



UNIVERSITÀ
DEGLI STUDI
FIRENZE

UNIVERSITÀ DEGLI STUDI DI FIRENZE
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)
CORSO DI DOTTORATO IN INGEGNERIA DELL'INFORMAZIONE
CURRICULUM: INGEGNERIA INFORMATICA

DEEP LEARNING-BASED OBJECT
DETECTION MODELS APPLIED TO
DOCUMENT IMAGES

Candidate
Zahra Ziran

Supervisor
Prof. Simone Marinai

PhD Coordinator
Prof. Fabio Schoen

CICLO XXXII, 2016-2019

Università degli Studi di Firenze, Dipartimento di Ingegneria
dell'Informazione (DINFO).

Thesis submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Information Engineering. Copyright © 2020 by
Zahra Ziran.

Acknowledgments

First of all, I would like to dedicate this thesis to my mother that I have no chance of her presence in this moment of my life, who, in all the difficult moments of his absence, just remembering her love, hard work and hope for my success made me stay and keep going. Second, I would like to dedicate this thesis to my father, who is a symbol of patience, fortitude, and unsparing affection for me in my life. I thank him from the bottom of my heart for never letting me down, for all he has down, which I will never forget. Third, I would like to dedicate this thesis to my brothers, Babak and Behnam, who, in addition to always accompanying me, helped me with their knowledge in preparing the valuable ISTA data set.

Moreover, I would like to dedicate my precious gratitude to my supervisor, Prof. Simone Marinai, who will be my role model for the rest of my life. I thank him for the moments when he was the only energizing entity, whose unique personality made me continue this path and overcome the hardships of this path both academically with his exemplary experience and knowledge and spiritually with his unsparing support. However, I can not include what he deserves in this paragraph and in this piece of paper to be able to do justice. So, I summarize in one sentence that I owe him for what I learned from him on this path for the rest of my life.

Also, I deeply would like to thank all my colleagues of the AI lab who were of great help during my research and shared the best moments and memories with me, Dr.Stefano Martina and Federica, whose friendship to me is like a valuable treasure I found in this geography, Dr.Samuele Capobianco, with whom we always had fun and good times because of his sense of humor, Dr.Tibero Uricchio, whose companionship made me always feel that there is a friend that I can share my problems with, and I also grateful to my other dear friends, Dr.Alessandro Lazzeri, Amin Zadenoori and Francesco Lombardi, for their energizing presence.

Further, I would like to express my special thanks to my dear friend, Dr.Ala Arman, whose experience, knowledge, patience and constant association with me on this path made me able to count on his affection, and his unsparing helps made it easier for me to follow this path.

Finally, I would like to appreciate Prof. Fabio Schoen, who did not spare any help and effort to improve this path, as much as possible.

Contents

Contents	v
1 Introduction	1
1.1 Contributions	1
1.2 Outline	2
2 Literature review	5
2.1 Document Image Analysis	5
2.1.1 Application of DIA	7
2.1.2 Object Detection in Floor Plan Images	12
2.2 Deep Learning for Object Detection	13
2.2.1 Region Proposal-Based Framework	14
2.2.2 Regression-based object detectors	22
2.3 Text Detection in scene images	25
2.4 Summary of conclusions	29
3 Object Detection in Floor Plan Images	31
3.1 Introduction	31
3.2 Previous Work	32
3.3 The Architecture	33
3.3.1 False Negative Calculation	35
3.4 Floor Plan Datasets	36
3.4.1 Experimental Results	38
3.4.2 Discussion	42
3.5 Conclusion	45

4	Text Recognition and Alignment	47
4.1	Introduction	47
4.2	Text alignment in early printed books	49
4.2.1	Text Segmentation	50
4.2.2	Dynamic programming	53
4.2.3	Dataset	56
4.3	Text Recognition in Floor plan Images	60
4.3.1	Text detection module	61
4.3.2	Text analysis and classification module	69
4.3.3	Data sets	70
4.3.4	Experimental results	70
4.4	Conclusion	72
5	Floor Plan data sets	73
5.1	Introduction	74
5.2	Floor Plan data sets	77
5.3	Data sets Creation	81
5.3.1	ISTA	81
5.3.2	FLo2Plan	83
5.4	Annotatation tools	83
5.4.1	Labeling	83
5.4.2	Labelme	86
5.5	Conclusion	88
6	Conclusion	89
6.1	Summary of contribution	89
6.2	Directions for future work	90
A	Publications	93
	Bibliography	95

Chapter 1

Introduction

In Document Image Analysis (DIA), which deals with solutions to obtain computer-readable description from document images, understanding and recognition of a wide spectrum of complex document images from business and financial documents to floor plans pose a key challenge due to high-level semantic information carried in such documents. The primary task is then to isolate different present contents in the documents (e.g., graphical and textual components).

In this thesis, the main objective is the recognition and understanding of graphical documents in order to generate accessible graphical documents using Deep learning-based object detection models. To do so, first, the object detection in floor plans is addressed by creating and extending floor plan data sets, and then, proposing reliable detection approaches to suitably operate in real scenarios. Second, the role of transcript alignment in early printed loosely annotated texts to support word detection inside unknown images is investigated.

1.1 Contributions

In general, this thesis follows three main lines of research. It is first focused on the understanding of floor plan images, using deep learning-based object detection methods for text and object recognition. Next, the lack of large data sets of floor plan images, that could be used to investigate object detection techniques in floor plan data sets, is addressed by creating a comprehensive data set. Then, text alignment in document images is considered

by focusing on both deep learning and dynamic programming.

The first contribution focuses on the use of deep neural networks for object detection in floor plan images. In particular, object detection architectures, which originally designed and trained to recognize objects in scene images, are used for recognizing furniture objects (e.g., doors, table, chair, bed) in floor plans. Deep neural networks based on Faster R-CNN is performed on this task and also to text detection in floor plan images.

The second contribution of this thesis is the creation of two data sets that have been used for performing the experiments covering different types of floor plans with different peculiarities. The first data set, called ISTA, comes from an architectural firm. The second one, called Flo2plan, which is at least 3 times larger than ISTA, includes different kinds of floor plans made by different companies. Although a lot of data sets in floor plan images have been created, there is still a lack of a comprehensive and reasonable large data set for implementing deep learning methods to create an accessible graphical document for visually impaired people.

The third contribution focuses on proposing a technique for transcript alignment in early printed books by using deep models in combination with dynamic programming algorithms. Two object detection models, based on Faster R-CNN, are trained to locate words. First, an initial model is trained to recognize generic words and hyphens by using information about the number of words in text lines. Using the model prediction on pages with a line-by-line ground-truth annotation is available, a second model is trained which is able to detect landmark words. The alignment is then based on the identification of landmark words in pages where only the text corresponding to zones in the page is known. The proposed technique is evaluated on a publicly available digitization of the Gutenberg Bible while the transcription is based on the Vulgata, a late 4-th century Latin translation of the Bible.

1.2 Outline

This thesis is organized in five chapters as follows:

- **Chapter 2** presents the state of the art of different approaches proposed for addressing the problems associated with document image analysis and also presents deep learning-based object detection methods in this thesis.

- **Chapter 3** investigate the use of deep networks for object detection in floor plan images. The usage and the adaptation of the Tensorflow Object Detection API are explored to identify floor plan objects in two data sets that have been built to address this task.
- **Chapter 4** first proposes a model for transcript alignment in early printed books by using deep models together with dynamic programming algorithms. Second, a textual contents processing model is proposed in floor plan images.
- **Chapter 5** introduces two novel floor plan image data sets, ISTA and Flo2plan. The labeling approach, adopted to annotate these data sets, is also highlighted.
- **Chapter 6** concludes this dissertation by highlighting the suitability of the presented deep learning-based object detection models for document image understanding together with a brief discussion about future research directions.

Chapter 2

Literature review

This chapter gives a brief survey of related work on Document Image Analysis using deep neural networks. The first part of the chapter briefly introduces the problem of object detection in document images and some useful applications in document image analysis, while the second part deals with the different architectures of convolutional neural networks for object detection. Finally, text Detection in scene images is presented as another task that has been used in this research.

2.1 Document Image Analysis

One general framework for document image analysis is described in this section. Figure 2.1 shows the sequence of steps in document image analysis. The steps include capturing data from a computer, binarization, feature level analysis, text, and graphical components' analysis and recognition.

Data capturing: the data in a paper document are usually captured using a scanner and stored in a file in the form of an image with different extensions (e.g., jpeg, tiff, bmp, png). The captured document may be a colorful or grayscale image.

Binarization and preprocessing: To separate foreground and background information, the document image is binarized. This step converts the captured document into pixels with intensity level 1 or 0. Preprocessing includes noise reduction, segmentation and converting the image into the required form for further processing. Noise in the document image is pos-

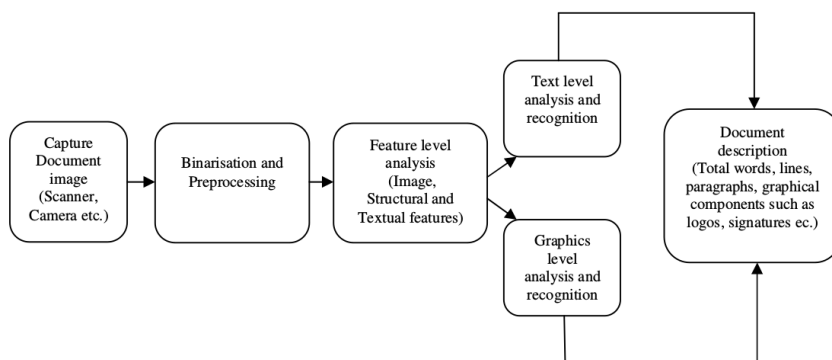


Figure 2.1: Steps involved in document analysis.

sible from many sources including degradation due to aging, photocopying, or during data capture. Segmentation is carried out to separate textual and graphical components of the document. Segmentation of textual information helps in locating columns, paragraphs, words, characters, and segmentation of graphics which aim at separating symbols, logo, signature, and lines from the document image.

Feature level analysis: This step is to carry out an analysis of textual and graphical components. Image features, structural features or textual features are extracted from the document image. These features may be either local or global. Figure 2.2 summarizes the different features used for document image analysis.

Text level analysis and recognition: Two main types of analysis are applied to text in documents. The first one is Optical Character Recognition (OCR) to extract the meaning of the characters and words from document images. The second method is page-layout analysis to recognize the formatting of the text in a document image that includes text bodies in different functional blocks (e.g., headers, footers, titles, subtitles).

Graphics level analysis and recognition: Graphics level analysis includes recognizing lines, curves, and other features to identify different components such as graphic symbols, logos, and layout of the document for further inspection.

Document description: The result of document image analysis is document description and consists of both textual and graphical components

present in the document.

Image features	Structural features	Textual features
1. Connected components.	1.Physical Layout.	1. Page/Text layout features.
2. Gaps between row/Columns.	2.Logical structures	2. Textual features from OCR results analysis
3. Location and size of cells.	3.Results of functional labeling	
4. Text histogram.	4.Spatial relations	

Figure 2.2: Different features used in document image analysis

2.1.1 Application of DIA

In this section, different aspects of real problems of document image analysis, as well as examples of successful applications are covered.

Table detection is a crucial step in many document analysis applications as tables are used for presenting essential information to the reader in a structured manner. This is a hard problem due to varying layouts and encodings of the tables. Researchers have proposed numerous techniques for table detection based on layout analysis of documents.

Pacha et al [75] performed a study called “Handwritten Music Object Detection: Open Issues and Baseline Results”. In their study, they stated that the main aim of Optical Music Recognition (OMR) is to automatically understand written music scores. By recognizing the visual structure and the objects within a music sheet, this method tries to understand the musical content of documents containing printed or handwritten music scores. After recognition of all the objects, there will be a semantic reconstruction step to identify the objects’ relations with each other and recover the musical semantics. In light of recent developments in computer vision, which have been promoted through the popularity of deep convolution neural networks, OMR received several groundbreaking contributions that result in inaccurate results for particular sub-problems including staff line removal or symbol classification. With the aim of accurate music objects detection in music scores, Pacha et al. studied the problems with music object detection. It is generally agreed that there are two possible forms for music objects

including primitive glyphs or compound symbols implemented in music notation. The job of a music object detector is to take an image and yield the bounding-box and class-label for every found object. Pacha et al. used a machine learning approach and an end-to-end trainable object detector on the MUSCIMA++ data set to state the way of creating an accurate and generalizable music object detector. The performance of different object detectors and feature extractors were studied through several experiments to evaluate their proposed approach. Other fields such as staff line removal effects and the impact of removing rare symbols are also investigated. They used the deep learning library TensorFlow to adopt a work on detecting music objects through. It is possible to find the entire source code such as detailed instructions and training protocols in Music Object Detector-TF which consists of 4 steps. In the first step three meta-architectures including -CNN, R-FCN, and SSD as object detectors have been investigated. The first two architectures including R-CNN and R-FCN are two-stage detectors and contain a region proposal network as well as a region classifier. Although a sliding window for classification is used in Faster R-CNN, position-sensitive score maps and per RoI pooling is used by R-FCN, which is more beneficial. However, there is fairly reduced precision. and in the second step, as feature extractors, Inception ResNet-v2, ResNet50, MobileNet-v1, and Inception-v2 exclude custom made networks that cannot take advantage of transfer-learning. In the third step, several images with and without staff lines are studied of all the 105 classes existing in the MUSCIMA++ data set, they reduced to 71 classes. 34 classes that appeared less than 50 times in ground truth are removed.

Pacha et al. in [74] performed a study to build a universal music symbol classifier that can classify music symbols no matter how they are well printed or just handwritten. They proposed a data-driven approach to build this kind of classifier. To this end, they made tools which can unify multiple data sets into a single large data set on which it is possible to train universal music symbol classifier. They unified seven data sets into a collection containing over 9000 samples which belonged to 79 classes. They stated that one significant feature of a universal music classifier is the ability to detect all types of music symbols irrespective of being printed or handwritten. A powerful and appropriate way for solving computer vision tasks is offered by deep natural networks including convolutional neural networks by some researchers in [53] and they built these classifiers through training a convolutional neural network on the aforementioned data set.

Nguyen et al. [72] stated that several methods have been proposed for extraction/ detection of characters from comics in the past few years which indicated reasonable performance. To train the comic characters detector, they have implemented the latest object detection deep networks. The properties and performance of the deep learning approach for detecting comic characters are analyzed in this study through their proposed data set called “Sequency612” as well as three data sets from the previous studies including Sun60 data set[95], Fahad18 data set[50], and Ho42 data set[43]. Since the standard approach for object detection uses CNN classifiers on various sub-windows or regions derived from the image, they preferred to use deep learning methods to do object detection. When the classification was completed, CNNs adjusted the object localization by eliminating the errors. Two modern neural networks are trained including RCNN and YOLOv2 via their Sequency612 dataset. In their study, it was concluded that YOLOv2 had better performance in terms of accuracy. They expanded YOLOv2 model according to the scratch, refined Darknet’s model, and kept Darknet’s model due to its better performance.

One of the most challenging steps in many document analysis applications is table detection because tables, with different layouts and encodings, use a structured way to present crucial information. Based on the layout analysis of documents, several techniques are suggested for table detection. A deep learning-based approach is proposed by Hao et al. in [37] for table detection. Region proposals from document images are computed by this system through some predefined set of rules. Then, these region proposals are passed to the CNN, which detects if a certain region proposal belongs to the table region or not. However, one of the main limitations is that although it is suitable for tables that have ruling lines, when the table is spanned across multiple columns, it is not able to localize table regions.

Gilani et al. in [32] proposed a deep learning-based method for table detection. This method includes two main modules including image transformation and table detection. Documents are blank spaces and content regions. For the separation of these regions, image transformation is used. However, faster R-CNN is preferred in the table detection module as a basic element of the deep network. Faster R-CNN depends on the combined network which contains Fast R-CNN and Region Proposal Networks. Their approach has been shown in [32].

Gatos et al. in [31] presented an automatic table detection technique in

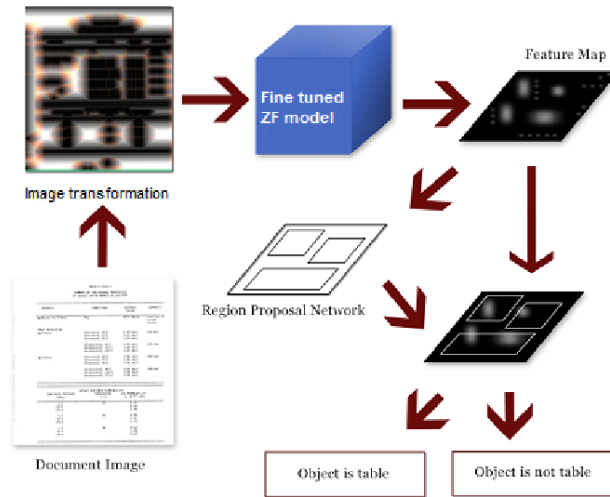


Figure 2.3: The document image is first transformed and then fed into a fine-tuned CNN model. It outputs a feature map which are fed into region proposal network for proposing candidate table regions. These regions are finally given as input to fully connected detection network along with the convolutional feature map to classify them into tables or nontables. [32]

document images. It is generally agreed that lines and tables are the very common graphic and non-textual entities in documents. Therefore, their detection is of importance in evaluating OCR performance and describing document layout. A workflow for table detection is proposed that includes three steps including image pre-processing, horizontal and vertical line detection and table detection. A performance evaluation scheme is used to determine the proposed method efficiency which includes newspapers, scientific journals, certificates and so on. Several steps are included in the process of table detection. First of all, the pixels of the detected lines are removed. Then, all the detected lines must be grouped horizontally and vertically. After that, each group is aligned based on the mean value of the horizontal or vertical positions. Finally, a reconstruction table is achieved via the drawing of the horizontal and vertical lines which connect all pairs of line intersection. Goa et al. in [30] stated that it is important to build a detection method for PDF documents. In Figure 2.4 an example of formula detection has been shown.

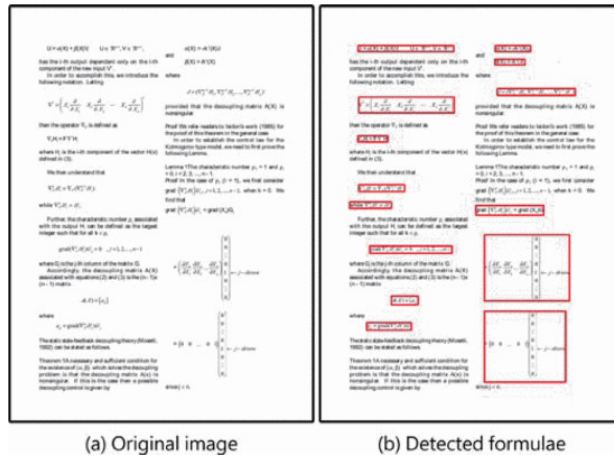


Figure 2.4: An example of formula detection. (a) Document page (b) detected results, formulae in it is detected by our method and annotated with red boxes. [30]

In their study, a CNN and a recurrent neural network (RNN) are combined and refined to detect formula based on their vision and character properties. They proposed a series of strategies for training and optimizing deep networks like implicit class down-sampling which can reduce unbalance dens between formula and the other page elements such as tables and figures. Also, they redesign the region proposal method to make moderate formula candidates via a combination of bottom-up and layout analysis. The results obtained in this study indicate that by combining CNN and RNN, it is possible to increase the validity of their proposed detection method. According to the framework of this study, a PDF file is used as the input, and two sequential parts are used to detect the formula appearing in the papers including candidate formula region generation and formula identification. In the first part, the page rendering information is parsed and some steps are used to check and correct the information. After that, top-down and bottom methods for layout analysis are used for generating candidate formula regions. In this step, the text, image information stream of each candidate region, and graphs, as the output, which is the input of the following process. For extraction of the candidate features, feature extraction networks are trained in the formula identification part. There are several post-processing rules

for adjusting and refining incomplete formula areas. Then classification and post-processing are implemented until no formula region is modified as it is shown in Figure 2.5.

