivity to continuity of balloon contour, we intend to come up with an algorithm capable of handling closed and opened balloons based on active contours, i.e., snakes. Since corners revealed themselves adequate to extract text balloons, the algorithm can also rely in a corner detector to start a snake. Then, we will try to fit a closed contour around the corners in order to make it possible to filter out the regions associated to text balloons, in a similar process to the one proposed by the algorithm described in Chapter 3. It is important to recall that, as far as we know, Rigaud et al. [RTBO13a] have introduced the only algorithm in the context of comics capable to handle closed and opened balloons together, which is also based on snakes. However, the algorithm relies on the assumption that text balloon background is identical for all balloons inside the same page [RKdW+13].

Bibliography

[AT11] Kohei Arai and Herman Tolle. Method for real time text extraction of digital manga comic. *International Journal of Image Processing*, 4(6):669–676, 2011. 12, 45

[Ben15a] Brian M Bendis. *Invincible Iron Man*, volume 1, no. 1. Marvel Comics, 2015. 8, 42

[Ben15b] Brian M Bendis. *Invincible Iron Man*, volume 1, no. 2. Marvel Comics, 2015. 14, 43

[Ben15c] Brian M Bendis. *Invincible Iron Man*, volume 1, no. 3. Marvel Comics, 2015. 44

[Byr84] John Byrne. *Alpha Flight*, volume 1, no. 6. Marvel Comics, 1984. 9

[CG15] João Correia and Abel Gomes. Balloon extraction from complex comic books using edge detection and histogram scoring. *Multimedia Tools and Applications*, 2015. 2, 7, 13, 17, 45, 49

[Dav] Jim Davis. Garfield and friends - daily garfield comic strip. http://garfield.com/comic/. 10, 23

[DC] DC. Dc | welcome to dc. http://www.dccomics.com/. 11

[GKSH06] Qinqlian Guo, Kyoko Kato, Norio Sato, and Yuko Hoshino. An algorithm for extracting text strings from comic strips. In *ACM SIGGRAPH Research Posters (SIGGRAPH'2006)*, page Article no. 76. held in Boston, Massachusetts, USA, July 30 - August 3, ACM Press, 2006. 12, 45

[GW07] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2007. 18

[hBO12] Anh Khoi Ngo ho, Jean-Christophe Burie, and Jean-Marc Ogier. Panel and speech balloon extraction from comic books. In *Proceedings of the 10th IAPR International Workshop on Document Analysis Systems (DAS'2012)*, pages 424–428. held in Gold Coast, Queensland Australia, March 27-29, IEEE Computer Society, 2012. 12, 17, 45

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, pages 147–151, 1988. 21

[Jur15a] Dan Jurgens. *Superman - Lois and Clark*, volume 1, no. 3. DC Comics, 2015. 36

[Jur15b] Dan Jurgens. *Superman - Lois and Clark*, volume 1, no. 4. DC Comics, 2015. 37

[Jur15c] Dan Jurgens. *Superman - Lois and Clark*, volume 1, no. 5. DC Comics, 2015. 38

[MA09] Raman Maini and Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *nternational Journal of Image Processing (IJIP)*, 3, 2009. 20

[Mac93] Scott MacCloud. *Understanding Comics*. Kitchen Sink Press, 1993. 10

[Mar] Marvel. Marvel - the official site. http://marvel.com/. 11

[Mar07] Christy Marx. *Writting for Animation, Comics and Games*. Focal Press, 2007. 10

[MD07] Grant Morrison and Tony Daniel. *Batman*, volume 1, no. 670. DC Comics, 2007. 19, 33

[MD08a] Grant Morrison and Tony Daniel. *Batman*, volume 1, no. 671. DC Comics, 2008. 34

[MD08b] Grant Morrison and Tony Daniel. *Batman*, volume 1, no. 672. DC Comics, 2008. 35

[OD08] David L. Olson and Dursun Delen. *Advanced Data Mining Techniques*. Springer, 2008. 32

[Pie] Nate Piekos. Comic book grammar and tradition by nate piekos. http://www.blambot.com/articles_grammar.shtml. 10, 11

[Rig14] Christophe Rigaud. Segmentation and indexation of complex objects in comic book images, 2014. 10

[RKdW+13] Christophe Rigaud, Dimosthenis Karatzas, Joost Van de Weijer, Jean-Christophe Burie, and Jean-Marc Ogier. Automatic text localisation in scanned comic books, 2013. 13, 45, 51

[RS13] S. Ranjini and Dr. M. Sundaresan. Extraction and recognition of text from digital english comic image using median filter, 2013. 12, 45

[RTBO13a] Christophe Rigaud, Norbert Tsopze, Jean-Christophe Burie, and Jean-Marc Ogier. An active contour model for speech balloon detection in comics, 2013. 13, 51

[RTBO13b] Christophe Rigaud, Norbert Tsopze, Jean-Christophe Burie, and Jean-Marc Ogier. Robust frame and text extraction from comic books, 2013. 12, 17, 45

[SA85] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics and Image Processing*, pages 32–46, 1985. 28, 29

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994. 17, 21

[WOS09] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Optimizing two-pass connected-component labeling algorithms. In *Pattern Analysis and Applications*, volume 12, pages 117–135. Springer, 2009. 23

[WZ16a] Danielle Wolf and Jim Zubkavich. *Ultimate Spider-Man Spider-Verse*, volume 1, no. 2. Marvel Comics, 2016. 39

[WZ16b] Danielle Wolf and Jim Zubkavich. *Ultimate Spider-Man Spider-Verse*, volume 1, no. 3. Marvel Comics, 2016. 40

[WZ16c] Danielle Wolf and Jim Zubkavich. *Ultimate Spider-Man Spider-Verse*, volume 1, no. 4. Marvel Comics, 2016. 41