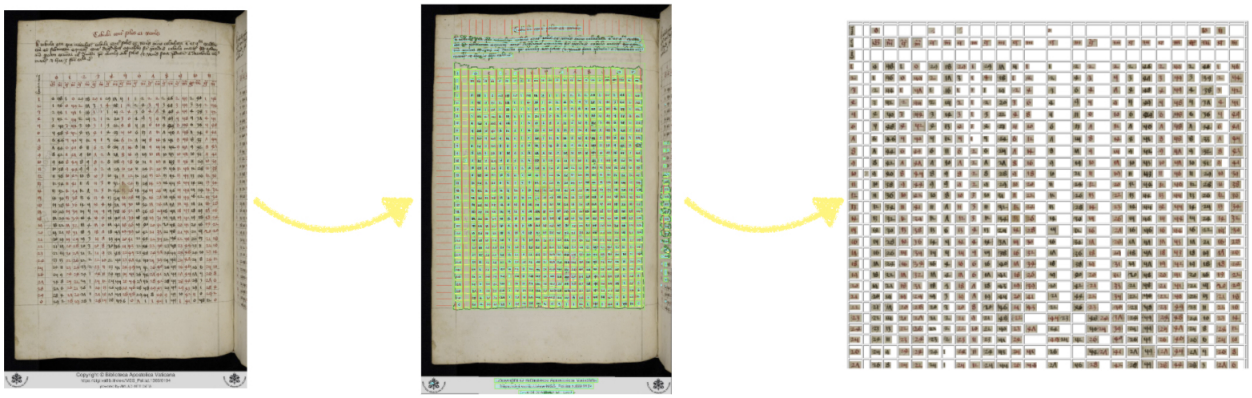
Table detection and structure recognition in historical documents



Author: Tristan Dot

Supervisors: Samuel Gessner,
Tom Monnier,
Matthieu Husson,
Mathieu Aubry

May - August 2020

Abstract

This internship, conducted in the laboratory SYRTE (Paris Observatory), in collaboration with the laboratory IMAGINE (École des ponts ParisTech), was focused on historical document analysis and, particularly, on table detection and table structure recognition. One main question was raised during my months of research: is it possible to accurately detect tables in historical documents, and to efficiently extract the information they contain ? By using a pre-existing generic element extraction network developed within the IMAGINE laboratory, and adapting it to tables extraction issues, satisfactory results were obtained. In particular, our approach currently gives promising results on the SYRTE's astronomical manuscripts dataset. Nevertheless, there is still some work to be done in order to obtain performing results on a wider range of historical datasets.

Keywords: historical documents analysis, table detection, table structure recognition

Acknowledgments

My internship took part in the laboratory SYRTE (Paris Observatory), in collaboration with the laboratory IMAGINE (École des ponts ParisTech). Its realization would not have been possible without the support and the presence of some main interlocutors. I would like to thank them here.

I would firstly like to thank **Samuel Gessner**, for his presence, his comments and advices throughout my internship.

Thanks also to **Tom Monnier**, PhD student at IMAGINE, for his help and comments.

In addition, for their advices and comments, thanks to **Matthieu Husson** and **Mathieu Aubry**.

Contents

1 Introduction

1.1 General introduction

In recent years, a new field emerged, mixing computer science and humanities: the digital humanities. Applying data analysis or artificial intelligence methods to historical documents, this field aims at supporting research in humanities thanks to computer science tools. Helping in the analysis, automatic labeling, or comparison of very large corpus of textual and visual documents, digital humanities should facilitate the exploration of past times, and could favor new discoveries in history, art history or history of science. Indeed, in a context where more and more data are accessible to researchers, powerful automatic analysis tools appear necessary to efficiently explore them, and draw conclusions from them.

At the Paris Observatory, the ALFA ('Alfonsine Astronomy')[1] project, which is part of the history of astronomy laboratory SYRTE ('Systèmes de Référence Temps-Espace')[2], has decided to take advantage of recent advances in computer vision in order to automatically analyze medieval manuscripts. Indeed, the ALFA project studies old astronomical manuscripts, with a main focus on one of their elements: the astronomical tables they contain.

If some algorithms, developed in the laboratory IMAGINE[3] and based on deep neural networks [1], already allow a very accurate segmentation of illustrations and handwritten text in historical documents, the goal of this internship was to extend this segmentation task to historical tables: in order to efficiently detect them, but also to efficiently extract all the information they contain. Such an achievement would be greatly beneficial to the ALFA team, paving the way for historical studies on larger corpus, and, more generally, saving a lot of transcription time.