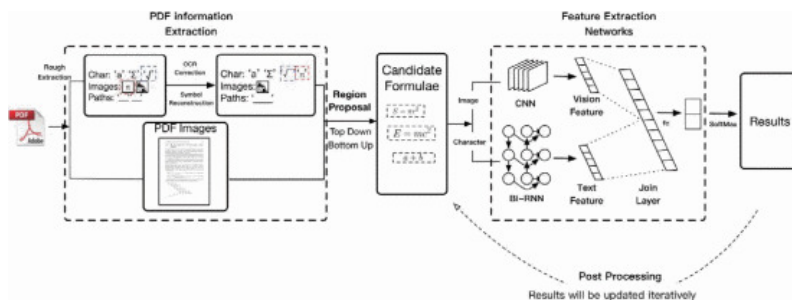


Figure 2.5: Method framework [30]

2.1.2 Object Detection in Floor Plan Images

In the past, floor plan analysis is used for different purposes. Heras et al. [15] presented CVC-FP data set. This database has been generated using the SGT-tool. They also proposed two different tasks of wall segmentation and room detection. Heras et al.[10] proposed a perceptual model to describe the drawing style in architectural plans without interpreting the building elements (e.g. symbols of the drawing and in a detection frame). This descriptor allows the search for perceptually similar plans into a database. Heras et al. [16] proposed a wall segmentation approach in floor plans with three functions of working independently to the graphical notation, needing no pre-annotated data for learning, and segmenting multiple-shaped walls such as beams and curved-walls. Macé et al. [63] presented a two-step method for the interpretation of architectural floor plans, more specifically for recognizing the rooms it contains. The method contains the extraction of the primitives in the image (the lines and the arcs that constitute respectively the walls and the doors) and the detection of the rooms that form the building. Sharma et al.[89] presented an automatic lookup tool for matching and retrieving similar floorplans from a large repository of digitized architectural floor plans. First, they perform a graph-based approach incorporating semantics in terms of the room layout and arrangement of furniture while analyzing the floor plans to distinguish between layouts with more specificity.

Second, they adopt a graph spectral embedding approach to represent the graphs obtained for room layouts as a three-component vector and finally, they used an algorithm to calculate the semantic difference between layouts. Ahmed et al. [4] (Automatic Room Detection and Room Labeling from Architectural Floor Plans) presented a complete system, containing a corpus of just 80-floor plans, for automatic room recognition and room labeling from architectural floor plans. In order to retrieve the room information, the system applies structural and semantic analysis steps. Additionally, the system extracts the room labels to identify the functions of the rooms. Samuel et al. [20] proposed a method for analyzing floor plan images using wall segmentation, object detection, and optical character recognition. They introduced a real-estate floor plan data set, R-FP, evaluate different wall segmentation methods, and propose fully convolutional networks (FCN). In this research a subset of the R-FP images was annotated for training and testing, with 6 different object classes (doors, sliding doors, kitchen stoves, bath tubs, sinks, and toilets). The average precision evaluated on 25 test images is 96.0% for doors, 35.9% for sliding doors, 76.2% for kitchen ovens, 95.8% for bath tubs, 69.2% for sinks, and 70.8% for toilets, for an IoU value of 50% . According to the state of the art in object detection in the floor plan, we can see still there are a lot of space in this area to be considered.

2.2 Deep Learning for Object Detection

This section is related to explain two types of object detection frameworks and various algorithms like Faster R-CNN, YOLO, SSD. Object detection is defined as the prediction of the location and class of an object. Instead of object class prediction, however, it is required to predict both class and rectangle (bounding box) of an object nowadays. To do this, there must be four variables to identify rectangles separately. Therefore, the variables which have to be predicted including class name, the top left of the x- and y-coordinates of the bounding box, the width and the height of the bounding box).

For any image, generic object detection is commonly used to classify and locate existing objects. To indicate the confidences of existence, generic object detection can label them by rectangular bounding boxes. As seen in Figure 2.6, the frameworks of generic object detection approaches are categorized into two types. In a group of approaches, generating region proposals

is the first step of the traditional object detection for pipeline and the second step is to classify each proposal into distinct object categories. The second group of approaches considers object detection as a classification or regression issue, using a unified framework to obtain final outcomes (i.e., locations and categories) directly. The region proposal-based approaches mainly consist in region-based fully convolutional network (R-FCN) [8], spatial pyramid pooling (SPP)-net [40], R-CNN [34], Faster R-CNN [84], Fast R-CNN [33], and feature pyramid networks (FPN) [58] as well as Mask R-CNN [39], some of which have been correlated to each other. For example, SPP-net modifies R-CNN by a layer of SPP. The main regression/classification-based approaches are MultiBox [24], Single Shot MultiBox Detector (SSD) [62], G-CNN [70], YOLOv2 [82], YOLO [81], deeply supervised object detectors (DSOD) [91], and deconvolutional single shot detector (DSSD) [29]. The correlations between these two pipelines are bridged by the anchors introduced in Faster R-CNN. Details of these methods are as follows.

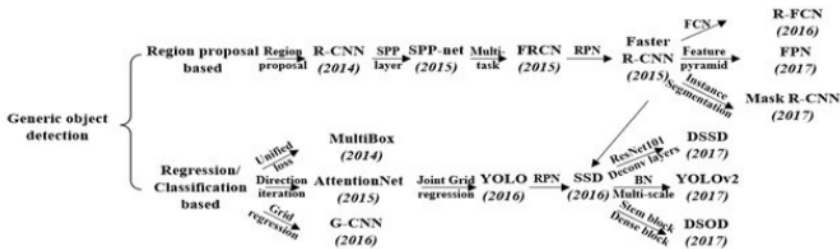


Figure 2.6: Two types of frameworks [108].

2.2.1 Region Proposal-Based Framework

The region proposal-based framework, which has a two-step process, can match the attentional mechanism in the term of the human brain to some extent. In the first step, it provides a coarse scan of the whole scenario and in the second step, it focuses on regions of interest (RoIs). Among [87], [99], and [42], the most representative study is Overfeat [87] because after providing the confidences of underlying object categories, this model considers CNN in the sliding window approach that estimates bounding boxes from locations of the topmost feature map, instantly.

Region-based Convolutional Neural Networks (R-CNN):

The classification accuracy affects success because object detection is modeled based on a classification problem. Although achieving deep learning had led to the logical idea of using a more accurate convolutional neural network based on a classifier instead of HOG classifiers, as proposed by Delal et al. in [9], there was still one problem to be solved. Since CNNs were too costly and slow, CNNs running on the high number of patches made, considering a sliding window detector, was not feasible. This problem is resolved by R-CNN through “Selective Search” which is an object proposal algorithm. Hence, the bounding boxes fed to the classifier are reduced to approximately 2000 region proposals. To provide all the possible locations for an object, this algorithm utilizes some cues including intensity, color, incidence measure, texture and so on. The R-CNN flow chart is shown in Figure 2.7. As observed, it is divided into three steps as:

- Implementing Selective Search to make probable objects. The R-CNN adopts selective search [102] to generate about 2000 region proposals for each image. The selective search method relies on simple bottom-up (BU) grouping and saliency cues to provide more accurate candidate boxes of arbitrary sizes quickly and to reduce the searching space in object detection [26], [19].
- Feeding these patches to CNN, followed by SVM for predicting each patch class. In this stage, each region proposal is warped or cropped into a fixed resolution, and the CNN module in [51] is utilized to extract a 4096-dimensional feature as the final representation. Due to large learning capacity, dominant expressive power, and hierarchical structure of CNNs, a high-level, semantic, and robust feature representation for each region proposal can be obtained.
- Optimizing patches through training bounding box regression separately. When there are scarce or insufficient labeled data, pretraining is usually conducted. Instead of unsupervised pretraining [88], R-CNN first conducts supervised pretraining on ImageNet Large-Scale Visual Recognition Competition, a very large auxiliary data set, and then takes a domain-specific fine-tuning. This scheme has been adopted by most of the subsequent approaches [33], [84].

Also, in addition to its improvements compared to common approaches and significance in bringing CNN into practical object detection, there exist some

disadvantages as follows: 1) The CNN needs a constant size (for example, 227×227) input image that could lead to the recomputation of the total CNN for obtaining analyzed region, directly, selecting a great deal of time for the testing period because of the existence of FC layers. 2) R-CNN training is known as a multi-stage pipeline. Firstly, a usual network (ConvNet) on object suggestions is fine-tuned and after that, to fit in with ConvNet features, the softmax classifier learned by fine-tuning was changed by SVMs. Lastly, bounding-box regressors were trained. 3) Training is known as expensive in time and space. Finally, the achieved region proposals are currently redundant, however, the selective search produces region proposals by approximately high recalls. This approach is time-consuming. Moreover, for solving the inaccurate localization issue, there exists some enhancement. For guiding the regressions of different BBs, sequentially, in [107] a Bayesian optimization-based search approach was utilized. Also, researchers trained class-specific CNN classifiers along with a structured loss for penalizing the localization mistake explicitly.

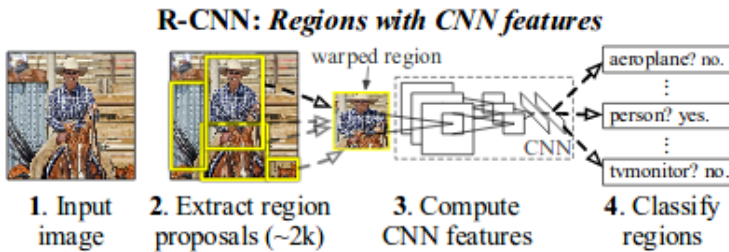


Figure 2.7: Flowchart of R-CNN [34], which consists of three stages: 1) extracts BU region proposals, 2) computes features for each proposal using a CNN, and then 3) classifies each region with class-specific linear SVMs.

Spatial Pyramid Pooling (SPP-net): Since it takes much time to run CNN on 2000 region proposals made using the selective search algorithm, RCNN has been too slow and there are still some disadvantages.

Fully connected layers of the CNN it is necessary to make inputs with the fixed size. That is why R-CNN chooses to warp or crop each region proposal into the same size. However, the object may exist partly in the cropped region and unwanted geometric distortion may be produced due to the warping operation. These content losses or distortions will reduce

recognition accuracy, especially when the scales of objects vary. To solve this problem, He et al. [40] took the theory of spatial pyramid matching (SPM) [52], [76] into consideration and proposed a novel CNN architecture named SPP-net. SPP-net allows the calculation of the CNN representation for the whole image just one time and it is possible to utilize that for the CNN representation calculation for patches produced through Selective Search. To do this, it is required to perform a pooling type of operation only on that part of the last convolution layer feature maps which are correspondent to the region. It is possible to calculate the rectangular part of the convolution layer corresponding to a region through the region projection on the convolution layer by considering the downsampling in the intermediate layers. In contrast to traditional max-pooling, SPP utilizes spatial pooling after the final convolutional layer. A region with any arbitrary size is divided into a fixed number of bins by SPP layer, then the max pool is implemented on all the bins and a vector with the constant size can be generated since the bins number is kept the same. SPP net had one serious disadvantage. In fact, performing backpropagation by the spatial pooling layer could not be ignored. Thus, the fully connected part of the network was only fine-tuned. SPP-Net contributed to the development of more popular and very fast RCNN which will be described in the following sections.

Fast R-CNN: Although SPP-net was obtained effective enhancement in the cases of efficiency and accuracy over R-CNN, it currently has numerous considerable drawbacks. Besides, SPP-net commonly selects the same multistage pipeline (R-CNN) like network fine-tuning, feature extraction, bounding box regressor fitting, and SVM training. Hence, an additional expense is still needed on storage space. As stated in [64], the layers of convolutional preceding the SPP layer cannot be updated by the fine-tuning algorithm proposed. It is then determined that the accuracy reduction of very deep networks is expected. To this end, Girshick [33] introduced a multitask loss on classification and bounding box regression and proposed a novel CNN architecture named Fast R-CNN. The architecture of Fast R-CNN is exhibited in Figure 2.8. R-CNN uses a simple back-propagation calculation to propagate the gradients via spatial pooling. It is not very different from max-pooling gradient calculation except for the fact that there is an overlapping among the pooling region and thus it is possible for a cell to have gradients pumping in from multiple regions. Another innovation represented by Fast R-CNN is that the bounding box regression was added to the neural

network training itself. Therefore, the network came up with two heads including bounding box regression head as well as classification head. One of the most outstanding features of Fast R-CNN is the same multitask objective since it does not require independent network training for localization and classification. Consequently, the overall time of training is reduced and the accuracy is increased compared to SPP-net due to the end to end learning of CNN.

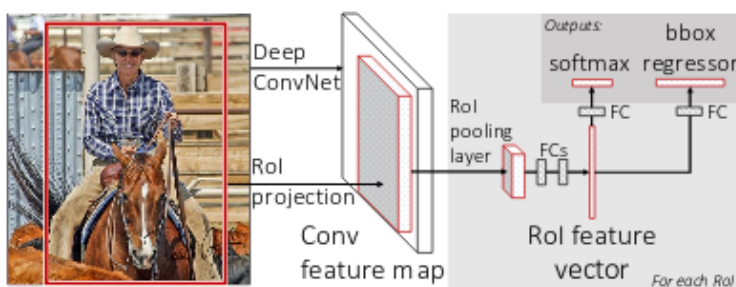


Figure 2.8: Architecture of Fast R-CNN [33]

Faster R-CNN: Many scholars attempted to produce candidate boxes with biased sampling as in [55]. However, to propose a candidate pool of isolated region proposals, state-of-the-art object detection networks are commonly based on additional approaches like Edgebox and selective search. Region proposal computation can act as a bottleneck in enhancing efficiency. In [84] and [83], authors have suggested an additional region proposal network (RPN). It is obtained by an FCN that can estimate object scores along with bounds at each position, simultaneously. RPN can act in a nearly cost-free method by sharing full-image convolutional features with detection network. Similar to [102], to make a set of rectangular object proposals, RPN takes a picture of arbitrary size. It also acts on a particular layer of convolutional with the preceding layers assigned to the object detection network.

As shown in Figure 2.9, the whole system contains a single and unified network for detecting objects. This process is modeled by a fully convolutional network since it aims to share computation result with a Fast R-CNN object detection network. It is possible to predict multiple region proposals at each sliding-window location simultaneously. The maximum number of possible proposals for each location is indicated as “k”. Therefore, the reg

layer contains $4k$ outputs that encode k boxes coordinates, and the cls layer has $2k$ outputs that approximate object or not object probability for each proposal. The parametrization of k proposals is according to the k reference boxes which are called anchors. The architecture of RPN is shown in Figure 2.10. Considering anchor boxes is an idea proposed by Faster R-CNN to manage the variations in the object scale and aspect ratio. Three types of anchor boxes are used by the original paper at each location for scale 128×128 , 256×256 and 512×512 . Likewise, three aspect ratios are used for aspect ratio as follows: 1:1, 2:1 and 1:2. Therefore, there will be 9 boxes at each location on which being background or foreground probability is predicted by PRN. Bounding box regression is applied with the aim of anchor boxes improvement at each location. Thus, various sizes of bounding boxes are given out with the similar probabilities of each class. Applying Spatial pooling like Fast R-CNN, allows bounding boxes with various sizes to be passed further. Faster-RCNN is approximately 10 times faster compared to Fast R-CNN with the same data set accuracy similar to VOC-2007. That is why Faster R-CNN is considered as the most accurate algorithm for object detection.

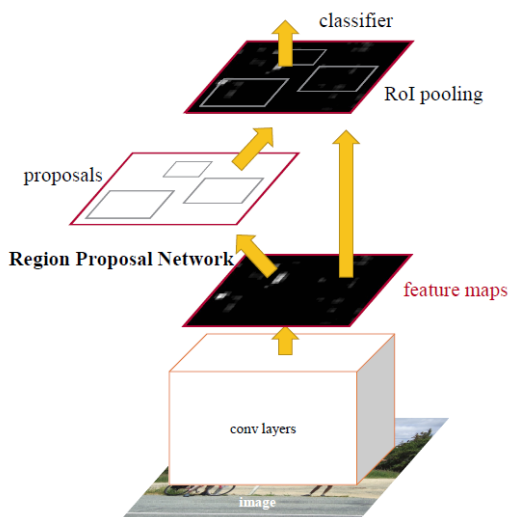


Figure 2.9: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the attention of this unified network. [83]

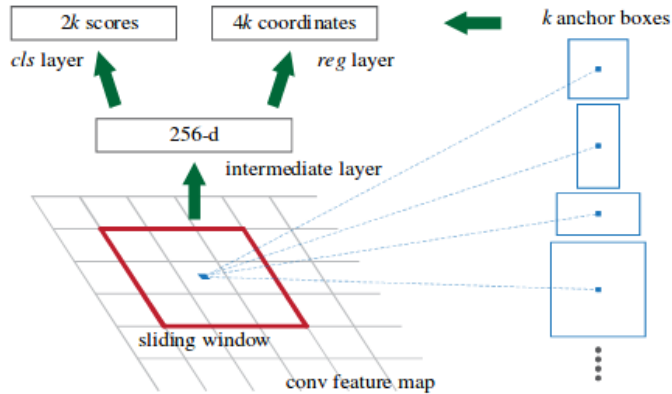


Figure 2.10: RPN in Faster R-CNN [84]. K predefined anchor boxes are convoluted with each sliding window to produce fixed-length vectors which are taken by cls and reg layer to obtain corresponding outputs

Region-based Fully Convolutional Networks (R-FCN):

For object detection, a prevalent family of deep networks can be included of two subnetworks: an unshared RoI-wise subnetwork and a shared fully convolutional subnetwork that is known independent of RoIs [33] and [84]. This classification is from pioneering classification architectures, for example, VGG16 and AlexNet [51]. It consists of various FC layers separated using a specific spatial pooling layer and a convolutional subnetwork. Image classification networks such as ResNets [41] and GoogleNets [96], [98] are fully convolutional. It is natural to make a fully convolutional object detection network (without RoI-wise subnetwork) to use to these architectures. Therefore, it cannot be inferior with like a naive solution [41]. In object detection, this inconsistency is because of the dilemma of respecting translation variance than enhancing translation invariance in image classification. In object detection, shifting an object within an image has to be indiscriminative in image classification whereas any translation of an object in a bounding box can be significant. At the expense of additional wise layers of the unshared region, a manual insertion of a layer in the RoI pooling in convolutions may break down translation invariance. Therefore, scholars [8] suggested an R-FCNs. For each category, the last conv R-FCN layer generates a total of a fixed grid of $k \times k$ along with k , two position-sensitive score maps, firstly,

and then for aggregating the responses from these score maps, a position-sensitive pooling layer of RoI is appended. It is different than Faster R-CNN. In order to carry out object detection in a fully convolutional architecture, more powerful classification networks may be used with R-FCN by sharing nearly all the layers. In [59], at a test speed of equal to 170 ms per image, latest results have been presented in term of both Microsoft COCO and PASCAL VOC data sets.

Feature Pyramids Network (FPN): As can be observed in Figure 2.11(a), for enhancing scale invariance, feature pyramids construct upon image pyramids, featured image pyramids, were selected and used in many object detection systems [26], [40]. However, memory consumption and training time enhance rapidly. Figure 2.11(d) shows that some approaches use a single input scale only for increasing the robustness to scale changes and representing high-level semantics. Moreover, image pyramids were generated at test time that makes an inconsistency among train/test-time inferences [33], [84]. Feature maps of various spatial resolutions can be generated by the in-network feature hierarchy in a deep ConvNet. The in-network feature hierarchy can introduce large semantic gaps made by different depths (refer to Figure 2.11(c)). Recently, scholars [62], [5] generated the pyramid starting from layers in middle (or just sum varied feature responses) to avoid utilizing low-level features.

The FPN [58] method is different from upper mentioned methods. As can be seen in Figure 2.11(b), for combining low-resolution as well as semantically strong features to high-resolution and semantically weak features, this method holds an architecture by a BU pathway, many lateral connections, and a top-down (TD) pathway. The BU pathway is the fundamental forward backbone ConvNet. It generates a feature hierarchy by downsampling the relating feature maps with a stride of 2. To construct the following TD pathway, the layers having the same size of output maps were gathered in a similar network step. Also, the output of the last layer of each stage was selected as the reference set of feature maps. To construct the pathway of TD, firstly feature maps of higher network steps were up-sampled and after that improved by those of the same spatial size from the pathway of BU through lateral connections. To decrease channel dimensions, a layer of 1×1 convolutional was added to the upsampled map. Then, the merge was obtained by adding element-wise. In the last step, to decrease the aliasing influence of upsampling, a 3×3 convolution was additionally added to each merged

map and the final feature map was produced. This approach was repeated to the finest resolution map was produced. The state-of-the-art representation can be obtained without sacrificing memory and speed because the feature pyramid extracts rich semantics of all levels. It also can be trained end to end by all scales. Meantime, FPN may be used to various stages of object detection (for example region proposal generation) and is independent of the backbone CNN architectures and to many other computer vision runs (for example instance segmentation).

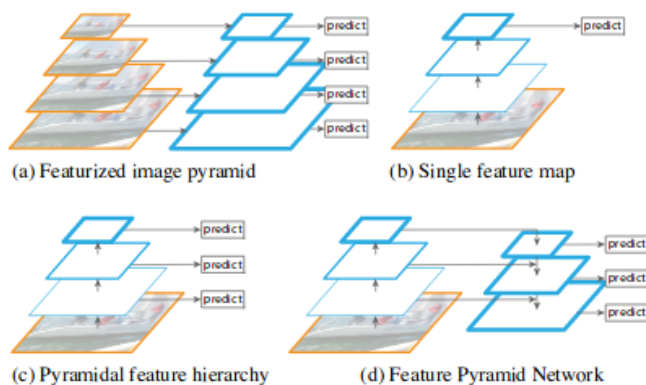


Figure 2.11: Main concern of FPN [58]. (a) It is slow to use an image pyramid to build a feature pyramid. (b) Only single-scale features are adopted for faster detection. (c) Alternative to the featurized image pyramid is to reuse the pyramidal feature hierarchy computed by a ConvNet. (d) FPN integrates both (b) and (c). Blue outlines indicate feature maps and thicker outlines denote semantically stronger features.

2.2.2 Regression-based object detectors

Region proposal-based frameworks are composed of several correlated stages, including region proposal generation, feature extraction with CNN, classification, and bounding box regression, which are usually trained separately. Even in the recent end-to-end module Faster R-CNN, an alternative training is still required to obtain shared convolution parameters between RPN and detection network. As a result, the time spent in handling different components becomes the bottleneck in the real-time application.

Mapping directly from image pixels to bounding box coordinates and class probabilities as well as one-step frameworks according to global regression classification decreases the time expense. In the present paper, some pioneer CNN models were reviewed. After that, we focused on two important frameworks, SSD [62] and YOLO [81]. Many studies have been performed to model object detection that is a classification or regression task before to SSD and YOLO.

Pinheiro et al. [77] proposed a CNN model with two branches: one generates class agnostic segmentation masks and the other predicts the likelihood of a given patch centered on an object. The inference is efficient since the class scores and the segmentation can be obtained in a single model with most of the CNN operations shared.

Erhan et al. [24] and Szegedy et al. [97] proposed the regression-based MultiBox to produce scored class-agnostic region proposals. A unified loss was introduced to bias both localization and confidences of multiple components to predict the coordinates of class-agnostic BBs. However, a large number of additional parameters are introduced to the final layer. Yoo et al. [106] adopted an iterative classification approach to handle object detection and proposed an impressive end-to-end CNN architecture named Attention Net. Starting from the top-left and bottom-right corners of an image, Attention-Net points to a target object by generating quantized weak directions and converges to an accurate object boundary box with an ensemble of iterative predictions. However, the model becomes quite inefficient when handling multiple categories with a progressive two-step procedure. Najibi et al. [70] proposed a proposal-free iterative grid-based object detector (G-CNN), which models object detection as finding a path from a fixed grid to boxes tightly surrounding the objects [70]. Starting with a fixed multiscale bounding box grid, G-CNN trains a regressor to move and scale elements of the grid toward objects iteratively. However, the G-CNN has difficulty in dealing with small or highly overlapping objects.

You only Look Once(YOLO): Redmon and Farhadi in [82] propose a method to harness the large amount of classification data we already have and use it to expand the scope of current detection systems. YOLO predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor. Instead of predicting coordinates directly Faster R-CNN predicts bounding boxes using hand-picked priors. Using only convolutional layers the region proposal network (RPN)

in Faster R-CNN predicts offsets and confidences for anchor boxes. Since the prediction layer is convolutional, the RPN predicts these offsets at every location in a feature map. Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn. They remove the fully connected layers from YOLO and use anchor boxes to predict bounding boxes. First, one pooling layer is eliminated to make the output of the network's convolutional layers higher resolution. They also shrink the network to operate on 416 input images instead of 448*448. They do this because they want an odd number of locations in their feature map so there is a single-center cell. Objects, especially large objects, tend to occupy the center of the image so it's good to have a single location right at the center to predict these objects instead of four locations that are all nearby. YOLO's convolutional layers downsample the image by a factor of 32 so by using an input image of 416 we get an output feature map of 13×13 . When they move to anchor boxes, the class prediction mechanism is decoupled from the spatial location, and instead, the class and the objectness for every anchor box is predicted. Following YOLO, the objectness prediction still predicts the Intersection over union of the ground truth and the proposed box and the class predictions predict the conditional probability of that class given that there is an object. The YOLO detection system is shown in Figure 2.13.

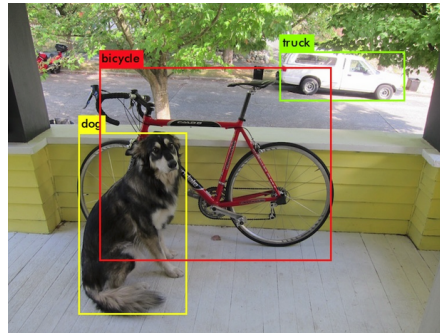


Figure 2.12: YOLO9000 can detect a wide variety of object classes in real-time. [82]

Single Shot Detector(SSD):

The first deep network-based object detector is proposed by Liu et al. [61]. In this innovation, Pixels or features are not resampled for bounding box hypotheses. It is also very accurate. Therefore, considerable improve-

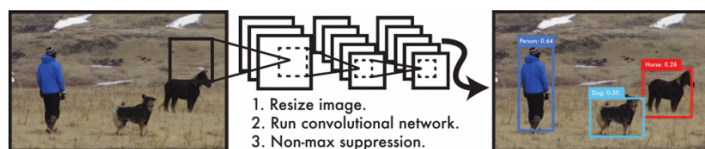


Figure 2.13: **The YOLO detection system.** processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence. [81]

ment happened in speed for high-accuracy detection. SSD approach has taken its root from the feed-forward convolutional network which generates bounding box collections with a fixed-size and measures the presence of object class instances in those boxes. After that, there is a non-maximum suppression step for making the final detections. VGG-16 network is used as a base. Then, an auxiliary structure is added to the network for producing detections with 4 key features of multi-scale feature maps for detection, convolutional predictors for detection and default boxes and aspect ratios, respectively. Architecture of SSD is shown in Figure 2.15.

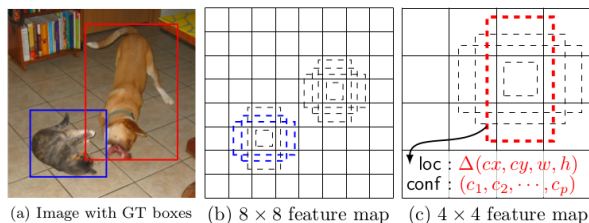


Figure 2.14: SSD framework [61]

2.3 Text Detection in scene images

Several methods have been proposed for text detection which are mostly based on sliding windows [46] and connected components (CCs) [104]. A new framework is introduced by Yi and Tian in [104] for extracting text strings that have different colors and sizes as well as arbitrary orientations scene images with a cluttered and complex background. The diagram of this

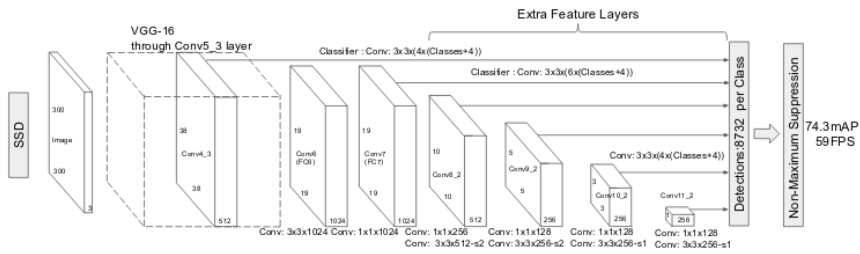


Figure 2.15: Architecture of SSD 300 [62]. SSD adds several feature layers to the end of VGG16 backbone network to predict the offsets to default anchorboxes and their associated confidences. Final detection results are obtained by conducting NMS on multiscale refined BBs.

framework is presented in Figure 2.16. It contains two main steps namely: a) Partitioning the image to find text character candidates based on color uniformity and gradient feature. In the aforementioned step, two methods are proposed for partitioning scene images into binary maps of non-overlapped connected components: color-based method and gradient-based method. To remove connected components, a post-processing method is implemented. b) Grouping of the Character candidate for detecting text strings according to the text characters joint structural features in every text strings like the size of characters, alignment of character as well as the two neighboring characters intervals. In the aforementioned step, two texts strings structural analysis are proposed including a text line grouping method and a character grouping method. This framework has some advantages compared to the existing methods. First of all, this framework allows text strings detection with different sizes, colors, and orientations. In contrast to the current methods which involve analyzing the single character independently, the text string structure focuses on distinguishing background interferences derived from the information included in the text. According to the experimental results, it can be said that the proposed framework has better performance on data set reading and text string detection with arbitrary orientations on the recently collected data set of scene text.

Yi et al.[7] provided some major contributions to robust detection of text strings with various colors, scales, orientation as well as clutter background from natural scene images, which are briefly summarized as follows:

- By integrating different types of features of text strings, this paper

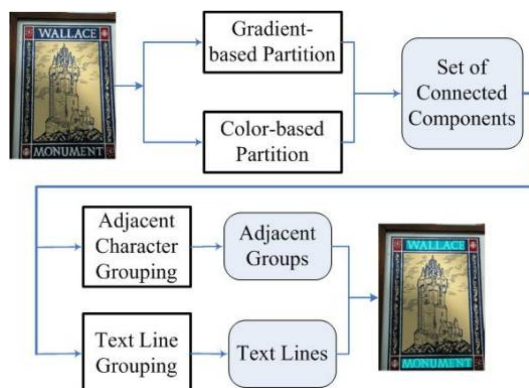


Figure 2.16: Flowchart of the proposed framework of text string detection. [104]

proposed a framework to robustly detect text strings with variations of orientation and scale from complex natural scene images with clutter background.

- A clear distinction is drawn between text character and text string through partitioning image for extracting candidate character components and grouping connected component for extracting text strings.
- A text character is modeled through stroke structure and local gradient features. Under this model, we develop a gradient-based partition algorithm to compute connected components of candidate characters. It is more robust and achieves better results than directly using morphological processing operators.
- Text string is modeled as text line and then a collection of text features are extended single character component to text line structure, which is implemented for text string detection in arbitrary orientations.
- An Oriented Scene Text data set(OSTD) is collected with text strings in arbitrary orientations which is more challenging compared to the current data sets for text detection.

One of the pioneer methods based on the analysis of connected components and their mutual position was proposed by Fletcher and Kasturi [28]. They developed a robust algorithm for text separation to separate text

strings from graphics, independent of string orientation and font size or style. In this algorithm, a simple heuristic is used based on the text strings characteristics. No character recognition is performed in this algorithm and the separation of characters from the text is done only according to the size and orientation. One of the drawbacks of this method is the fact that coping with the text touching graphics is neglected.

An improved approach is proposed by Tombre et al. [101] which can separate text touching graphical parts. Due to the scalability of the method proposed by Fletcher Kasturi, they decided to choose this method as the base of their text or graphics separation since they are dealing with various kinds of complex graphics documents. They have analyzed certain text features under the ICDAR 2003 train data set to achieve a distinctive feature set that can distinguish character objects from non-character objects. It contains three main stages as follows: first, the Segmentation stage to find character candidates. Second, Connected component analysis: it is based on fast-to-compute but robust features for accepting characters and discarding non-text objects and text line classifier: it is based on gradient features and support vector machines. Experimental results obtained with several challenging data sets show the good performance of the proposed method, which has been demonstrated to be more robust than using multi-scale computation or sliding windows. so robust it, according to the experimental results from some challenging data sets, the appropriate performance of the proposed method, which was assumed to be more effective than a sliding window or using multi-scale computation, is proved.

Chen et. al performed a study in [6]. They linearly adjusted the image intensities at the system input to enhance the contrast. Hence, MSER regions are efficiently extracted from the image [68] and they have been enhanced using Canny edges acquired from the original gray-scale image. Using a distance transform, the stroke width information is robustly computed. Then highly varied objects in terms of stroke width are rejected. After having been grouped pairwise, text candidates can form text lines. At last, words included in the text are separated and the segmented word patches will be given at the system outputs.

2.4 Summary of conclusions

In this chapter, first, main object detection algorithms in document image analysis, addressed in the state of the art, are discussed. A subset of these algorithms is used in the proposed models in this thesis. In particular, Faster R-CNN, which is one of the most robust algorithms in object detection, is used to detect objects (see Chapter 3) and for text recognition (see Chapter 4) in floor plan analysis scenarios. Also, the major applications in the object detection area, addressed by the academic and industrial communities, presented including, musical symbol detection, table detection, floor plan detection, formula detection, just to mention a few. Finally, major approaches in the state of the art to deal with text detection in scene image problems briefly discussed. These approaches are then adopted to support word location detection by improving text alignment in early printed books.

Chapter 3

Object Detection in Floor Plan Images

In this work, we investigate the use of deep neural networks for object detection in floor plan images. Object detection in images is important for understanding floor plans and is a preliminary step for their conversion into other representations. In particular, we evaluate the use of object detection architectures, originally designed and trained to recognize objects in scene images, for recognizing furniture objects as well as doors and windows in floor plans. Even if the problem is somehow easier than the original one, in the case of this research the data sets available are small and therefore the training of deep architectures can be problematic. In addition to the use of object detection architectures for floor plan images, another contribution of this work is the creation of two data sets that have been used for performing the experiments covering different types of floor plans with different peculiarities [110]

3.1 Introduction

Detecting and recognizing objects in floor plans is an essential task for the understanding of these graphical documents. Our research on this topic is part of the overall task of understanding of graphical documents for generating accessible graphical documents for visually impaired people [35] [66]. A

comprehensive perception of a floor plan is crucially important for blind people, allowing them to find their path as they face a new building. It is worth noticing that accessing the floor plan images for non-blind people enables them to download images of floor plans from real estate websites. A decision has been made to use images instead of CAD files because CAD files are only available for authors, and not distributed for the public. Object detection in natural images is basically defined as finding the location of objects in one image and classifying them. In many cases, the object location is based on the identification of the bounding boxes surrounding it. Also in this application, the identification of the object bounding box is sufficient for our purposes. Starting from widely studied architectures based on convolutional neural networks, a few object detectors have been recently proposed, such as Faster R-CNN [83], R-FCN, Multibox, SSD [61] and YOLO [82]. The key idea is to study, how the same modern convolutional object detection performs on our data sets given the special nature of them, like when the complexity of the shapes vary, the classes are not balanced and the size of object samples is small.

3.2 Previous Work

As in several domains also the document analysis community faced a growing use of deep learning in recent research work. Considering the application of deep learning in the area of graphics recognition, there are a limited, but interesting research works. Among various techniques, object detectors have been used to address various problems in document analysis. Symbol detection in on-line graphical documents is proposed in [48] where the authors use Faster R-CNN to address the task. In particular, the work addresses the recognition of mathematical expressions and flowcharts in handwritten documents by using the Tensorflow Object Detection API [45]. Another application of the latter API is related to handwritten music object detection [75] where the Faster R-CNN is used to recognize musical symbols. In both papers, the number of training items is relatively high and the results are evaluated only considering the accuracy of the model without taking into account the recall. Other authors used Faster R-CNN for page layout identification [105], for comic character face detection [78], and for arrow localization on handwritten industrial inspection sheets [36]. One recent effort to extract structural information from floor plan images is described in [21]

where the authors parse floor plan images to estimate the size of the rooms for interactive furniture fitting. They first perform wall segmentation by using a fully convolutional neural network, subsequently, they detect objects using a Faster R-CNN, and finally, they do optical character recognition to obtain the dimensions of the room. One interesting feature of this work is the combination of three methods to achieve the overall floor plan understanding. Unfortunately, very few details are provided in the paper about the use of Faster R-CNN for object location. Moreover, the floor plan dataset created by the authors only contains the ground-truth about the wall position.

In the work described in [10] the authors address the floor plan understanding by segmenting walls, windows, and doors. One of the main focuses of the paper is to address images with different notations (e.g. for walls or for furniture objects). The proposed techniques are tested on four floor plan datasets (named CVC-FP) which are freely accessible to the public. As discussed also in Section 3.4 the CVC-FP dataset only contains objects of six classes: `sink`, `toilet`, `shower`, `bath`, `door`, and `window`, without including furniture objects.

For the neural architecture, one important paper for this work is [44] where the authors evaluate and compare different object detection architectures. The goal of [44] is to identify the most successful architectures and support users when choosing one architecture based on various perspectives: speed, memory, and accuracy. To this end, the authors in [44] implement some modern convolutional detectors: Faster R-CNN, R-FCN, and SSD in a unified framework, as a part of the Tensorflow Object Detection API [45]. The authors pre-trained the architectures on several datasets, but the best performance was achieved by pre-training with the COCO dataset [60].

3.3 The Architecture

In this research, we work with one widely used Tensorflow Object Detection API [45] for an easy comparison of alternative architectures. We initially evaluated one COCO-pre-trained Single Shot Detector with MobileNets that we fine-tuned with floor plan images. We selected this architecture because it is a small and flexible model that has the benefit of fast training times compared to larger models, while it does not sacrifice much in terms of accuracy. In these preliminary tests, we also compared the SSD with Faster R-CNN with ResNet 50 and with ResNet 101. After these preliminary ex-

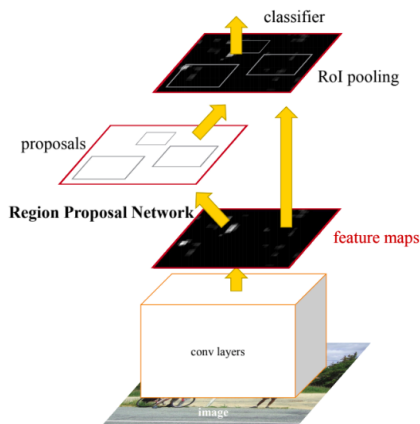


Figure 3.1: The internal architecture of the Faster R-CNN as a single, unified network for object detection (image from [83]).

periments, it turned out that Faster R-CNN performs significantly better than SSD. Moreover, comparing the performance of ResNet 50 with ResNet 101 on the floor plan datasets, there was no real difference. We, therefore, used Faster R-CNN with ResNet 50 as a basic model for our work.

Faster R-CNN is one of the most accurate and fast neural object detectors proposed so far. The internal structure of the network is as follows (see Figure 3.1): first, the image is passed through some convolutional layers to produce several feature maps. Then, the main component of Faster R-CNN, the region proposal network (RPN), uses a 3×3 sliding window and takes the previous feature maps as input. The output of the RPN is a tensor in a lower dimension. At this stage, each window location generates some bounding boxes, based on fixed-ratio anchor boxes (e.g. 2.0, 1.0, 0.3) and an "objectness" score for each box. These are the region proposals for the input image which provide approximate coordinates of the objects in the image. The "objectness" scores, if above a given threshold, determine which region proposal can move forward in the network. Subsequently, the good regions pass through a pooling layer, then a few fully-connected layers, and finally a softmax layer for classification and a regressor for bounding box refinement.

As previously mentioned to perform our experiments we use the TensorFlow Object Detection API. This is an open-source framework, built on top

of the widely used Tensorflow library, that takes care of the training and evaluation of the different architectures implemented. One interesting feature of the API is that it makes it easy to train different models on the same dataset and compare their performance. In addition to average precision performance per category, we extended the API to calculate the number of false negatives as well as the average recall per class.

3.3.1 False Negative Calculation

By default, the Tensorflow Object Detection API supports the PASCAL Visual Object Classes (VOC) 2007 [25] detection metric. This metric is designed to evaluate visual object detection and recognition models, which helps machine learning researchers have standard evaluation procedures.

In the detection metric, for a detection bounding box to be a true positive, three conditions must be true:

- The area of the intersection of the detected bounding box B_d and the ground truth bounding box B_{gt} over the union area of the two bounding boxes must be greater than 0.5, according to the following equation:

$$r_i = \frac{area(B_d \cap B_{gt})}{area(B_d \cup B_{gt})} > 0.5 \quad (3.1)$$

- The class label of the detection bounding box and the ground truth bounding box must be the same.
- The probability of the object's recognition must be greater than some specific thresholds. In most cases, and also in this work, we consider the object as found if the probability is higher than 0.50.

To find false negative detections we first matched all the detections to objects in the ground truth. True/false positives are determined and detections matched to difficult boxes are ignored. In the next stage, the ground truth objects that have not been detected are determined as false negatives.

After computing the true positives and false negatives number for each category it is easy to calculate the average recall in addition to the average precision computed by the API: $Recall = \frac{TP}{TP+FN}$.

3.4 Floor Plan Datasets

To evaluate the object detection in floor plans, we need one or more labeled datasets. In the past decade, some datasets have been proposed for evaluating research on floor plan analysis. The SESYD dataset [17] contains synthetic floor plans where furniture objects are randomly placed on a few fixed floor plan layouts. Even if this approach for dataset generation is very interesting, the actual dataset contains only ten floor plan layouts and the objects come from a limited number of categories (for instance, there is only one model for the bed). Moreover, the generated floor plans are somehow unrealistic with very small beds or similar mistakes. Another dataset widely used for this research has been proposed in [64]. This dataset contains 90 actual floor plans generated from one architectural firm. While more realistic than the others, these floor plans contain only a few objects and therefore are not suitable for the research carried out in this work. A deeper analysis of data sets proposed to support research in floor plan analysis is proposed in chapter 5.

To work with realistic images we first created one small dataset (referred to as *d1*) using the images that show up in Google’s image search. This dataset consists of 135 images of variable size containing objects in 15 classes and a total of 4973 objects (in the experiments we considered 2697 objects in the training set, 1165 objects in the validation set, and 1111 objects in the test set). In Figure 3.4 we show one example of floor plan in this collection, while Figure 3.2 shows the distribution of objects in the different classes. Some object types are not present in all the images. For instance, the floor plan of an office might not have any bed in it. Among all the classes, the oven is the rarest one and the door is the most frequent one.