Besides the ALFA project, and its specific goals, table detection and structure recognition constitutes a hot topic in document analysis. While table detection can be considered as just another document layout analysis task, table structure recognition appears particularly challenging, considering the high variability of tables and cells structures / layouts in modern documents, and, even more, in archival documents. Interestingly, most of the current papers and available datasets in the field are focused on pdf-born tables analysis, leaving aside a huge variety of handwritten tables. Because of this lack, our work, which is focused on archival tables analysis, appears as an interesting contribution.

Working on the Paris Observatory's medieval dataset, and on the ICDAR (International Conference on Document Analysis and Recognition) 2019 table competition archival dataset [2], we achieved satisfactory scores on tables detection tasks. Our tables' structure recognition pipeline works qualitatively well on the Paris Observatory's dataset, but still needs some adjustments to be effective on more generic corpus of tables. The final goal would be to get an algorithm which exactly converts images of tables into corresponding XML / HTML translated

[1] https://alfa.hypotheses.org/
[2] https://syrte.obspm.fr/spip/?lang=en
[3] https://imagine-lab.enpc.fr/

tables, with the adequate structure and spanning for every cell, the content of which should be digitized. In a first time, cells' content digitization can be set aside.

Our contributions can be summed-up in three points:

- we make available a new dataset of annotated historical tables, very different from existing ones, which could be useful to the community: the *DISHAS table dataset* (Sec. 2.2)
- we adapted Monnier et al. synthetic generation process [1], and created a learning dataset adapted to historical tables: *Syndoc_tables* (Sec. 3.1)
- we proposed two different networks architectures in order to detect tables and their rows / columns separators (Sec. 3.2.1 and Sec. A.2), one of which gives very encouraging results (Sec. 4)

Context:

This internship took place in a particular context: I was part of the ALFA team, a history of science team -composed of varied profiles- within which I was able to satisfy my curiosity for the history of science, and for which I had to achieve a relatively concrete objective: an efficient table extraction tool, on their own dataset. In parallel, I worked in collaboration with the IMAGINE team for technical and theoretical issues concerning machine learning.

The ALFA team is dedicated to the study of Alfonsine Astronomy, which flourished in Europe from the second half of 13th century to the middle of the 16th century. Based on the Ptolemy's geocentric model of the solar system, slightly adapted to the observation of the astronomers of the times, those tables constituted a major European scientific achievement, and allowed the emergence of thinkers like Regiomontanus or Copernicus. The ALFA team is particualrly interested in the digital availability of their study corpus, as well as its digital analysis. Please check the website of one of their main projects: the DISHAS (Digital Information System for the History of Astral Sciences)[1] project, for more details and illustrations of their works. This internship aims to add a new stone to the DISHAS project: the automatic extraction of information relating to astronomical tables, directly from the manuscripts' scans images.

1.2 Related works

Because of important variations in tables structures and layouts, but also in tables' cells layouts (empty cells, cells spanning multiple rows / columns), table detection and recognition tasks are particularly tricky. Recent works tried to address those issues using different machine learning methods and, particularly, deep learning methods. If research in table detection already constitutes an old field, with a substantial literature, research in table structure recognition is a much more recent and challenging topic.

[1] https://dishas.obspm.fr/

1.2.1 Table detection

Before the advent of machine learning, traditional approaches for table detection were based on heuristics, metadata or predefined tables templates. In 1997, Pyreddy et al. [3] exploited documents structural information to identify tables and their components fields. In 2002, Cesarini et. al. [4] used MXY tree representations of documents to detect tables, constituting the first machine learning method for such a task. More recently, Kasar et al. [5] used a set of hand-crafted low-level features combined with a SVM classifier in order to detect tables. Probabilistic graphical models were also used for table detction: Silva [6] developed a succession of Hidden-Markov-Models (HMMs) to model the joint probability distribution over sequential observations of visual page elements and the hidden state of a line belonging to a table or not.

After 2016, deep learning models, based on object detection and semantic segmentation, brought significant performance improvements. Those models have been efficiently used for documents layouts general understanding [1, 7, 8], but also, more specifically, for table detection.

In 2017, DeepDeSRT [9], employed the object detection network Faster R-CNN [10] and transfer learning to locate tables on images, and achieved particularly performing results. The same year, Gilani et al. [11] used distance transform encoded information in an image, and also applied Faster R-CNN on these images. In the same way, Siddique et al. [12] used a Faster R-CNN network, with a Deformable Convolutional Neural Network [13] as feature extractor. Kavasidis et al. [14] employed Conditional Random Fields (CRFs) on saliency maps extracted by a Fully Convolutional Network (FCN) [15] to detect tables, and different types of charts. More recently, Huang et al. [16] proposed a table detection network based on an adapted YOLOv3 [17] network.