The second data set that we gathered is called *ISTA*. It should be noted that *ISTA* data set is called *ISTA* in [110]. This data set contains Middle Eastern floor plans, with object shapes different from the ones in *d1*. Another important feature is that the floor plans in *ISTA* come from one architectural firm and are therefore more homogeneous in their content. The *ISTA* data set consists of 300 images, but only 160 images have been labeled so far. The 160 images contain objects in 12 classes and a total of 7788 (in the experiments we considered 4535 objects in the training set, 1457 objects in the validation set, and 1796 objects in the test set). In Figure ?? we show one example of floor plan in this collection, while Figure 3.3 shows the distribution of objects in the different classes. As a particular property of

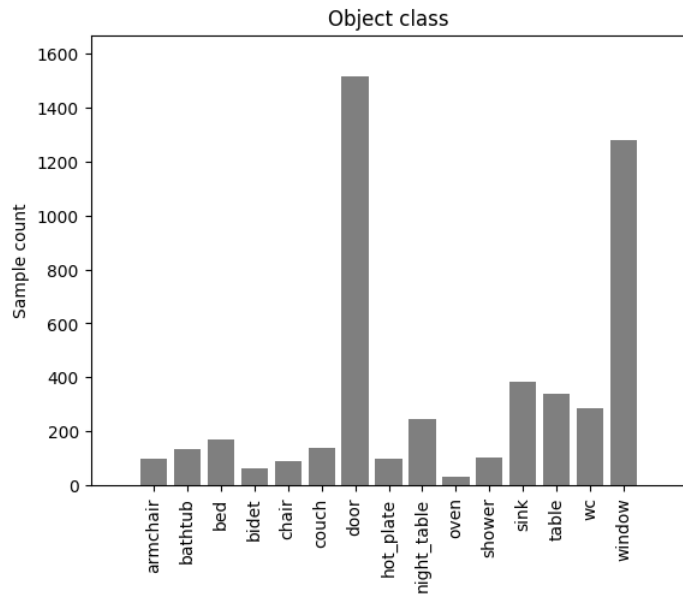


Figure 3.2: Object class distribution of the *d1* data set



Figure 3.3: Object class distribution of the *ISTA* data set

these floor plan data sets, it is worth to note that the images are mostly grayscale and contain simple shapes. As we will see in the experimental part this property has a positive effect on the performance of the model, compared to data sets that contain images with more complex features and more noise. The data set *d1* has the greatest imbalance in the number of objects in each class. Moreover, images in *d1* have more diversity. For example, almost none of the objects in *ISTA* are filled with color, while in *d1* all the floor plans are painted, for presentation purposes. It is noted that since both Flo2plan and *d1* data sets are real ones (as they downloaded using Google search), they are considered as noisy ones. Also, the images in these data sets mostly are different with respect to their size. As a result, object and text detection in them are highly more challenging compared to *ISTA* data set.

3.4.1 Experimental Results

The Faster R-CNN is first trained on the *d1* data set, which contains 135 images. As mentioned earlier, this data set is substantially diverse and contains

random non-standard images collected from the Internet.

In the first experiments performed on a smaller data set with the default configuration of the API, the results on the validation set were not satisfying with a maximum mean average precision of about 0.26.

To aid generalization, we threw in a few of data augmentation options. In particular we considered `random horizontal flip`, `random vertical flip`, `random rotation 90`, and `random RGB to gray`. These options are provided by the Tensorflow Object Detection API. In addition to data augmentation, we also changed the scales and aspect ratios of anchor generator to take into account the peculiarities of the floor plan objects.

Table 3.1: Final Evaluation Results

Dataset	Objects	False Negatives		Mean Average Precision		Mean Average Recall	
		Val	Test	Val	Test	Val	Test
<i>d1</i>	1111	411	445	0.32	0.31	0.60	0.56
<i>d2</i>	1796	87	102	0.83	0.86	0.92	0.92

Final evaluation results of the d1 data set after 46916 training steps, and the ISTA data set after 18550 training steps

With the above mentioned modified configuration we also ran our experiments on the *d2* data set. After stopping the training considering the validation set, the mean average precision and recall on the test set for both data sets are shown in Figure 3.4.

Taking into account the features of the two data sets it is not surprising that the best results are achieved on the *ISTA* data set with a mean average precision of 0.86, and a mean average recall of 0.92. Part of the difference in performance is probably related to the special nature of the data set: compared to *d1*, the objects are cleaner, less diverse and not different across images other than rotation and scale. As it turns out, the performance of the model is not too much affected by an imbalanced data set (*ISTA*). For instance the model achieves 0.80 average precision for the `couch` class that is the less frequent one. At the same time the model achieved 0.93 average precision for the `door` class which has at least 10 times more samples than the `couch` class. Concerning the average recall it is very interesting

to notice that the test images had zero false negatives for the **hot-plate** and **bed** classes. On the basis of the superior performance of the network on the *ISTA* data set we wanted to explore more in details the possibility of doing some transfer learning of this network to improve the performance on the *d1* data set. As we can see in Table 3.4 by finetuning on the *d1* data set one network previously trained on the *ISTA* data set we achieve a 0.39 mean average precision and 0.69 mean average recall. This is better than the previously mentioned results obtained by finetuning on the *d1* data set one network previously trained on the COCO data set.

Table 3.2: Average precision and recall calculated by category, for the *d1* and *ISTA* data set after 46916 and 18550 training steps, respectively

Class	Average Recall(Val)		Average Recall(Test)		Average Precision(Val)		Average Precision(Test)	
	d1	ISTA	d1	ISTA	d1	ISTA	d1	ISTA
armchair	0.92	0.97	0.50	0.95	0.21	0.92	0.21	0.92
bathtub	0.72	N/A	0.80	N/A	0.57	N/A	0.57	N/A
bed	0.56	1.00	0.70	1.00	0.47	0.98	0.47	0.98
bidet	0.57	N/A	0.54	N/A	0.11	N/A	0.11	N/A
chair	0.58	0.68	0.44	0.76	0.12	0.50	0.12	0.62
couch	0.75	0.80	0.52	0.88	0.46	0.52	0.48	0.80
door	0.63	0.97	0.63	0.95	0.60	0.94	0.60	0.93
hot_plate	0.67	0.97	0.52	1.00	0.40	0.92	0.39	0.96
night_table	0.65	0.93	0.34	0.81	0.37	0.86	0.37	0.79
oven	0.12	N/A	0.55	N/A	0.01	N/A	0.01	N/A
shower	0.55	0.97	0.71	0.93	0.19	0.90	0.16	0.90
sink	0.57	0.95	0.47	0.94	0.28	0.80	0.28	0.85
table	0.58	0.94	0.46	0.98	0.33	0.86	0.33	0.93
wc	0.45	1.00	0.65	0.95	0.13	0.79	0.13	0.78
window	0.58	0.91	0.54	0.88	0.49	0.87	0.48	0.85

When the results of different data sets have been compared, it shows how the same modern convolutional object detector performs on your new data sets given the special nature of them, like when the complexity of the shapes vary, the classes are not balanced, and the total size of object samples are

small. As we can see in Table 3.3 the result on *d1* and *ISTA* and *SESYD-ROBIN* data set has been compared. Some objects in this data set, such as armchairs, bathtubs, are included in all of these data sets. As shown, although the value of the data set in the literature is larger than the data set mentioned in this study, the mAp in some objects is better than them.

Table 3.3: Average precision *d1* and *ISTA* and *SESYD-ROBIN* data set

Class	Average Precision		
	<i>d1</i>	<i>ISTA</i>	<i>SESYD-ROBIN</i> [93]
armchair	0.21	0.92	0.89
bathtub	0.57	N/A	0.68
bed	0.47	0.98	0.92
bidet	0.11	N/A	N/A
chair	0.12	0.62	N/A
couch	0.48	0.80	N/A
door	0.60	0.93	N/A
hot_plate	0.39	0.96	N/A
night_table	0.37	0.79	0.67
oven	0.01	N/A	N/A
shower	0.16	0.90	N/A
sink	0.28	0.85	0.55
table	0.33	0.93	0.94
wc	0.13	0.78	N/A
window	0.48	0.85	N/A

Table 3.4: More Transfer Learning

Dataset	Objects	False Negatives	Mean Average Precision	Mean Average Recall
COCO - <i>d1</i>	1111	445	0.31	0.60
ISTA - <i>d1</i>	1111	368	0.39	0.69

The final evaluated results of the pre-trained models that is fine tuned on data set *d1*.

3.4.2 Discussion

From the experiments performed on the two data sets we can notice that by using convolutional object detectors, the recognition performance is not too much influenced by the class imbalance in the training set. The only exception is related to the `oven` class from data set *d1*, with extremely low performance, is probably due to the very low sample size and the variability of the appearance of this object in the data set. On the other hand, the `door` and `window` classes are responsible for 52% of the false negatives in *d1* validation set, while they make up 58% of the object samples. In these two classes the model performs relatively well in terms of average precision, but it is not capable of detecting many objects. At first, this result might contradict the intuition that more samples led to better performance. However, it is important to recall that the performance of the model in one class heavily depends on the diversity of object samples. It is useful to remark that in the case of doors and walls the objects are connected to the walls while other objects are usually more isolated in the rooms. The diversity of walls and doors is, therefore, higher with respect to other classes because of the variable context. Another source of errors for windows is also the higher variability of the aspect-ratio with respect to other objects that in most cases are simply scaled and rotated, in particular in data set *ISTA* where reasonable performance on the data set is obtained. Regarding the three most frequent classes in *ISTA* `armchair`, `door`, and `window`, it can be seen from Table 3.2 that the model is nearly perfect in terms of average precision and the class `bed` (whose items are more regular) achieve an average precision of 98%.



Figure 3.4: An inference result of the model trained on the *d1* data set. False negatives have black bounding boxes and detection bounding boxes are colorful. Note how new shapes and colors in this test image damage the performance of the model.



Figure 3.5: This shows an inference result of the model trained on the *ISTA* dataset.

3.5 Conclusion

In [110] two different floor plan data sets have been created to cover different architectures and drawing conventions of floor plans from all over the world. The performance of an object detector which is originally designed for detecting objects in natural images was tested to identify objects in the floor plans in these data sets. The floor plan has an essential misrepresentation issue in terms of the sample size of objects. To better analyze the performance, false-negative objects of each class have individually been counted to find out whether the detection results suffer from differences in the number of samples for each class. We noticed that the performance of the model in a class heavily depends on the diversity of object samples, object rotation and scale somehow outweighing the role of sample size. It is interesting also to notice how a network that pre-trained from another domain (COCO pre-trained Faster-RCNN with Res Net 50) can perform well on the floor plan data sets using just a one hundred images. To further improve the results on this task, it is recommended to either collect larger data sets to cover different graphical conventions or implement data augmentation techniques more suitable for object detection in floor plans.

Chapter 4

Text Recognition and Alignment

In this chapter, a technique for transcript alignment in early printed books by using deep models together with dynamic programming algorithms has been done. The proposed technique is evaluated on a publicly available digitization of the Gutenberg Bible while the transcription is based on the Vulgata, a late 4th century Latin translation of the bible. Also, a model is presented to improve text recognition in floor plan images, using traditional text detection methods with image processing techniques and deep learning-based methods [80].

4.1 Introduction

Document image analysis involves the processing of both handwritten and printed document images. Text recognition in graphical documents is an important task in document image analysis. In particular, textual information in engineering drawings, like floor plans, is momentous for further analysis, specifically when the semantics of rooms should be detected. From a broader point of view, reading text in images is a relevant topic in computer vision with several applications in automatic digitization of documents, real-time multi-language translation, augmented reality, support to blind people, etc. The first work presented in this chapter is related to a broader project whose aim is to allow visually impaired users to access graphical information [67]. By considering the recognized text it is possible to provide visually impaired users useful information about the room's functions and size. Hence, by ex-

tracting and gathering primary information obtained from the floor plan, we can gain information about the entire building.

On the other hand, training deep architectures require the availability of large collections of annotated data. In text alignment, the recognition of early printed books with neural models initially proposed for object location in scene images has been addressed. The main purpose of this work is the alignment of digital images of the first pages of the Gutenberg's Bible (the *Genesis* book) with a text (the Biblia vulgata) that was the basis of Gutenberg's work. Besides applications for training text recognition networks, text alignment with a reference text is useful also for paleographic studies where different instances of specific words need to be compared by scholars. Studies in the area of text recognition of interest for this work can be categorized into two groups: works using convolutional models for object detection in the field of document analysis that I will explain completely in the related section and text alignment algorithms.

The network training is performed on a semi-automatic transcription of the document to be recognized. First, an initial model to recognize stop-words and hyphens in a set of images with a well-aligned transcription made by hand has been trained. Using the model prediction on a loosely annotated text we then train another model able to detect some words inside unknown images. As I have mentioned before, among of some convolutional architectures that have been recently proposed for addressing object detection such as, Faster R-CNN ([83]), R-FCN, Multibox, SSD ([61]) and YOLO ([82]), Faster R-CNN is one of the most accurate and fast neural object detectors proposed so far. For example, in [110] I used Tensorflow Object Detection API in object detection in floor plan images. I also compared the SSD with Faster R-CNN with ResNet 50 and with ResNet 101. After these preliminary experiments, it turned out that Faster R-CNN performs significantly better than SSD for the task addressed in the paper. Also, [69] proposes a trainable CNN model for text detection considering the achievement of Faster R-CNN. The contribution also includes an architecture with multiple RPNs and an RoI merge layer in addition to the original Faster R-CNN. Therefore, for having the best results, Both deep networks in this research are based on Faster R-CNN initially proposed to detect objects in scene images.

The second category of text alignment studies is related to text alignment in the transcribed images of documents. One method was proposed by [79] for aligning historical documents with transcriptions, through which

a representative of each letter is cropped in the historical texts and then, the transcribed images are changed to synthetic word images by showing the letters in the transcription. Then, the images are classified into a group of interrelated components in the source text using a dynamic programming algorithm. [85] designed a method to align the handwritten text images with the related transcripts by which the texts are classified into certain lines to be detected. [27] introduced one data set composed of images and corresponding transcriptions of a Latin manuscript. The alignment system is based on character Hidden Markov Models to deal with alignment difficulties. Besides, [38] have presented a system, through which the image of a historical text directly with a synthetic image created from the transcript. [94] put forward a transcript mapping technique that is guided by the number of words as well as the characters per word of a text line. Using a scoring algorithm, the proposed method synthesizes the results of a local and a global approach. [86] presents a system for aligning transcript letters to their coordinates in a manuscript image.

A suitable GUI and an automatic line detection method enable the user to perform an exact alignment of parts of document pages. The system employs an optical flow engine for directly matching at the pixel level the image of a line of a historical text with a synthetic image created from the transcript's matching line. [56] proposed word-to-word alignment in a segmentation-free and learning-free way. The best word-to-word match lies in an adapted edit cost between signatures extracted on Unicode characters and on the images. The results are evaluated on the "Queste del Saint Graal" (13th c.) by palaeographers through a validation interface. One important feature of the work presented in this thesis is that in the data used in our experiments a diplomatic transcription of the images is not available. As a proxy of this transcription, we start from a text version of the Bible (the Vulgata) that scholars consider as source for Gutenberg's work. Since there is no exact transcription of the images there are missing (or new) words in the text and there are also different spellings for the same words.

4.2 Text alignment in early printed books

The basis of the text alignment method described in this work is word location by using the Faster neural network. In particular, I look for the five most common words and for the hyphens that split words at the end of text

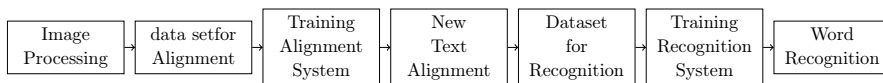


Figure 4.1: Pipeline of the whole system.

lines. The aligned text is then used to increase the amount of data, to train a neural network able to detect words inside new scanned pages. The word alignment system proposed in this work is made out of six main parts that are described in the rest of the chapter: preliminary segmentation of the training pages at word level; construction of training data set using well-aligned text; training of the neural network for segmentation; text alignment on loosely annotated pages; training of the neural network for word recognition, the test of word recognition.

4.2.1 Text Segmentation

The scope of text segmentation is to detect the words and the hyphens to generate the data set to train the first network. Segmentation to detect columns and rows is used to analyze deriving cuttings for word isolation. A different technique has been used to find the hyphens, through which the final part of each row (about 14 pixels) is isolated and with the help of the transcription, by which we can realize that small cutting are a hyphen or not.

Image pre-processing

In this phase, a pre-processing of the images is implemented to clean and rectify the pages (Figure 4.3). Since some pages have a content inclination that does not allow an accurate analysis the first step is a skew correction. The edges of the images are then cut to make the images as clean as possible removing the noise in the page borders. For improving the next phases one white *padding* is added. Eventually, a reverse binarization is applied to each image to detect the rectangle's rotation angle with the minimum range containing all white pixels to ignore it.

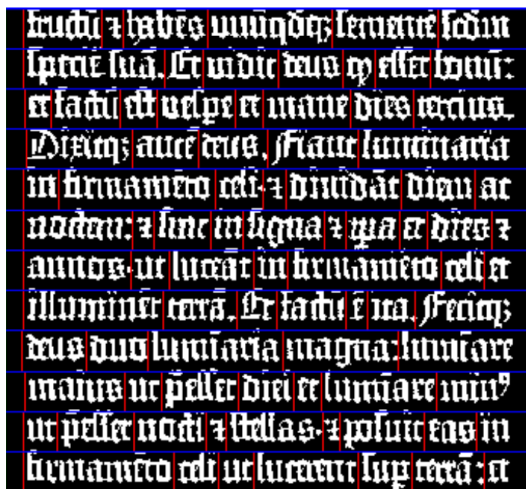


Figure 4.2: Word segmentation by projection profiles.

Columns, rows and word segmentation

After the pre-processing, we need to perform a preliminary column, row, and word segmentation. After binarizing the images with a global threshold we have used the projection profile method on the binarized image since the text lines are regularly spaced. Considering modern printing in the Gutenberg bible, like in medieval manuscripts, the words are a quite close one to the other. For word detection, we empirically separated words when we found three consecutive white pixels. This algorithm is inaccurate on general texts, however, in our case, we know from the reference text how many words are expected in each text line. We, therefore, modify the expected white space between words decreasing it if the detected words are less than expected and increasing it if the words are more than the real ones. To take into account the GPU memory, when applying the object detection algorithm we split each column in three slightly overlapping parts. Figure 4.2 shows the segmentation results of a portion of the column.

Word location training

To find words we trained a convolutional neural network using the Faster R-CNN with a Resnet-50 as a basic model. The word location network is



Figure 4.3: Genesis first page, before (left) and after (right) pre-processing.

trained with three classes: ‘background’, ‘=’, ‘word’. In this case, ‘word’ means a generic word and ‘=’ represents a hyphen at the end of the next row. The model was trained for 25,000 epochs. The network reaches a good level of accuracy after 7000 iterations.

We have evaluated the results of word location using the COCO detection metrics. The results are shown in Table 4.1. Noticing some examples of results in Figure 4.5, we can see how the network has difficulty to detect objects with small sizes, but the network’s behavior is interesting in other cases.

Table 4.1: Results of preliminary word location.

Metric	AP	AP50	AP75	APs	APm
Value	0.682	0.843	0.817	0.651	0.795

<pre> _P17_C0_P1 ad pueros suos expectate hic cum asino ego et puer illic usque properantes postquam adoraverimus revertetur ad vos tulit quoque ligna holocausti et inposuit super Isaac filium suum ipse vero portabat in manibus ignem et gladium cumque duo pergerent simul dixit Isaac patri suo pater mi at ille respondit quid vis filii ecce inquit ignis et ligna ubi est victima holocausti dixit autem Abraham Deus providabit sibi victimam holocausti filii mi pergebant ergo pariter et venerunt ad locum quem ostenderat ei Deus in quo aedificavit altare et desuper ligna </pre>	<pre> _P17_C0_P1 x x x x x x = x x x x x x x = x x x x x = x </pre>	<pre> _P17_C0_P1 ad pueros suos expectate hic cum as = ino ego et puer illic usque properantes post = quam adoraverimus revertetur ad vos tu = lit quoque ligna holocausti et inposuit super Isaac filium suum ipse vero portabat in manibus ignem et gladium cumque duo pergerent simul dixit Isaac patri suo pater mi at ille respondit quid vis filii ecce inquit ignis et ligna ubi est victima holocausti dixit autem Abraham Deus providabit sibi victimam holocausti fi = li mi pergebant ergo pariter et venerunt ad locum quem ostenderat ei Deus in quo aedificavit altare et desuper ligna </pre>
---	---	--

Figure 4.4: Main phases in text alignment.

4.2.2 Dynamic programming

Starting from the output of the preliminary word location we can rebuild the structure of each text row. First of all, we delete overlapping bounding-boxes and add missing ones by computing the distance between two consecutive bounding-boxes and adding a bounding box if the distance is larger than a threshold. We generate a “guide text” composed of ‘x’ and ‘=’. The ‘x’ symbol represents a generic word, the ‘=’ symbol represents a hyphen. We consider bounding-boxes as belonging to the same row if they have similar ordinates (y values). For alignment, we dispose the unaligned text according to the structure of the “guide text” considering each hyphen. Figure 4.4 represents all the different phases in the text alignment process. When generating the training data for the first network we know the portion of reference text corresponding to each line (right part of Figure 4.4) and with the help of the guide (central part) it is possible to know the meaning of each word. The result of this word alignment is shown in Figure 4.7. When dealing with loosely aligned pages only the left part of Figure 4.4 is available and in this case, we need to use a dynamic programming-based approach.

When a line-by-line alignment of the reference text is not available we need to follow a different strategy based on two steps. First, we train a word location network used to find the five most common words in the text: *ad*, *cum est*, *et*, *in*. This network is trained on the output of the previous alignment and then used to find reference words in the loosely aligned text where we know the reference text for each of the six portions of the page. The output is a string containing “X” in the case of a generic word and placeholders “A”-“D” corresponding to each stop word (**BBX in the second line** in Figure 4.6). From the reference text, we can also compute a similar string that we consider as a ground truth. (**GT in first line** in

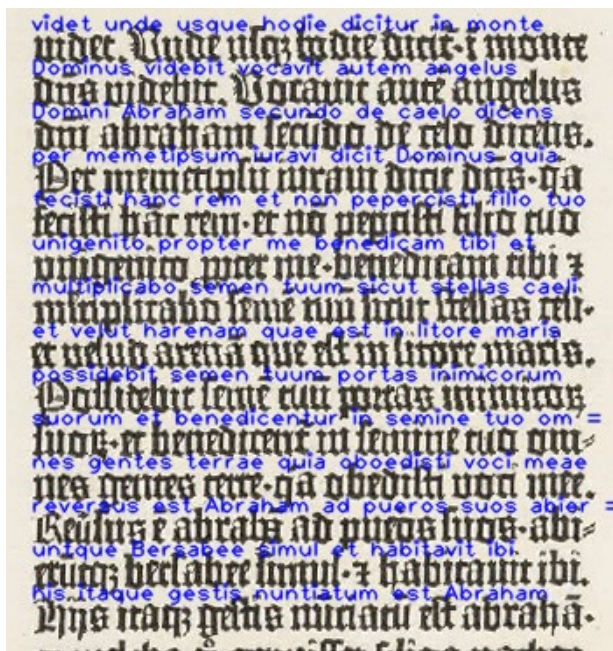


Figure 4.7: Alignment result.

thing that is needed is to change the label of the bounding box that has been detected for being such an error and change it to the proper label. The algorithms to detect these errors count the number of characters in the LCS at the left and the right of these characters in the original sequence. If the number of characters on the left parts of both sequences is the same, and if this is also true for the right part, then we can exchange the characters. In Figure 4.6, we show the results of the LCS and its improvement (LCS2) on one part of the page and its transcript. The characters in red and blue are the ones that were not selected by the first LCS algorithm. So there will not be those letters and the related bounding boxes in the output because they are considered as mistakes of the detector. In blue are the errors that could be corrected by the system, so the bounding boxes related to those words have been added again to the series of correct bounding boxes that are considered to be aligned. Figure 4.8 shows the improvement on test page 17. We can see that some improvement has been made, especially when the word location misses special words that we know are present in the ground

truth. The system is sometimes able to replace correctly those words with the correct bounding box. The skip-lines in each output are determined by the changes in the y coordinates of the sequence of bounding boxes so it is a good representation of what the system can do. Underscores represent parts where the realignment algorithm struggled to find a good match and we decided to delete this word and the corresponding bounding box. This phenomenon occurs in mainly three cases: when there is a hyphen at the end of a line, or when the system deletes a word because it is too hard to find a correspondence or because some words are over-segmented in the input. In the third case, the remaining bounding box might have satisfying accuracy in terms of positioning, but not so much in terms of IoU. In general, we can conclude that lines with underscores have an uncertainty slightly increased.

IN :	GT :	OUT:
Dxxxx	xxxxEx=	_xxx_x
xxAxxxx=	xxAxxxx=	xxAxxx_
xxxDxx=	xxxDxx=	xxxDx_
xBxxxAx	xxxxAxx	x_xxxAx
xxxxxxB	xxxxxx	xxxxxxA
xxxxxx	xxAxxx	xxxxxx
xxxxxx	xxxxxx	xxxxxx
xxxx	xxxxxxx	xxxx
xAxxxxxx	xxxxAxx	x_xxxxxx
xxxxxx	xxxxxx	xxxxxx
xxxxxx=	xxxxxx	xxxxxx_
xxxxxxA	xxxxxxA	xxxxxxA
xDxxxxx	xDxxxxx	xDxxxxx
Bxxxxxx	BxxxAxx	BxxxAxx

Figure 4.8: Results of the LCS-based realignment process on a part of page 17

4.2.3 Dataset

The pages used in the experiments have been downloaded from the Bayerische Staatsbibliothek Muenchen, Germany ¹

¹ Biblia [Gutenbergbibel] ; [1-2] [Mainz] [1454/55, nicht nach 1456.08.24. bzw. 08.15.] [BSB-Ink B-408 - GW 4201 - ISTC ib00526000] <http://daten.digitale->













Word	Image	%	Word	Image	%
et		52.65%	est		41.04%
		13.68%			47.01%
		33.65%			11.94%
in		30.80%	cum		78.43%
		69.68%			21.56%
ad		5.53%			
		92.47%			

Figure 4.9: Different forms of words in the Gutenberg Bible.

At this point, we analyzed the problem of word recognition. Having two available texts (pages 0-16 and pages 17-34), we have decided to train the network to recognize the most five frequent words in common between the two texts. These result to be "et", "est", "in", "cum", "ad".

To create the new COCO dataset, we have associated at each word cut (described by page, column, row, position in the belonging row) to the corresponding text word. Thanks to this informations, we have generated a new COCO data set having similar characteristics to the previous one. In this case categories are six: {"__background__", "et", "est", "in", "ad", "cum"}.

Incunabula (books printed before 1501) share with medieval manuscripts an extensive use of ligatures and abbreviations as we observed in our previous work on these data [92]. As a consequence in the Gutenberg bible, several words are printed in different ways and this can be problematic for word recognition. Different representations for the five-stop words considered in the experiments are shown in Figure 4.9 and together with their relative distribution in Figure 4.10. For instance "et" and "est" can be represented in three ways: one with the first word capitalized and two for the regular form one of which is abbreviated. It interesting to notice that "est" (to be in Latin) remained as such in modern French, but became é in modern Italian.

Table 4.2: Segmentation results.

Total Rows	Rows with correct cuts	Rows with wrong cuts
1674	1334 (74,4%)	340 (21,3%)

In the experiments described in this paper, we consider three sets of data having different ground-truth annotations.

The first training (Set I) has been executed over the first 17 pages of Genesis, that is those having a correctly aligned text checked by hand. In this case, the reference text of each line is checked line by line (including words split by hyphens). However, the accurate position of each word in the line is not annotated.

The second training has been executed with another set (Set II) of 17 pages of Genesis (20-34), where the reference text is only split in parts corresponding to regions in the page processed by Faster R-CNN (there are six regions for each page). The test set is made by pages 19 – 19 in the Genesis. In this case, the position of each word and the corresponding transcription is annotated in the ground truth.

Preliminary word segmentation

In the first test, we evaluate the results of preliminary word segmentation based on projection profile and alignment with the reference set in Set I as described in Section 4.2.2. Numerical results are shown in Table 4.2 among the 340 wrong rows there are 189 rows that have a smaller number of cuts and 151 with a larger number of cuts. Since this alignment is used to train the stop-word recognizer not all these errors necessarily impact the training performance as will be discussed later.

Evaluation of LCS alignment

To evaluate the efficiency of the dynamic programming part, we have developed an evaluation based on both the text accuracy and the bounding box accuracy. We count the number of words that are identical as the one in the transcript and that have an Intersection over Union (IoU) with its relative bounding box in the ground truth above a certain threshold. We tested this with different threshold values to study more precisely what our model was

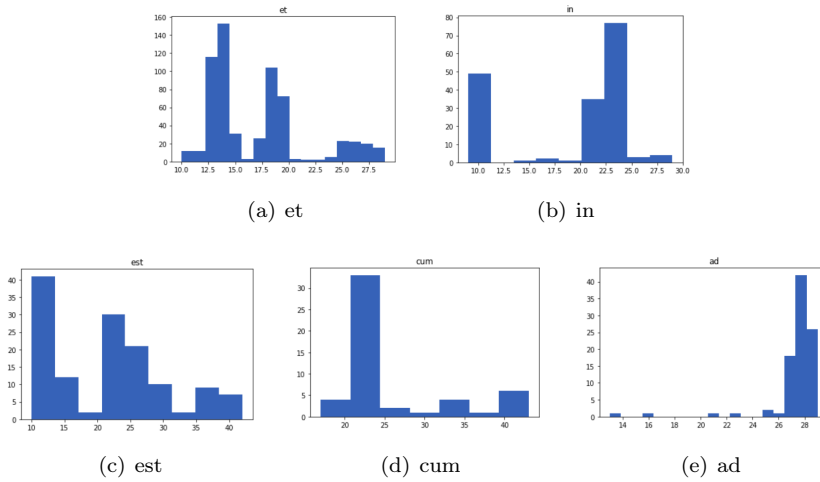


Figure 4.10: Distribution of most common words

doing and to observe if we had a significant improvement in the quality of the data set while using dynamic programming. The results for different IoU thresholds are shown in Table 4.3. The results seem to show a slight but constant improvement in the quality of the data set in all of the possible configurations.

Table 4.3: Evaluation of the dynamic programming process with different IoU thresholds

Threshold	input acc	LCS acc	LCS2 acc
0.4	0.931	0.943	0.939
0.5	0.824	0.837	0.833
0.6	0.677	0.688	0.685
0.7	0.565	0.574	0.571
0.8	0.484	0.492	0.489

Word location tests

To evaluate the LCS-based alignment from a different perspective we compared the performance for the recognition of stop-words in the test set when

training a word location on ground truth generated with the well-aligned reference text is Set I and when training on Set II with the loosely aligned text.

Regarding the training on Set I, we have employed three hours with 30.000 iterations, setting a batch-size equal to 2. The network reaches a good level of accuracy after 10.000 iterations. The second training has executed on Set II where we needed three hours with 30.000 iterations, setting a batch-size equal to 2. The network reaches a good level of accuracy after 10.000 iterations.

Table 4.4: Word recognition on test set.

	<i>ad</i>	<i>cum</i>	<i>est</i>	<i>et</i>	<i>in</i>	Total
Occurrences	27	10	24	103	66	230
Detections	26	8	22	91	53	200
training on set I	(96,30%)	(80%)	(91,67%)	(88,35%)	(80,30%)	(86,96%)
Detections with LCS	16	2	11	73	22	124
training on set II	(59,26%)	(20%)	(45,83%)	(70,87%)	(33,33%)	(53,91%)
Detections	16	7	12	71	44	150
set II - filtered cuts	(59,26%)	(70%)	(50%)	(68,93%)	(66,67%)	(65,22%)

After training the models over the corresponding datasets, these have tested on the Test Set; the results are shown in Table 4.4 that shows in the *Occurrences* column the total number of each stop words (*ad*, *cum*, *et*, *est*, *in*) in the test pages.

Analyzing the case of the trained network over Set I, we can see that the percentage of correctness is about 90%. In the second case (Set II), the performance of the network is worse than the other dataset because the second network has been trained with training data generated from loosely aligned reference text. For Set II we have worse results because of some imperfection presents in the text. Therefore, we filtered some cuts erasing those longer than the expected word length. For this reason, it has been possible to reduce errors as shown in the last row of Table 4.4.

4.3 Text Recognition in Floor plan Images

As we discussed earlier in this chapter, text recognition in graphical documents is an important task in document image analysis which deals with the processing of graphical document images. In floor plan scenarios, text de-

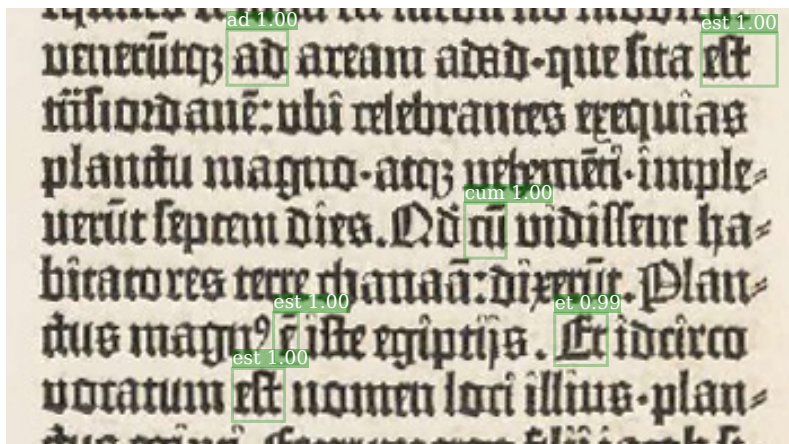


Figure 4.11: Example of network detection.

tection can significantly support floor plan understanding, specifically when it is focused on the semantics of rooms.

The overall processing pipeline is that given floor plan image in input, three files are generated as output. An XML file containing the information about text regions; the input image with annotations about results (regions of text, recognized text, and its type); the input image cleaned up from the detected text. The system is composed of two main modules: text detection and text analysis and classification. Detecting text regions is the first step of the text recognition systems called Optical Character Recognition. This process requires the separation of text region from non-text region.

4.3.1 Text detection module

The text detection module takes an image as input and returns a list of rectangles around detected text regions as output. Four different detection procedures have been used: MSER and SWT, CTPN, EAST, and COMBINED. First implemented starting from one library based on MSER and SWT [73, 2], EAST is based on its OpenCV implementation, and for CTPN we used the library in [1]. COMBINED combines the results of the other three. Starting from an input image, the libraries in their original form calculate a set of points identifying the text areas (EAST and CTPN) or modify the input image to highlight the located text (MSER and SWT). To

be integrated into our project, the libraries have been modified to produce a list of structured data (rectangles identifying text). Moreover, we added functionalities to each detection method to improve the performance of the libraries as we will discuss later. Regardless of the detection procedure used, the module includes a final post-processing phase where rectangles are refined to better locate text within them and facilitate its processing by the analysis module.

MSER and SWT

At the first step of text regions detection in images only image processing techniques with MSER (Maximally Stable Extremal Regions) and SWT (Stroke Width Transform) are used . And also Tesseract-OCR tool is used optionally, as assistance to the algorithm. MSER is an affine feature extraction algorithm proposed by Neumann [71] et al. In MSER algorithm, the image is converted into a gray image firstly, and then, the image is converted into a series of binary images by using the continuous threshold range from 0 to 255. With the increase or decrease of a gray threshold, there is a region of constant occurrence, and the variation of the two thresholds in the region is considered stable in a certain range. Mathematical definitions are as follows: The definition of the image I is the mapping of the region D on the gray. Among them, the scan meets the gray level full sequence structure. The relationship between adjacent pixels is defined as $A \subset D \times D$. The region $Q \subset D$ can be defined as a subset of satisfy connected region criterion, which means for any pixel $p, q \in Q$ there are more than one path to connect p and q. The following formula will illustrate the connected region criterion.

$$pAa_1, a_1Aa_2, \dots, a_nAq \quad (1)$$

where $a_i \in Q, i = 1, 2, \dots, n$. The definition of the boundary aQ is as follows: aQ is adjacent to at least one pixel in the Q , and aQ does not belong to the region Q .

$$\partial Q = \{q \in D - Q, \exists P \in Q, aAp\} \quad (2)$$

For $\forall_p \in Q$ and $\forall_q \in Q$, if $I(p) > I(q)$ then Q is the maximum region. For a series of extremal regions, if change rate $q(i)$ of the region is at the local minimum, it is considered to be the maximally stable extremal regions. $q(i)$ defined as follow:

$$q(i) = \frac{|Q_{i+\Delta} - Q_{i-\Delta}|}{|Q_i|} \quad (3)$$

After obtaining MSER, the connected component analysis is applied to detect the candidate characters. Text detection methods are usually based on connected components (CCs) [104], or sliding windows [46]. Fletcher and Kasturi [28] proposed one of the first methods based on the analysis of connected components and their mutual position. One weakness of this method is that it does not cope with text touching graphics. Tombre et al. [101] proposed an improved approach to separate text touching graphical parts. Connected component methods divide pixels into characters, then group these into words. For example, Epshtein et al. [23] address text detection in natural images considering characters as CCs of the Stroke Width (SWT) transform and then propose an image operator to find the value of the stroke width. The SWT is used to group pixels into letter candidates. Li and Lu [57] adopt a stroke width based text detection approach and propose unique contrast-enhanced Maximally Stable Extremal Region (MSER) to extract character candidates. Gonzalez et al. [23] efficiently combine MSER and locally adaptive thresholding.

The text is made up of strokes, which consist of two parallel edges. Hence, SWT algorithm is based on the fact that texts in the same text line usually have similar stroke width. Therefore, the canny operator is firstly used to detect the edge of the image [49] in the SWT algorithm, and then get the corresponding edge response, finally calculate the distance between the two parallel edges in certain conditions. This method can effectively detect the text line and has a robust fault tolerance ability. It detects stroke pixels by looking for the relative edge pixels q from the beginning of the pixel p along the gradient direction of the dpq , as shown in Figure 4.12. Only when the gradient direction of the edge pixels is opposite to each other, the ray is considered to be effective. All through the ray path pixel has the same stroke width, which is the distance between two parallel edges. Stroke width calculation method is as follows: Assume that the bottom left corner of the edge image is the origin of the coordinate axis, then the coordinates of the edge pixel p is $(x_{p0}, +y_{p0})$, the gradient direction is θ_p . Then from the starting point p along the gradient direction for ray representation:

$$y_p = \tan \theta_p (x_p - x_{p0}) + y_{p0}, \quad x_p > x_{p0} \quad (4)$$

Search along the ray direction until you find the next edge pixel q , con-

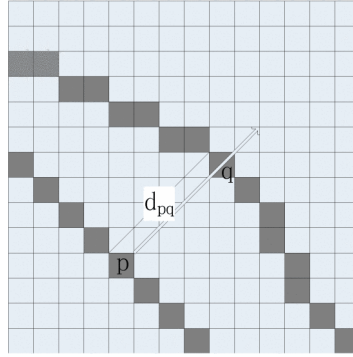


Figure 4.12: Stroke width schematic (image from [54])

sider the relationship between θ_p and gradient direction θ_q of pixel q . If the following constraints are satisfied:

$$\text{abs}(\theta_p - \theta_q) \leq \pi \pm \frac{\pi}{6} \quad (5)$$

Also, if the distance is the minimum value, then d_{pq} is the stroke width. If you do not find the matching pixel q from the p point, or the gradient direction does not meet the criterion's, the ray is discarded. The algorithm compares the stroke width between two adjacent pixels, and if there is a similar value, two pixels will be aggregated. In our research, the original library [2] is used in combination with Tesseract to discard regions containing non-readable text. The library detects single words of text returning a rectangle for each of them. However, in our work near, aligned and semantically correlated words need to be grouped into a unique text object. A post-processing is carried out to merge detection areas by comparing their mutual distance and alignment.

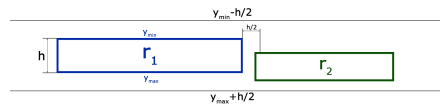


Figure 4.13: Example of rectangles respecting alignment and nearness conditions and therefore merged because they identify the same text object.

All possible pairs of rectangles (r_1, r_2) are considered for merging by checking their alignment (Fig. 4.13). Let h be the height of the taller rectan-

gle (r_1), y_{min} the coordinate of the upper edge of r_1 and y_{max} the position of the lower edge of r_1 . r_1 and r_2 are aligned if r_2 is entirely contained in the interval $[y_{min} - h/2, y_{max} + h/2]$. We allow an offset of $h/2$ at the top and bottom to deal with ascenders and descenders in r_2 . If two rectangles are aligned we then check their horizontal distance: r_1 and r_2 are near if their distance is lower than $h/2$. Pairs of near and aligned rectangles are then merged into the minimum enclosing rectangle defined by (x_{min}, y_{min}) , (x_{max}, y_{max}) that corresponds to the minimum and maximum x and y coordinates occupied by the two rectangles.