Finally, in a revealing way, nearly all the participants (10 out of 11) of the ICDAR 2019 Table competition [2] used, for the table detection task, one of the following models (more or less adapted): Faster R-CNN, YOLO, FCN, or U-Net [18]. The only team that did not use deep models, but preferred a table detection based on morphological operations, achieved very low scores in this competition.

1.2.2 Table structure recognition

Table structure recognition is linked to documents structural and semantic understanding. It consists in extracting tables content into a machine-readable format. And it recently became a relatively active topic in the community.

The first tables recognition methods, developed during the 1990s, were based on PDF metadata, hand-crafted features and heuristic. In 2004, Wang et al. [19] proposed a new process, using probability optimization methods similar to the X-Y cut algorithm. Differently, Kieninger et al. [20] used a bottom-up approach to recover the tables structures: grouping detected words into columns based on their x-axis overlap. Those methods made strong assumptions on tables layouts.

More recently, thanks to deep learning, more generic methods appeared. DeepDeSRT [9] were among the first networks to deal with table structure analysis. After a global table detection step, realized by a Faster R-CNN network, the tables rows and columns were detected thanks to a FCN network. The processed images were previously stretched, vertically and horizontally, to facilitate the tables rows and columns separation. Another method, TableNet

[21], is based on a single FCN network, divided in two branches, to detect tables, and each one of their columns. TableNet slightly outperformed DeepDeSRT on the ICDAR 2013 table competition benchmark dataset [22]. Interestingly, it still uses heuristics (i.e. particular rules) to segment tables rows. Another deep model, called SPLERGE [23], first splits tables into grids, and then merges grid elements which are semantically connected to obtain the final table structure. Recently, Prasad et al. [24] used a Cascade R-CNN [25] to simultaneously detect tables and tables cells, before deducing the tables structures.

Far from the precedent FCN-based approaches, Vine et al. [26] explored an interesting top-down approach, which maps tables images into their corresponding skeletons thanks to a generative adversarial network (GAN), and then fit latent table structures into these skeletons using a genetic algorithm.

In the past months, many works have preferred a graph-based formulation of the problem, as graphs are inherently ideal structures to model structural associativity. Qasim et al. [27] formulated table recognition problem as a graph problem, and used graph neural networks to model table-level associativity between words. Li et al. [28] proposed another graph-based problem formulation and solution, such as Chi et al. [29] who used a graph attention mechanism. In the same idea, Raja et al. [30] mixed, in a single network, a top-down table cells detection, performed thanks to a Mask R-CNN network [31], and a bottom-up structure recognition process, built on top of graphical model [27]: once all the cells were correctly detected, their adjacency matrices were computed thanks to a graphical neural network.

Nota bene:

Most of the previous related works are only applied to (and thought for) modern documents analysis. Nevertheless, their approaches are conceptually adaptable to other kind of documents, and, particularly, archival documents - which interest us. To our knowledge, our work is the first one to specifically deal with historical tables analysis.

1.2.3 Competitions

There are two benchmark competitions for table detection and table structure recognition: ICDAR 2013 Table Competition [22], and ICDAR 2019 Competition on Table Detection and Recognition (cTDaR) [2]. Each of these competitions is associated with a specific dataset, and evaluates both the ability to correctly detect tables within input images, and to recognize their structures, i.e. to associate a correct row and column index to each of its detected cells (taking into account multiple-rows and multiple-columns cells).

Interestingly, cTDaR 2019 provides an archival tables dataset to analyze. It will then be the competition on which we will focus. Eleven teams (scholars, corporations or individuals) participated to its table detection track, while only two teams participated to its table structure recognition track. Nevertheless, some recent papers brought new state of the art results concerning this structure recognition track, such as [24] (for modern documents) or [30] (for archival documents).

2 Datasets

A considerable number of benchmark datasets exist for table detection and structure analysis, such as *SCiTSR* [29], *ICDAR 2013* table competition [22], *ICDAR 2019* table competition [2], *UNLV* [32], *Marmot* extended [21], *TableBank* [33] and *PubTabNet* [34]. But only one, from ICDAR 2019 (cTDaR), is composed of archival images. Since our work is focused on table analysis in historical documents, it will be the only public dataset of interest for us.

2.1 cTDaR 2019 archival dataset

The historical dataset of cTDaR 2019 [2] is composed of a great variety of tables scans coming from 23 different institutions. Different tables origins can be noted: hand-drawn accounting books, stock exchange lists, train timetables, prisoner lists, simple tabular prints in books, etc. All those documents come from modern times.