EAST (Efficient accurate scene text detector)

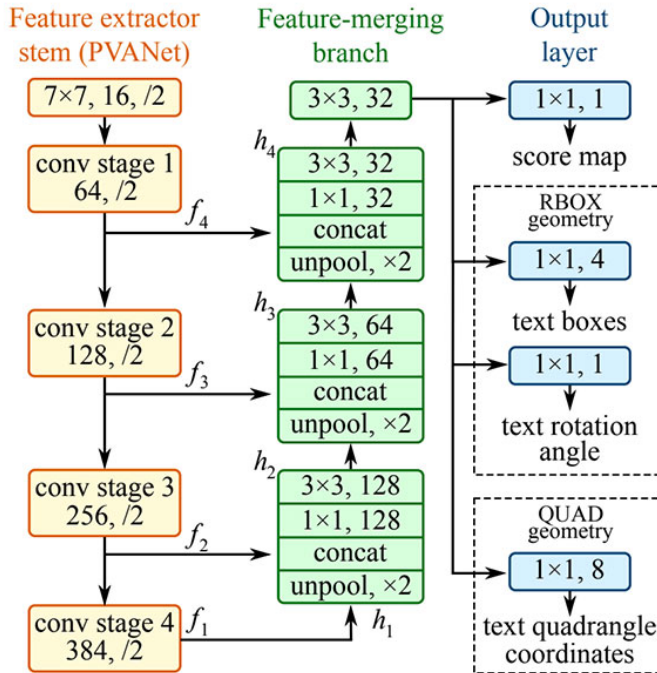


Figure 4.14: The structure of the EAST text detection Fully-Convolutional Network (image from [109])

EAST is a very robust deep learning method for text detection based on

[109]. It is worth mentioning as it is only a text detection method. It can find horizontal and rotated bounding boxes. It can be used in combination with any text recognition method. The text detection pipeline in this paper has excluded redundant and intermediate steps and only has two stages as shown in Figure 4.14. One utilizes the fully convolutional network to directly produce word or text-line level prediction. The produced predictions which could be rotated rectangles or quadrangles are further processed through the non-maximum-suppression step to yield the final output. EAST can detect text both in images and in the video. As mentioned in the paper, it runs near real-time at 13FPS on 720p images with high text detection accuracy. Another benefit of this technique is that its implementation is available in OpenCV 3.4.2 and OpenCV 4. We will be seeing this EAST model in action along with text recognition in our research. On the other hand, EAST accepts as input only images having width and height multiples of 32. We, therefore, add suitable padding of white pixels to those images that do not respect this constraint. EAST detects single words of text like STD. Therefore, also in this case, after the detection, we perform the algorithm for merging near and aligned rectangles described above. The simplicity of this pipeline is to concentrate effort on designing loss function and neural network architecture.

Connectionist Text Proposal Network (CTPN)

A Connectionist Text Proposal Network (CTPN) that directly localizes text sequences in convolutional layers have been proposed in [100]. This overcomes several main limitations raised by previous bottom-up approaches building on character detection. They leverage the advantages of strong deep convolutional features and sharing computation mechanisms, and propose the CTPN architecture which is described in Figure 4.15. It makes the following major contributions:

- First, they cast the problem of text detection into localizing a sequence of fine-scale text proposals. We develop an anchor regression mechanism that jointly predicts the vertical location and text/non-text score of each text proposal, resulting in excellent localization accuracy. This departs from the RPN prediction of a whole object, which is difficult to provide a satisfying localization accuracy.
- Second, they propose an in-network recurrence mechanism that ele-

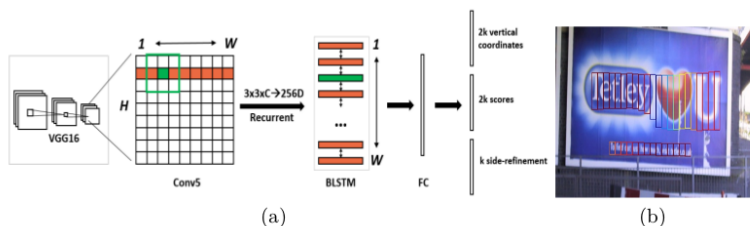


Figure 4.15: (a)Architecture of the Connectionist Text Proposal Network (CTPN), (b)The CTPN outputs sequential fixed-width fine-scale text proposals. (image from [100])

gantly connects sequential text proposals in the convolutional feature maps. This connection allows our detector to explore meaningful context information of text line making it powerful to detect extremely challenging text reliably.

- Third, both methods are integrated seamlessly to meet the nature of text sequence, resulting in a unified end-to-end trainable model. This method can handle multi-scale and multi-lingual text in a single process, avoiding further post filtering or refinement.

On the other hand, CTPN library [1] can only process images with a maximum size of 1200 pixels. For larger images the library automatically scales the dimensions before performing the detection, thus introducing a loss of information. When the image is much larger the loss of information causes a very poor detection. For this reason, in our research large images are first split into sub-images, which are processed separately by CTPN. The original image is split into equal parts both in height and in width so that the dimensions of each sub-image does not exceed the threshold. Unfortunately, the image split gives rise to new problems that need to be addressed. Text objects lying across images will be detected as two separate objects; in other cases, the text area might be wrongly computed by the library, which detects erroneously text too close to the image borders. To address these problems, additional sub-images are generated, centered in the transition areas. After executing CTPN in all the sub-images, we have a list of rectangles returned by CTPN. Some rectangles can overlap because the same text is recognized in overlapped regions. These redundant detections are addressed in the post-

processing. Also, there are rectangles identifying parts of the same text object recognized in different sub-images. These rectangles are addressed as follows. For each sub-image (with size (h, w)) its vertical transition areas are defined as the two vertical rectangles along the vertical border of the sub-image with the width equal to $w/4$. The horizontal transition areas are identified in a similar way considering the horizontal border and height equal to $h/10$. The algorithm merges aligned rectangles lying on adjacent transition areas (that share an edge but belong to different sub-images). The definition of aligned rectangles and the features of the merged ones are the same used when merging side by side rectangles in the STD detection method. As we mentioned above, this analysis of the transition areas is needed because CTPN struggles to accurately identify text very close to the image borders.

Combined Detection Methods

After applying all three methods and obtaining three distinct lists of detected rectangles the combined method performs a voting process to discard false positives. Each rectangle r of a list is compared with all the rectangles of the other two lists. If there is almost one rectangle coinciding with r , then r is added to the final detection list, otherwise, it is discarded. Furthermore, each time two rectangles are found to be coincident, they are merged to refine the detection area. In this case, two rectangles are coincident if their IoU (Intersection over Union) is greater than 50%. The post-processing performs two steps to refine the detection areas. In the first step, the rectangle list is cleaned to eliminate redundant detections (mostly generated by the analysis of intersection areas between different sub-images in CTPN). For each pair of rectangles, we calculate their IoU: if it is greater than 40% they are merged. Otherwise, we check if one is contained for 70% in the other, and in this case, the smaller rectangle is discarded as the two rectangles are detecting the same object. In the second step aligned and overlapped rectangles are merged. These are generated by text objects detected as two split objects for several reasons: splitting into sub-images, presence of noise or other objects superimposed to the text, or to the incorrect evaluation by the library. The concept of alignment is the same presented in MSER/SWT and EAST detection methods.

4.3.2 Text analysis and classification module

This module gets the floor plan image and the rectangles calculated by the detection module and computes additional information for each text object: the text type; the recognized text; a list of rectangles for all the characters in the area. The module also returns a floor plan with the text erased.

Detection of areas belonging to buildings

we calculate the image CCs and identify a CC corresponding to the building if the CC is not contained in any other CC and its area is larger than 10% of the whole image. The convex hull is then calculated from each non-discarded CC.

Text recognition

Each text area is cropped and passed to Tesseract for recognition, computing also the position of words and single characters. After recognition, we check the validity of the text discarding empty strings or not containing letters or numbers.

Text classification

Each text box is classified as follows: If the text object does not intersect with a building (the CC's convex hull computed in step 1) it is classified as generic text. Otherwise, we examine the recognized text: if there are more letters than numbers then it is classified as room description, otherwise, it is classified as room size choosing room size measured in m^2 or room size measured in ft^2 according to the suffix of the recognized text (e.g. a room size measured in ft^2 ends with single apex, double apex or "ft").

Text correction and detection area refinement

the text recognition may not be perfect (e.g. because of noise in the area or poor image quality). We correct the text for the objects classified as room description using some heuristics. Each word is compared with a dictionary of valid words and then replaced with the closest according to the edit distance. In the case of words with the same edit distance, the word with the highest frequency is chosen. Text corresponding to room size is corrected by removing from the string non-valid characters (digits, punctuation marks,

letters, and symbols for units of measure). Since we are not able to make assumptions about the correctness of their content, no correction is made on generic text objects. The area of each text object is refined considering the rectangles of single characters. In the last step, the rectangles surrounding each character are filled with white.

4.3.3 Data sets

We first present the data sets used in the experiments, explaining their features and peculiarities. Then, the detection and classification results are shown separately. Finally, a qualitative analysis of the results is carried out. For the experiments, we used two data sets (Table 4.5) CVC-FP and Flo2Plan, respectively. CVC-FP [14] contains 90 high-quality floor plan images with a uniform style and a well-readable text. Flo2Plan is a subset of the data set proposed in [110] and contains 64 floor plans downloaded from the Internet with very different styles and resolutions. Both data sets have been manually labeled to provide information about text objects.

Dataset	Images	Text Objects	Description	Size (m^2)	Size (ft^2)	Generic Text
CVC-FP	90	2779	1261	925	0	593
Flo2Plan	64	1632	793	0	376	453

Table 4.5: Images and number of labeled text objects in each dataset.

4.3.4 Experimental results

In these tests, we compared the performance of the proposed methods for text detection and text classification. We ignored the accuracy of the read text because it mainly depends on Tesseract and the coverage of the dictionary used to a lesser extent. To evaluate the detection performance we compute the Precision (P), Recall (R) and F_1 Score indices. To compute these values, the detection of a text object is considered correct if its IoU with the ground truth is larger than 65%. We first performed the detection to the CVC-FP and the Flo2Plan datasets using the STD, EAST and CTPN libraries in their original version, without modifications. The results, together with the average execution time per image (on a Mac OSX PC with a 7th generation Intel i5 CPU without GPU), are shown in Table 4.6. Exploiting a GPU the

Original Library	CVC-FP				Flo2Plan			
	P	R	F_1	Time	P	R	F_1	Time (s)
STD	14.60%	19.78%	16.80%	21.10	34.26%	23.72%	28.03%	39.21
EAST	17.82%	33.58%	23.28%	16.28	33.11%	43.17%	37.48%	2.34
CTPN	25.62%	5.24%	8.70%	4.52	25.85%	22.43%	24.02%	5.19
Modified Library	P	R	F_1	Time	P	R	F_1	Time (s)
STD	58.05%	60.00%	59.01%	35.04	56.34%	32.93%	41.57%	45.61
EAST	58.61%	64.90%	61.59%	31.94	61.06%	60.16%	60.61%	13.26
CTPN	84.85%	81.74%	83.27%	103.25	50.36%	39.10%	44.02%	12.27
COMBINED	86.11%	81.05%	83.50%	138.65	68.13%	37.45%	48.33%	52.01

Table 4.6: Detection performance of the methods on the two datasets.

CTPN and EAST execution times would be considerably reduced. As we can see, the performance is very poor in general. For STD and EAST, this is mostly because the both libraries detect single words instead of grouping neighboring words on the same text line under a single text object like CTPN does. Conversely, CTPN shows significant limits in the analysis of poor quality images (Flo2Plan dataset) and large images (CVC-FP dataset) because of the image scaling and subsequent information loss.

In Table 4.6 we also report P , R , F_1 Score, and average execution time of the four detection methods. The execution time includes the time required for the classification and analysis module. The execution time grows linearly with the number of text objects in the image, therefore it can significantly vary from image to image. Evaluating the results we must first notice that no method can detect vertical or curved text. With the clean and high-resolution images of CVC-FP the CTPN method obtains very good results ($P = 85\%$ $R = 82\%$), significantly better than those obtained with the original library. The higher execution time is due to the sub-images splitting. Its performance, however, drops dramatically if applied to the Flo2Plan dataset. The noise, the compression and the poor resolution of these images strongly affect the analysis by the CTPN library. On the other hand on the Flo2Plan data set the EAST method achieves good results ($P = 59\%$ $R = 65\%$). Furthermore, it maintains low execution times. This is possible because EAST has been developed for real-time detection in video streams. This result also confirms that text detection methods based on CNNs perform a better detection than STD not only for scene images but also for graphical documents.

The combined method shows good performance on both datasets, overcoming the other methods in terms of Precision. The Recall is very low

(37%) on the Flo2Plan dataset; this is an expected result considering the voting system it adopts which increases the number of false negatives on images where the other methods have some difficulties.

4.4 Conclusion

In this chapter the combined use of object detection deep architectures, used for word location, and dynamic programming techniques for automatically generating training data from loosely aligned reference text proposed. The experiments performed on the first pages of the Gutenberg's Bible show promising results even if there is still space for improvements. Besides word recognition, the word alignment obtained with the proposed method can be useful for paleographers to better understand the evolution of graphemes over the years. In the rest of the chapter, we also address text extraction, classification, and recognition in floor plan images and compare traditional approaches and methods based on deep learning. The performance of the original methods is significantly improved thanks to suitable pre and post-processing steps specifically implemented to address this task.

Chapter 5

Floor Plan data sets

Nowadays, the number of available data sets for research purposes has been increased owing to the cooperative work of the community. Although research centers, universities, and technical committees contributed through maintaining, updating and sharing their resources, the lack of enough representative benchmark data sets can be seen for the different scenarios in the analysis of the document. There are only a few labeled collections that tested and compared various approaches in different domains. Hence, the ad hoc systems that fit very well in the existing data sets can be benefited more compared to those that better fill in the high variation of the real world. Therefore, it is required to propose modern, detailed and well-structured annotated data sets that can fill empty spaces in any research domain. Regarding the object detection in floor plans, multiple available databases have been included due to the different sub-areas covered by it. It is possible to create these data sets either through synthetic data generation or through real document annotation. On the one hand, synthetic databases contain data produced by a particular predefined set of parameters to model different degrees of noise, distortion, and degradation compared to real documents. The generation of these types of collections would be much faster than the annotated ones. Therefore, the model must be as similar as possible to the reality to provide strong conclusions when using them. On the other hand, the real variability of the world is reflected by the an-

notated databases of real documents. However, image calculation and manually ground-truthing of it can be very time-consuming. This problem can be solved by modifying the annotation processes.

5.1 Introduction

In the literature, the analysis of the floor was utilized for various reasons. Heras et al in [15] have provided CVC-FP. To produce this data set, they have utilized and proposed SGT-tool and found that this type of data is created particularly in a natural approach. Room detection along with wall segmentation was provided. In the document analysis domain, a large set of tools expanded for the generation of GT has been proposed. These data were evaluated by representing the limitations and functionality. Two distinct labeling options namely bounding box and polygonal were presented because SGT is user-friendly. The authors considered the output as the standard Scalable Vector Graphics (SVG).

In [12], in architectural plans, a perceptual model has been proposed to demonstrate the drawing style without interpreting the building parameters, for example, symbols of the drawing. It can facilitate the search for similar plans in the case of perceptual similarity in a database. The method provided in that work is appearance-based. Therefore, the authors do not identify the parameters of the building, however, correlate conceptual semantic concepts with features obtained from raw pictures. In [12], the authors focused on two topics namely lines and spaces. Lines in a broader sense provide information about the walls, texture, and width, and also length. Space provides information about the layout, the size and the shape of rooms. Lines and spaces provide a concept of the building structure, for instance, big squared rooms along with thick external walls can be considered as a query in terms of lines and spaces, perceptually. The structure may be associated with the building function, which means a public theater can have bungalow small ones, thin lines defining the walls and big spaces. In this work, a very simple image signature model was suggested, but descriptive sufficient to show semantic topics about some properties of line and space. Also, a run-length histogram descriptor was proposed. The intuitive concept is introduced as thick lines showing orthogonal walls provide a high frequency for the codeword of long runs in the horizontal and vertical direction. Also, it generates another high frequency in the codeword of run lengths relating to the width of the walls.

Therefore, large rooms or spaces can be shown by histogram maxima in the run lengths relating to their size. The suggested retrieval model is a query with an example framework. The query architectural image, similar images (in terms of the line and space topics) were retrieved.

This pattern was examined to retrieve like sketch drawings of building facades. In the paper, qualitative outcomes on arbitrarily downloaded pictures from the Internet were presented. Our signature can explain in the same approach for buildings of a similar architectural style and also the drawing model.

In [11], a method has been proposed for automatic segmentation and detection of rooms for any kind of floor plan. This method is a combination of two distinct stages. Firstly, in the floor plans, scholars utilized an enhanced version of the statistical patch-based segmentation in terms of walls to graphical entities segmentation. This approach runs at the pixel level and also can facilitate dealing with any kind of graphical notation (refer to [10]). A process of off-line learning is needed to adapt the system to a novel notion. In the second step, a structural approach runs from the record (with grouping the graphical entities), achieved as an outcome of the segmentation, in rooms, independently. Figure 5.1 shows the method pipeline. In the first step, vectorization of the areas of the picture including pixels segmented and also outcomes in extracting wall entities. In the second step, for searching the final window and door entities on a vectorized image graph, the segmentation of window and door along with the structural context of walls were combined by utilizing the algorithm of A*. Rooms were specified as the cycles in the entity plane graph of windows, doors, and walls utilizing the method suggested in [47], eventually.

The whole system was carried out on novel released data sets of real architectural floor plans using different graphical notations. To analyze the system, two evaluation protocols were utilized. One is utilized to the rooms' recognition and another one is used to wall segmentation. By pixel-based approach, the picture pixels belonging to windows and walls and also doors were labeled and segmented. Figure 5.2 shows the pipeline of structural-based recognition. The fundamental graphical segmentation was gathered into three sorts of structural entities including windows, walls, and doors. The rooms were discovered by detecting cycles in a graph of entities. It is important to note that these approaches have capable to extract the structural parameters without prior knowledge of the graphical modeling convention

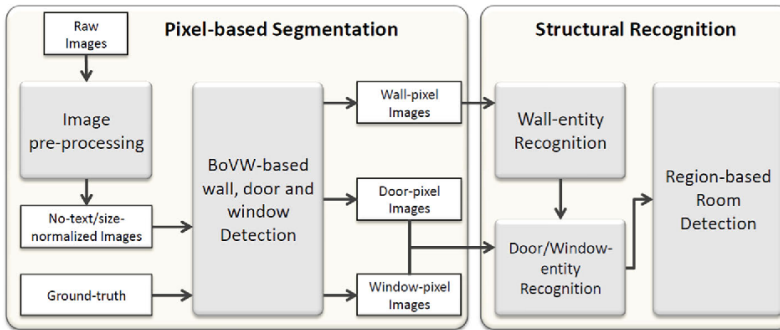


Figure 5.1: Pipeline of the method(image from [10])

in terms of the floor plans. However, they require some corpus of ground-truthed documents for learning each novel notation.

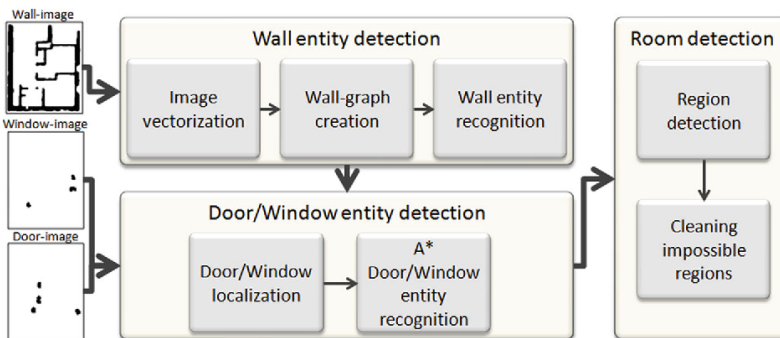


Figure 5.2: Structural recognition pipeline (image from [10])

An unsupervised wall segmentation method is proposed by Heras et al [12] that is based on combining two approaches. First, walls are independently detected to their notation and structure via this method and no labeled data are required for learning the graphical appearance of the walls. Then, their appearance is learned according to a revised version of [13] and the initial segmentation is refined. After that, some Internet images and four available data sets that were presented in [10] are used for testing the proposed method. In addition, the latest strategies that reported their results on these collections are compared to this proposed method. Sébastien, et al in [65] performed a study on wall and room detection. They proposed a two-step

method to interpret the architectural floor plans. The presented method is designed particularly to recognize the rooms. This method includes the extraction of basic components in the image such as the arcs and the lines that compose the doors and the walls respectively. It also contains the detection of rooms that form the building. This algorithm which is proposed for line detection is based on the coupling of Hough Transform (HT) with image vectorization (IV) with a fixed threshold for the minimum length of the concatenation of all the lines that are on the same direction. An automatic lookup tool is presented by Sharma et al. in [90] to match and retrieve similar floorplans from a large repository of digitized architectural floorplans. They performed a graph-based approach that incorporates semantics in terms of the furniture arrangement and the room layout. They also analyzed the floor plans for distinguishing between layouts with more specificity. Also, a graph spectral embedding approach is adopted by them to present the graphs obtained for room layouts as a three-component vector. They implemented an algorithm for calculation of the semantic difference between layouts.

In their study, Ahmad et al. [3] represented a complete system that contains a corpus of just 80-floor plans. They used this system for automatic room recognition and room labeling from architectural floor plans. In this system, structural and semantic analysis steps are applied for retrieving room information. Also, the room labels are extracted by this system to identify the functions of the rooms.

5.2 Floor Plan data sets

Although there are many data sets of floor plan images, there is still a significant need for having a real data set to cover different types of floorplans in the world. One or more labeled data sets are required to evaluate the object detection in floor plans. There are some data sets related to floorplan for analyzing the research in this task. For example, The CVC-FP collection comprises 122 scanned floor plan documents which are divided into four different subsets concerning their origin and style. It contains documents of different qualities, resolutions, and modeling styles, which is suitable for testing the robustness of the analysis techniques. The data set is fully ground-truthed for the structural symbols: rooms, walls, doors, windows, parking doors, and room separations. Two of these collections are ground-truthed at the pixel level for the structural elements wall, door, window, and room, and they are

used to evaluate both wall and room detection. A generic method is also presented in [10] can analyze any type of floorplan without considering the used notation. These floor plans contain only a few furniture objects and therefore are not suitable for the research conducted in my work. To obtain a suitable result in this work, compared to the related studies in the literature, objects (e.g., toilet, bathtub) without labels were considered. The ground truth then has been manually created by humans. One of the main reasons to use CVC-FP data set in my research is that the type of CVC-FP data set is real and provides similarity features for comparing the results. Two tests were administered on these new labeled data sets. There is another data set called SESYD that comes from [18] which is a synthetic graphics documents and contains non-isolated symbols in a real context (examples are shown in Figure 5.3). These documents are as follows:

- Drawings which include electrical diagrams and architectural floorplans
- Bags of symbols which include arbitrary compositions of segmented symbols
- Query symbols including cropped images of symbols

The whole data set contains five document collections involving around 11,100 images representing 128,700 symbols. When furniture objects are randomly placed on a few fixed floor plan layouts a synthetic floor is designed. Although this data set generation approach is very appealing, the current data set includes only ten-floor plan layouts and the objects are derived from a limited number of categories.

There are 500-floor plan images in the Rakuten-FP data set (with jpg format) and the pixel-level annotations are divided into the wall and non-wall pixels in png format as well as 6 different object classes including doors, sliding doors, kitchen stoves, bathtubs, sinks, and toilets. One example of this data is shown in Figure 5.4 For training and testing purposes, a subset of the R-FP images was annotated and is used in [22]. In the aforementioned paper, the authors parse floor plan images for the estimation of the rooms' size for interactive furniture fitting. First of all, they performed wall segmentation via a fully convolutional neural network. Then, they detected objects with a Faster R-CNN. Finally, to obtain the rooms' dimensions, they used optical character recognition. In [21], a combination of three methods is used to achieve more understandings about the overall floor plan. However,



Figure 5.3: (a) two instances of document (b) reproduction of domain rules (c) bags of symbols (image from [18])

there are limited details on the use of Faster R-CNN for object location in this study. Furthermore, the proposed floor plan data set only includes the ground-truth about the wall position.

The ROBIN data set includes three broad categories of layouts. One of the significant features of ROBIN is that the design of this data set emphasizes the need for a potential buyer. The three broad categories included in this data set differ from the number and type of rooms represented in a floor plan. According to the global layout shape of the floor plan, every broad category is then classified into 10 sub-categories. ROBIN can be used for retrieving the image as well as for other floor plan analysis tasks. ROBIN is useful for visualizing the floor plans and helps to efficiently capture of various high-level features while fine-grained retrieval. Although the proposed approach is appealing for the needs of buyers, the actual data set includes only 51-floor plan layouts and the objects are derived from a limited number of categories. One example of this ROBIN data set is shown in Figure 5.5

There is another large-scale data set called “LIFULL” which has ground-truth for vector- graphics floorplan conversion. It is based on the LIFULL

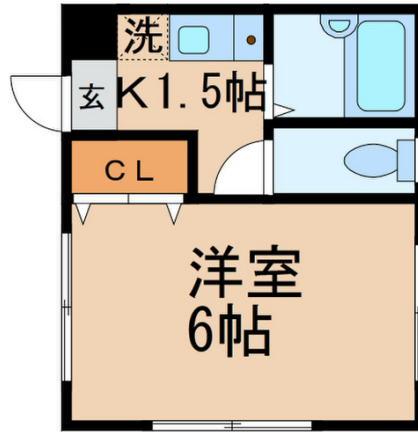


Figure 5.4: One example of Rakuten-FP data set

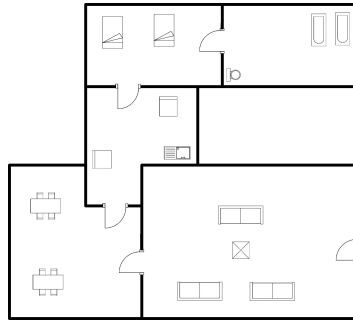


Figure 5.5: Example of a sample floor plan image from ROBIN data set

HOME'S data set containing 5 million floorplan raster images. 1000 floorplan images are randomly sampled to create the ground-truth when for each floor plan image, the annotation of the geometric and semantic information performed manually. Annotators can draw a line representing either a wall or an opening. They can also draw either a rectangle and choose an object type for each object or attach a room label at a specific location. Since LIFUL data set is really large, I used this data to compare some statistics about the distribution of objects. These statistics are briefly summarized in Figures 5.6 and 5.7.

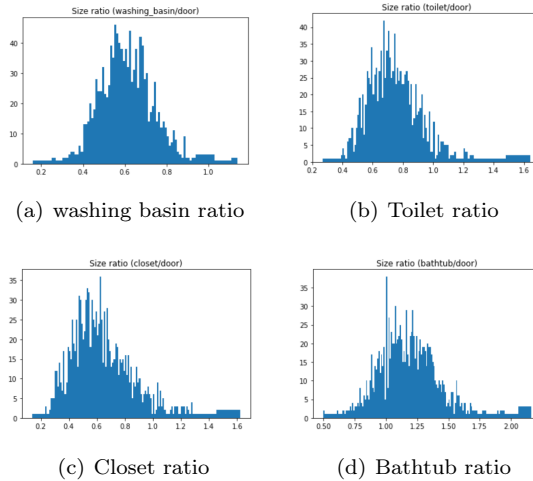


Figure 5.6: In terms of the size of each object regarding the door size, it can be seen a similar pattern in toilet, closet, and washing-basin, which shows that the most number of objects have almost the same ratio ranges from 0.5 to 0.7.

5.3 Data sets Creation

In this section, we briefly describe two new data sets that have been gathered and annotated to support the approach described in this thesis.

5.3.1 ISTA

Middle Eastern floor plans are included in this data set, and more diversity is seen among object shapes which differ from other data set that they emerged from Google's image search. ISTA is known as having some significant features, for example, since the floor plans are derived from a single architectural firm and real floor plans are contained in ISTA their contents are more homogeneous and all the objects are identified in the real-mode as it is shown in Figure 5.8. Also, the ratio of the dimensions of the object to each other is real and standard as these floor plans are considered in the same firm, a similar sample is used for a particular object among different kinds of samples (e.g., bed, table).

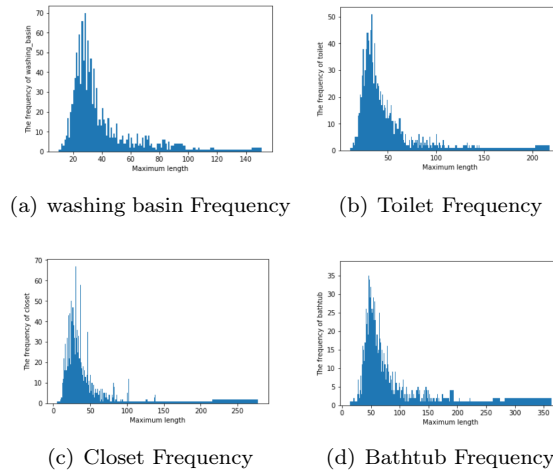


Figure 5.7: The graphs show how many objects are in the same size. According to the chart, frequency levels for most of the objects such as toilet, bathtub, and closet have risen during the maximum length between 20 to 60 whilst for washing-basin the most frequent length is between 20 and 40

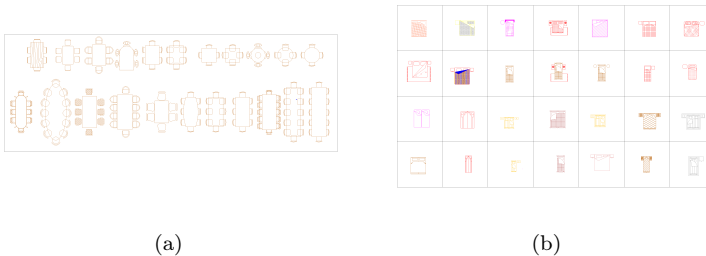


Figure 5.8: ISTA: (a) Different kind of tables, (b) Different kind of beds

Although ISTA includes 300 images, only 160 images have been labeled so far in my research. The labeled images are divided into 12 classes with 7788 objects. One of the distinct characteristics of these floor plan data sets is that the images have simple shapes and most of them are grayscale. It is empirically proven that this feature affects the model performance positively compared to those data sets with images that contain more noise and more

complex properties. In the first step, Otsu method is used to binarize Floor plan images. As a result, they will be able to extract runs. Then, the text-graphic separation method is implemented to filter out components of the text because they do not appear in all the images and their global perception can be biased slightly. Images are also resized and the heights of the images are fixed to 1200 pixels. On the other hand, the dynamic width re-scaling is performed to keep the aspect ratio. As a result, the same image proportions will be preserved as the ones used for generating the ground truth, in which a fixed screen resolution of 1900×1200 is used to show the images to the observers. An example of this process is shown in Figure 5.9.

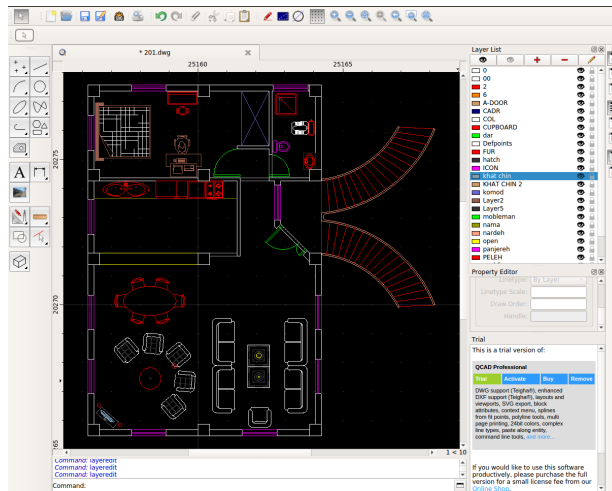
5.3.2 FLo2Plan

As the size of the considered data set is increased, the performance of the training model is increased. Therefore, considering the limited size of *d1* data set, a larger data set with the same features, called Flo2Plan, is created. The Flo2Plan includes 452 floor plan images from different countries (e.g., Italy, USA, France, Germany), including colorful and grayscale ones, which can be different with respect to their size and type (e.g., office, house, hotel). Moreover, since it has been downloaded using Google search, it can be considered as a highly noisy data set, in particular, considering blurry or watermarked images. Also, the object shapes in Flo2Plan do not follow a fixed design rule. For example, in an image, a table can be a $100 * 100$ (cm) rectangular, while another one is a $30 * 30$ (cm) oval. Also, Flo2Plan includes 11517 symbol instances and 13 object categories including, door, sink, toilet, bathtub, shower, bidet, table, chair, couch, armchair, hotplate, night table, bed.

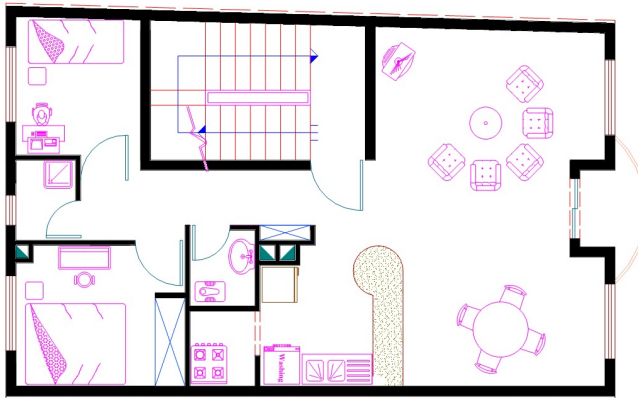
5.4 Annotation tools

5.4.1 Labelimg

Recently, a graphical image annotation tool known as LabelImg is proposed that can label object bounding boxes in images. It is considered as an open-source image labeling tool with pre-built binaries for Windows. Therefore, its installation is easy. However, bounding boxes are only supported by this tool. There are also two other versions that one of them is in the



(a) Original Image



(b) Preprocessed Image

Figure 5.9: (a)First version of images, (b)Preprocessed Image in ISTA

RotatedRect format and the other is an optimized version used for one-class tagging. PascalVoc XML format is used. For every image included in the source file, annotation files are saved separately. Although it contains no particular project management features, it provides a simple way for

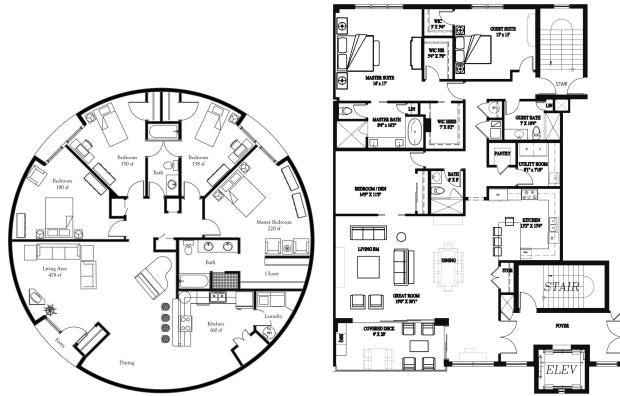


Figure 5.10: Two examples of Flo2plan

importing and visualizing annotations and it corrects them. The annotation process is fastened through a simple offline interface, however, it can support a few hotkey shortcuts. In Figure 5.11 is shown an example of this labeling method in floor plan data set.



Figure 5.11: labeling with Labeling method

5.4.2 Labelme

Using LabelMe annotation tool which proposed in [103], users will be able to contribute to the project. It is possible to access this tool either anonymously or through logging into a free account. If the users tend to access this tool, they are required to have a compatible web browser that supports JavaScript. After being loaded, a random image from the LabelMe data set is selected displayed on the screen via this tool. The object levels will be overlaid on top of the image in polygon format. If there are associated object labels in the image, they will be overlaid on top of the image in polygon format. Each distinct object label is represented with a different color. The user will be able to use the mouse and draw a polygon including an object in the image if the image is not completely labeled. As an example, if there is a person in the adjacent image who is standing in front of the building, the user will be able to click on a point on the border of the person and click along the outside edge until returning to the starting point. A bubble pops up on the screen after closing the polygon that permits the user entrance into a label for the object. The user can select all the labels that believe the best describe the object. When the users do not agree with the previous labeling of the image, they can click on the outline polygon of an object. They can also delete the polygon completely or edit the text label and give a new name to it. When the users change the image, they are saved and anyone can download them from the Labelme data set. Therefore, due to the contribution of the users who implement the tool, the data is always changing. When the users finish working on an image, they can click on the Show me another image link. Then, another random image will be chosen to show to the user.

Modified Labelme

It is required to label a floor plan image in which the background is usually fixed. However, the objects are often small and close to each other. To solve these problems, an open-source implementation is extended by my colleague Samuele Capobianco to add some useful properties for Flo2plan annotation to allow defining its object classes and properties, considering the real scenarios. Therefore, several features are added to label instances with rectangles that define a bounding box area. In this case, object annotation is allowed with two mouse clicks defining the left-top and right-bottom coordinates. Another feature is added to reduce wasting of time and to start labeling

phase that starts from the computed output. As a result, a floor plan image is annotated which is started from the output of an existing trained model and the labeling time is reduced.

Given the aim of this study, the background is of great importance. Therefore, the object contour detection embedded in a larger bounding box area will be possible. In the first step, edge detection through a difference of Gaussian (DoG) is performed at different scales. In the next step, the major key points are only selected via a dense fixed grid. After that, the instance mask is computed as the convex hull on the selected points given the best key points inside the bounding box which have already been selected. As a result, the mask creation around each object can be possible. For better detection of object contour, it can correct the computed mask adding or removing key points if necessary. The computed key points are indicated in Figure 5.12 via the difference of the gaussian (DoG) technique. They are correspondent to the most probable edges which are beneficial for contour definition. DoG edges are computed to select the best key points (red points) which can object contour definition. In Figure 5.13 it is indicated that how can edit the computed contour after the movement of the selected area. In the first window, the selected area can be seen from the input image. Then the opportunity of adding or removing points in the second window is possible and the obtained contour is checked via convex hull in the third window. Inside of the second window can be seen that how beneficial is to Delaunay tessellation to join selected key-points during the editing phase. Another tool is added to the system to group defined instances in different super categories. Also, a software is implemented to define a rectangle area to surround an object annotating a super-category area that defines the relationship among instances to represent more structural information.

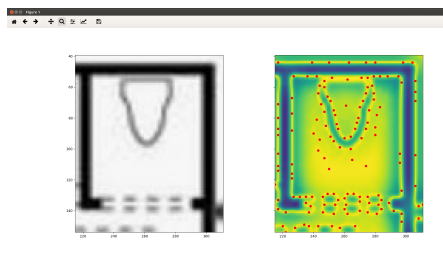


Figure 5.12: DoG edges to select the best key points

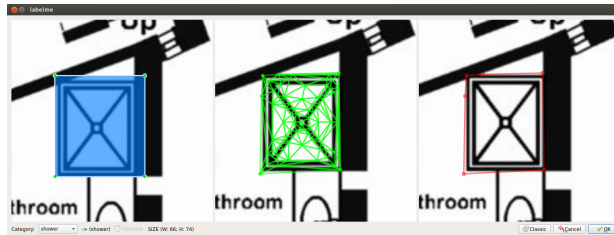


Figure 5.13

5.5 Conclusion

Based on the previous study performed on floor plan data set creation, we find that it is required to have a more comprehensive data set with some more features in the field of floor plan analysis. In this chapter, two novel floor plan image data sets are proposed including ISTA and Flo2plan. In comparison with the other existing annotated floor plan data sets, ISTA is more realistic and more varied in its annotations which cover over 12-floor plan object categories. On the other hand, Flo2plan is a comprehensive and uniform data set and floor plans are highly noisy and non-homogeneous. A modified labeling method is provided together with the novel data sets, which is extended to an open-source implementation to add some useful properties for Flo2plan annotation to allow defining its object classes and properties.

Chapter 6

Conclusion

This chapter summarizes the contribution of the thesis and discusses avenues for future research.

6.1 Summary of contribution

The main contributions of this thesis can be summarized as follows:

Object detection in floor plan images We evaluated the performance of an object detector which is originally designed for detecting objects in natural images applied to identify objects in the floor plans in two different data sets with different features. From the experiments performed on these data sets, we can notice that by using convolutional object detectors, the recognition performance is not highly influenced by a class imbalance in the training set while the performance of the model is influenced by data sets nature. We also noticed that the performance of the model in a class heavily depends on the diversity of object samples; object rotation and scale somehow outweighing the role of sample size. It is interesting also to notice how a network pre-trained from another domain (COCO pre-trained Faster-RCNN with Res Net 50) can perform well on the floor plan data sets using just a one hundred training images.

Text recognition in floor plan images we present a method for text recognition in floor plan images. In particular, we are concerned about locating, reading, and categorizing text inside floor plan images to obtain information about the building. To do so we compare traditional text detection methods, based on image processing techniques, with recent approaches rely-

ing on convolutional neural networks. Two data sets with different features, including quality and size, were considered in the experiments performed. The performance of the original methods is significantly improved thanks to suitable pre and post-processing steps specifically implemented to address this task.

Text alignment in early printed books We proposed the combined use of object detection deep architectures, used for word location, and dynamic programming techniques for automatically generating training data from loosely aligned reference text. The experiments performed on the first pages of the Gutenberg's Bible show promising results. Besides word recognition, the word alignment obtained with the proposed method can be useful for paleographers to better understand the evolution of graphemes over the years.

Floor Plan Data sets creation and labeling We propose two novel floor plan image data sets: ISTA and Flo2plan. In comparison with other existing annotated floor plan data sets, ISTA is more realistic and more varied in its annotations which cover over 12-floor plan object categories. On the other hand, Flo2plan is a comprehensive and uniform data set and floor plans are in this case highly noisy and non-homogeneous. A modified labeling method is provided, which is extended to an open-source implementation with the aim of adding some useful properties to Flo2plan annotation to allow defining its own object classes and properties.

6.2 Directions for future work

In spite of rapid development and achieved promising progress in document image analysis, there are still many open issues for future work.

The planned future work on understanding floor plans includes the integration of text detection and object location within a single architecture and the use of textual information to improve results on object detection, in particular, if we find the useful textual information in the floor plan, it could help us to improve the results in object detection. For instance, considering the room's name to limit the object that could be detected (e.g. a WC cannot contain a bed). We can also perform experiments and tune the aforementioned model on other data sets such as "Rakuten" or "LIFULL".

From the data sets point of view, even if some modifications have been done to make easier the process of labeling, there is still need to have more

automatic labeling.

Another future work includes the use of the proposed alignment system to automatically obtain a larger corpus starting from more pages and then train the recognizer with more classes or even use LSTM-based word recognition. It would be interesting to study a way to combine relational networks, convolutional networks, and LSTMs to construct an end to end neural network that can learn relations between words.

Appendix A

Publications

This research activity has led to some publications in international conferences and workshops. These are summarized below.

International Journal

1. **Z.Ziran**, X.Pic, S.Undri Innocenti, D.Mugnai, S.Marinai. “Text alignment in early printed books combining deep learning and dynamic programming”, *Pattern Recognition Letters*.

International Conferences and Workshops

1. **Z.Ziran**, S.Marinai. “Object Detection in Floor Plan Images”, in *book: Artificial Neural Networks in Pattern Recognition, ANNPR 2018*, pp. 383-394, Siena (Italy), September 2018 Lecture Notes in Computer Science DOI: 10.1007/978-3-319-99978-430
2. J.Ravagli, **Z.Ziran**, S.Marinai. “Text Recognition and Classification in Floor plan Images”, in *Proceedings Workshops International Conference on Document Analysis and Recognition*, pp. 1-6 , Sydney (Australia)
3. Simone Marinai, Samuele Capobianco, **Zahra Ziran**, Andrea Giuntini and Pierluigi Mansueto. Recognition of Concordances for Indexing in Digital Libraries. Submitted to: *Italian Research Conference on Digital Libraries (IRCDL 2020)*

Bibliography

- [1] “CTPN,” <https://github.com/eragonruan/text-detection-ctpn>.
- [2] “Text detection with MSER and SWT,” <https://github.com/azmiozgen/text-detection>.
- [3] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel, “Automatic room detection and room labeling from architectural floor plans,” in *2012 10th IAPR International Workshop on Document Analysis Systems*, March 2012, pp. 339–343.
- [4] S. Ahmed, M. Weber, M. Liwicki, C. Langenhan, A. Dengel, and F. Petzold, “Automatic analysis and sketch-based retrieval of architectural floor plans,” *Pattern Recognition Letters*, vol. 35, pp. 91–100, 2014.
- [5] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick, “Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks,” *CoRR*, vol. abs/1512.04143, 2015. [Online]. Available: <http://arxiv.org/abs/1512.04143>
- [6] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk, and B. Girod, “Robust text detection in natural images with edge-enhanced maximally stable extremal regions.”
- [7] J. Dai, K. He, and J. Sun, “Instance-aware semantic segmentation via multi-task network cascades,” *CoRR*, vol. abs/1512.04412, 2015. [Online]. Available: <http://arxiv.org/abs/1512.04412>
- [8] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: object detection via region-based fully convolutional networks,” *CoRR*, vol. abs/1605.06409, 2016. [Online]. Available: <http://arxiv.org/abs/1605.06409>
- [9] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05) - Volume 1 - Volume 01*, ser. CVPR ’05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 886–893. [Online]. Available: <https://doi.org/10.1109/CVPR.2005.177>

- [10] L.-P. de las Heras, S. Ahmed, M. Liwicki, E. Valveny, and G. Sánchez, “Statistical segmentation and structural recognition for floor plan interpretation,” *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 17, no. 3, pp. 221–237, Sep 2014. [Online]. Available: <https://doi.org/10.1007/s10032-013-0215-2>
- [11] —, “Statistical segmentation and structural recognition for floor plan interpretation,” *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 17, no. 3, pp. 221–237, Sep 2014. [Online]. Available: <https://doi.org/10.1007/s10032-013-0215-2>
- [12] L.-P. de las Heras, D. Fernández, E. Valveny, J. Lladós, and G. Sánchez, “Unsupervised wall detector in architectural floor plans,” in *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, ser. ICDAR ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1245–1249. [Online]. Available: <https://doi.org/10.1109/ICDAR.2013.252>
- [13] L.-P. de las Heras, J. Mas, G. Sánchez, and E. Valveny, “Notation-invariant patch-based wall detector in architectural floor plans,” in *Graphics Recognition. New Trends and Challenges*, Y.-B. Kwon and J.-M. Ogier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 79–88.
- [14] L. de las Heras, O. R. Terrades, S. R. Mestre, and G. Sánchez, “CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool,” *IJDAR*, vol. 18, no. 1, pp. 15–30, 2015. [Online]. Available: <https://doi.org/10.1007/s10032-014-0236-5>
- [15] L.-P. de las Heras, O. R. Terrades, S. Robles, and G. Sánchez, “Cvc-fp and sgt: a new database for structural floor plan analysis and its groundtruthing tool,” *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 18, no. 1, pp. 15–30, 2015.
- [16] L.-P. de las Heras, E. Valveny, and G. Sánchez, “Unsupervised and notation-independent wall segmentation in floor plans using a combination of statistical and structural strategies,” in *Graphics Recognition. Current Trends and Challenges*, B. Lamiroy and J.-M. Ogier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 109–121.
- [17] M. Delalandre, E. Valveny, T. Pridmore, and D. Karatzas, “Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems,” *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 13, no. 3, pp. 187–207, Sep 2010.
- [18] M. Delalandre, E. Valveny, and J.-Y. Ramel, “Recent contributions on the sesyd dataset for performance evaluation of symbol spotting systems,” 2011.

- [19] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.
- [20] S. Dodge, J. Xu, and B. Stenger, “Parsing floor plan images,” in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, May 2017, pp. 358–361.
- [21] S. Dodge, J. Xu, and B. Stenger, “Parsing floor plan images,” in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, May 2017, pp. 358–361.
- [22] S. Dodge, J. Xu, and B. Stenger, “Parsing floor plan images,” in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, May 2017, pp. 358–361.
- [23] B. Epshtein, E. Ofek, and Y. Wexler, “Detecting text in natural scenes with stroke width transform,” in *CVPR 2010*, 2010, pp. 2963–2970. [Online]. Available: <https://doi.org/10.1109/CVPR.2010.5540041>
- [24] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, “Scalable object detection using deep neural networks,” *CoRR*, vol. abs/1312.2249, 2013. [Online]. Available: <http://arxiv.org/abs/1312.2249>
- [25] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010. [Online]. Available: <https://doi.org/10.1007/s11263-009-0275-4>
- [26] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010. [Online]. Available: <https://doi.org/10.1109/TPAMI.2009.167>
- [27] A. Fischer, V. Frinken, A. Fornés, and H. Bunke, “Transcription alignment of latin manuscripts using hidden markov models,” in *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, ser. HIP ’11. New York, NY, USA: ACM, 2011, pp. 29–36. [Online]. Available: <http://doi.acm.org/10.1145/2037342.2037348>
- [28] L. A. Fletcher and R. Kasturi, “A robust algorithm for text string separation from mixed text/graphics images,” *IEEE TPAMI*, vol. 10, no. 6, pp. 910–918, Nov 1988.
- [29] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, “DSSD : Deconvolutional single shot detector,” *CoRR*, vol. abs/1701.06659, 2017. [Online]. Available: <http://arxiv.org/abs/1701.06659>

- [30] L. Gao, X. Yi, Y. Liao, Z. Jiang, Z. Yan, and Z. Tang, “A deep learning-based formula detection method for pdf documents,” in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, Nov 2017, pp. 553–558.
- [31] B. Gatos, D. Danatsas, I. Pratikakis, and S. J. Perantonis, “Automatic table detection in document images,” in *ICAPR*, 2005.
- [32] A. Gilani, S. R. Qasim, M. I. Malik, and F. Shafait, “Table detection using deep learning,” *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, pp. 771–776, 2017.
- [33] R. B. Girshick, “Fast R-CNN,” *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: <http://arxiv.org/abs/1504.08083>
- [34] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: <http://arxiv.org/abs/1311.2524>
- [35] C. Goncu, A. Madugalla, S. Marinai, and K. Marriott, “Accessible on-line floor plans,” in *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, 2015, pp. 388–398. [Online]. Available: <http://doi.acm.org/10.1145/2736277.2741660>
- [36] G. Gupta, Swati, M. Sharma, and L. Vig, “Information extraction from hand-marked industrial inspection sheets,” in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, 2017, pp. 33–38.
- [37] L. Hao, L. Gao, X. Yi, and Z. Tang, “A table detection method for pdf documents based on convolutional neural networks,” *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pp. 287–292, 2016.
- [38] T. Hassner, L. Wolf, and N. Dershowitz, “Ocr-free transcript alignment.”
- [39] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06870>
- [40] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *CoRR*, vol. abs/1406.4729, 2014. [Online]. Available: <http://arxiv.org/abs/1406.4729>
- [41] —, “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.
- [42] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming auto-encoders,” in *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, ser. ICANN’11. Berlin, Heidelberg:

- Springer-Verlag, 2011, pp. 44–51. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2029556.2029562>
- [43] H. N. Ho, C. Rigaud, J.-C. Burie, and J.-M. Ogier, “Redundant structure detection in attributed adjacency graphs for character detection in comics books,” in *10th IAPR International Workshop on Graphics Recognition*, United States, Aug. 2013. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-00937632>
- [44] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 3296–3297.
- [45] —, “Tensorflow object detection api,” github.com/tensorflow/models/tree/master/research/object_detection, 2018.
- [46] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Deep features for text spotting,” in *ECCV*, 2014, pp. 512–528.
- [47] X. Jiang and H. Bunke, “An optimal algorithm for extracting the regions of a plane graph.” *Pattern Recognition Letters*, vol. 14, no. 7, pp. 553–558, 1993. [Online]. Available: <http://dblp.uni-trier.de/db/journals/prl/prl14.html#JiangB93>
- [48] F. D. Julca-Aguilar and N. S. T. Hirata, “Symbol detection in online handwritten graphics using faster R-CNN,” in *Proceedings of the 13th International Workshop on Document Analysis Systems*, 2018, pp. 151–156.
- [49] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazàn, and L. P. de las Heras, “Icdar 2013 robust reading competition,” in *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, ser. ICDAR ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1484–1493. [Online]. Available: <https://doi.org/10.1109/ICDAR.2013.221>
- [50] F. S. Khan, R. M. Anwer, J. van de Weijer, A. D. Bagdanov, M. Vanrell, and A. M. Lopez, “Color attributes for object detection,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 3306–3313.
- [51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, may 2017. [Online]. Available: <https://doi.org/10.1145%2F3065386>
- [52] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *2006 IEEE*

- Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, June 2006, pp. 2169–2178.
- [53] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature Cell Biology*, vol. 521, no. 7553, pp. 436–444, 5 2015.
- [54] Leibin Guan and Jizheng Chu, “Natural scene text detection based on swt, msr and candidate classification,” in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, June 2017, pp. 26–30.
- [55] K. Lenc and A. Vedaldi, “R-CNN minus R,” *CoRR*, vol. abs/1506.06981, 2015. [Online]. Available: <http://arxiv.org/abs/1506.06981>
- [56] Y. Leydier, V. Églin, S. Brès, and D. Stutzmann, “Learning-free text-image alignment for medieval manuscripts,” in *2014 14th International Conference on Frontiers in Handwriting Recognition*, Sep. 2014, pp. 363–368.
- [57] Y. Li and H. Lu, “Scene text detection via stroke width,” in *Proc. ICPR*, 2012, pp. 681–684. [Online]. Available: <http://ieeexplore.ieee.org/document/6460226/>
- [58] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: <http://arxiv.org/abs/1612.03144>
- [59] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: <http://arxiv.org/abs/1405.0312>
- [60] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, 2014, pp. 740–755.
- [61] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” 2016, to appear. [Online]. Available: <http://arxiv.org/abs/1512.02325>
- [62] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, “SSD: single shot multibox detector,” *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: <http://arxiv.org/abs/1512.02325>
- [63] S. Macé, H. Locteau, E. Valveny, and S. Tabbone, “A system to detect rooms in architectural floor plan images,” in *The Ninth IAPR International Workshop on Document Analysis Systems, DAS 2010, June 9-11, 2010, Boston, Massachusetts, USA*, 2010, pp. 167–174. [Online]. Available: <https://doi.org/10.1145/1815330.1815352>

- [64] S. Macé, H. Locteau, E. Valveny, and S. Tabbone, “A system to detect rooms in architectural floor plan images,” in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, ser. DAS '10. New York, NY, USA: ACM, 2010, pp. 167–174. [Online]. Available: <http://doi.acm.org/10.1145/1815330.1815352>
- [65] —, “A system to detect rooms in architectural floor plan images,” in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, ser. DAS '10. New York, NY, USA: ACM, 2010, pp. 167–174. [Online]. Available: <http://doi.acm.org/10.1145/1815330.1815352>
- [66] A. Madugalla, K. Marriott, and S. Marinai, “Partitioning open plan areas in floor plans,” in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, 2017, pp. 47–52. [Online]. Available: <https://doi.org/10.1109/ICDAR.2017.17>
- [67] —, “Partitioning open plan areas in floor plans,” in *ICDAR*, 2017, pp. 47–52.
- [68] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image Vision Comput.*, vol. 22, no. 10, pp. 761–767, 2004. [Online]. Available: <https://doi.org/10.1016/j.imavis.2004.02.006>
- [69] Y. Nagaoka, T. Miyazaki, Y. Sugaya, and S. Omachi, “Text detection by faster r-cnn with multiple region proposal networks,” in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 06, Nov 2017, pp. 15–20.
- [70] M. Najibi, M. Rastegari, and L. S. Davis, “G-CNN: an iterative grid based object detector,” *CoRR*, vol. abs/1512.07729, 2015. [Online]. Available: <http://arxiv.org/abs/1512.07729>
- [71] L. Neumann and J. Matas, “Real-time scene text localization and recognition,” *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3538–3545, 2012.
- [72] N. Nguyen, C. Rigaud, and J. Burie, “Comic characters detection using deep learning,” in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 03, Nov 2017, pp. 41–46.
- [73] A. C. Ozgen, M. Fasounaki, and H. K. Ekenel, “Text detection in natural and computer-generated images,” in *26th Signal Processing and Communications Applications Conference*, 2018, pp. 1–4.
- [74] A. Pacha and H. Eidenberger, “Towards a universal music symbol classifier,” in *Proceedings of the 12th International Workshop on Graphics Recognition*, 2017, talk: International Workshop on Graphics Recognition, New York;

- 2017-11-09 – 2017-11-10. [Online]. Available: http://publik.tuwien.ac.at/files/publik_266238.pdf
- [75] A. Pacha, H. Eidenberger, B. C. Kwon-Young Choi, Y. Ricquebourg, and R. Zanibbi, “Handwritten music object detection: Open issues and baseline results,” in *Proceedings of the 13th International Workshop on Document Analysis Systems*, 2018, pp. 163–168.
- [76] F. Perronnin, J. Sánchez, and T. Mensink, “Improving the fisher kernel for large-scale image classification,” in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ser. ECCV’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 143–156. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1888089.1888101>
- [77] P. H. O. Pinheiro, R. Collobert, and P. Dollár, “Learning to segment object candidates,” *CoRR*, vol. abs/1506.06204, 2015. [Online]. Available: <http://arxiv.org/abs/1506.06204>
- [78] X. Qin, Y. Zhou, Z. He, Y. Wang, and Z. Tang, “A faster R-CNN based method for comic characters face detection,” in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, 2017, pp. 1074–1080.
- [79] I. Rabaev, R. Cohen, J. El-Sana, and K. Kedem, “Aligning transcript of historical documents using dynamic programming,” in *Document Recognition and Retrieval XXII, San Francisco, California, USA, February 11-12, 2015.*, 2015, p. 940201.
- [80] J. Ravagli, Z. Ziran, and S. Marinai, “Text recognition and classification in floor plan images.”
- [81] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: <http://arxiv.org/abs/1506.02640>
- [82] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: <http://arxiv.org/abs/1612.08242>
- [83] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, June 2017.
- [84] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: <http://arxiv.org/abs/1506.01497>

- [85] V. Romero-Gómez, A. H. Toselli, V. Bosch, J. A. Sánchez, and E. Vidal, “Automatic alignment of handwritten images and transcripts for training handwritten text recognition systems,” in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, April 2018, pp. 328–333.
- [86] G. Sadeh, L. Wolf, T. Hassner, N. Dershowitz, and D. S. Ben-Ezra, “Viral transcript alignment,” in *Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, ser. ICDAR ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 711–715. [Online]. Available: <http://dx.doi.org/10.1109/ICDAR.2015.7333854>
- [87] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” in *International Conference on Learning Representations (ICLR2014)*, CBLIS, April 2014, 2014.
- [88] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, “Pedestrian detection with unsupervised multi-stage feature learning,” *CoRR*, vol. abs/1212.0142, 2012. [Online]. Available: <http://arxiv.org/abs/1212.0142>
- [89] D. Sharma, N. Gupta, C. Chattopadhyay, and S. Mehta, “Daniel: A deep architecture for automatic analysis and retrieval of building floor plans,” in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Nov 2017, pp. 420–425.
- [90] —, “Daniel: A deep architecture for automatic analysis and retrieval of building floor plans,” in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Nov 2017, pp. 420–425.
- [91] Z. Shen, Z. Liu, J. Li, Y. Jiang, Y. Chen, and X. Xue, “DSOD: learning deeply supervised object detectors from scratch,” *CoRR*, vol. abs/1708.01241, 2017. [Online]. Available: <http://arxiv.org/abs/1708.01241>
- [92] M. Simone, “Text retrieval from early printed books,” *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 14, no. 2, pp. 117–129, Jun 2011. [Online]. Available: <https://doi.org/10.1007/s10032-010-0146-0>
- [93] D. Singh, “Detection of objects in a floor plan and architectural images,” <https://github.com/dwnsingh/Object-Detection-in-Floor-Plan-Images>, 2019.
- [94] N. Stamatopoulos, B. Gatos, and G. Louloudis, “A novel transcript mapping technique for handwritten document images,” in *2014 14th International Conference on Frontiers in Handwriting Recognition*, Sep. 2014, pp. 41–46.
- [95] W. Sun, J. Burie, J. Ogier, and K. Kise, “Specific comic character detection using local feature matching,” in *2013 12th International Conference on Document Analysis and Recognition*, Aug 2013, pp. 275–279.

- [96] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: <http://arxiv.org/abs/1409.4842>
- [97] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov, “Scalable, high-quality object detection,” *CoRR*, vol. abs/1412.1441, 2014. [Online]. Available: <http://arxiv.org/abs/1412.1441>
- [98] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016. [Online]. Available: <http://arxiv.org/abs/1512.00567>
- [99] G. W. Taylor, I. Spiro, C. Bregler, and R. Fergus, “Learning invariance through imitation,” in *CVPR 2011*, June 2011, pp. 2729–2736.
- [100] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, “Detecting text in natural image with connectionist text proposal network,” in *ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., 2016, pp. 56–72. [Online]. Available: https://doi.org/10.1007/978-3-319-46484-8_4
- [101] K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy, and P. Dosch, “Text/-graphics separation revisited,” in *DAS*, D. Lopresti, J. Hu, and R. Kashi, Eds., 2002, pp. 200–211.
- [102] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for object recognition,” *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, Sep 2013. [Online]. Available: <https://doi.org/10.1007/s11263-013-0620-5>
- [103] K. Wada, “labelme: Image Polygonal Annotation with Python,” <https://github.com/wkentaro/labelme>, 2016.
- [104] C. Yi and Y. Tian, “Text string detection from natural scenes by structure-based partition and grouping,” *IEEE TIP*, vol. 20, no. 9, pp. 2594–2605, Sep. 2011.
- [105] X. Yi, L. Gao, Y. Liao, X. Zhang, R. Liu, and Z. Jiang, “CNN based page object detection in document images,” in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, 2017, pp. 230–235.
- [106] D. Yoo, S. Park, J. Lee, A. S. Paek, and I. Kweon, “Attentionnet: Aggregating weak directions for accurate object detection,” *CoRR*, vol. abs/1506.07704, 2015. [Online]. Available: <http://arxiv.org/abs/1506.07704>
- [107] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee, “Improving object detection with deep convolutional networks via bayesian optimization

- and structured prediction,” *CoRR*, vol. abs/1504.03293, 2015. [Online]. Available: <http://arxiv.org/abs/1504.03293>
- [108] Z. Zhao, P. Zheng, S. Xu, and X. Wu, “Object detection with deep learning: A review,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2019.
- [109] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, “EAST: an efficient and accurate scene text detector,” in *CVPR 2017*, 2017, pp. 2642–2651. [Online]. Available: <https://doi.org/10.1109/CVPR.2017.283>
- [110] Z. Ziran and S. Marinai, “Object detection in floor plan images,” in *Artificial Neural Networks in Pattern Recognition - 8th IAPR TC3 Workshop, ANNPR 2018, Siena, Italy, September 19-21, 2018, Proceedings*, 2018, pp. 383–394. [Online]. Available: https://doi.org/10.1007/978-3-319-99978-4_30