A part of this dataset has been annotated for table detection only, while another part has been annotated for table structure recognition. Each image is linked to an XML file containing:

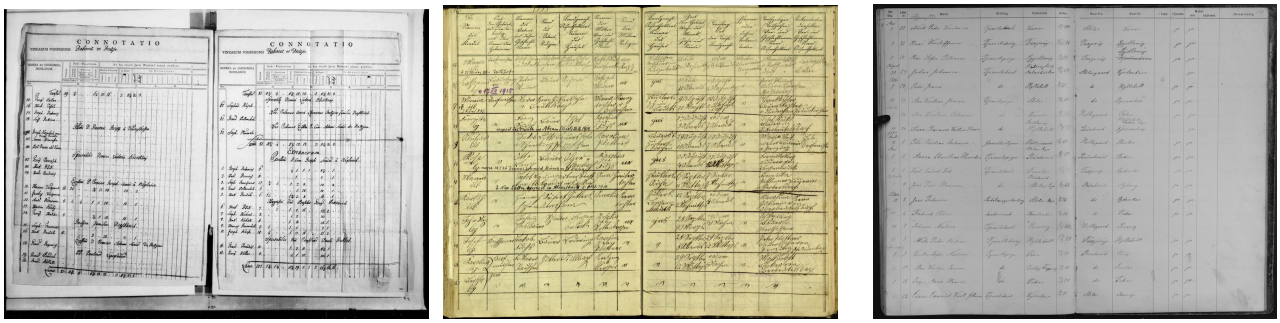
- for the detection dataset: the coordinates of the bounding polygons of the tables boxes
- for the structure recognition dataset: the coordinates of the tables polygons plus, for each cell element, the attributes: start-row, start-col, end-row and end-col, and the coordinates of its corresponding box. N. B.: the corresponding box of a cell is the cell boundary, and not the convex hull of its content

The number of training and test images, in each case, is summed up in Table 5.

Table 1: Number of training and test images of the cTDar historical dataset

	Train	Test
Table detection	600	199
Table struc. reco.	600	150

Some tables examples can be seen on Fig. 11.



(a) (b) (c)

Figure 1: Example documents of the cTDaR 2019 historical dataset

2.2 DISHAS dataset

Within the ALFA project, an important number of European medieval astronomical manuscripts (more than 600) are studied. Those manuscripts mainly date from the 14th, 15th and 16th centuries, are written in Latin, and, in a vast majority of cases, contain astronomical tables. These tables are for the most part derived from the Alfonsine tables (named after the Castilian King Alfonso the Wise ruling from 1252 to 1284). Those tables enabled calculation of eclipses and the positions of the Sun, Moon and planets for any given time. For more than two centuries in Europe, these tables were the main tool for astronomical computations. To be more precise, Alfonsine tables relied on the tradition of Ptolemy's theory and tables (1st c. CE), modified over the centuries in the Muslim world from which the Latin scholars inherited computational astronomy during the 12th and 13th centuries.

The DISHAS project provides an online tool for table transcription, edition and analysis, including an interface for historical and systematic browsing[1], for manuscripts in the computational astronomy worldwide, including the Alfonsine manuscripts. Moreover, a part of those manuscripts, which have been publicly digitized, has been grouped by the DISHAS project, and is available online[2]. This catalog contains 89 manuscripts, coming from the Bibliothèque nationale de France (Paris), the Biblioteca apostolica vaticana (Vatican), and the Bayerische Staatsbibliothek (Munich). In the future, some medieval Arabic manuscripts could join this corpus.

Although dating from different centuries, the tables studied by DISHAS have many common points in terms of layouts. They are mainly composed of numbers, and present very tight rows / columns. Sometimes, entire columns are written in colored ink (to help with calculations). The vast majority of their cells only span one row and one column, apart from a few header cells. Their overall layout is very compact.

From this DISHAS corpus, we annotated different sub-ensembles, considering three main criteria:

- *table presence annotation*: ∼7500 pages, coming from 17 manuscripts, were binary annotated, depending on the presence or absence of at least one table within them
- *table area annotation*: ∼310 pages with tables, coming from 9 manuscripts, were annotated using VGG Image Annotator (VIA) [35]: each table surrounding polygon was annotated. **N.B.** Tables regions can be constituted of blank cells, if a grid is present.
- *table cells annotation*: ∼20 pages with tables, coming from 9 manuscripts, were annotated at the text-cell level using VIA: each cell's content (x-height) convex hull was annotated

Consequently, we can propose two new datasets of interest to the community:

1. a new table detection dataset, of 310 elements: the DISHAS table detection dataset
2. a new tables cells detection dataset, of 20 elements: the DISHAS cell detection dataset

[1] https://dishas.obspm.fr/historical-navigation
[2] https://dishas.obspm.fr/js/mirador-alfa/index.html

N.B.: for the moment, we have not added cells' row and column index informations to the DISHAS cell detection dataset. We should do so in order to transform it into a truly usable table structure recognition dataset.

We think that these datasets constitute a valuable contribution, given the great lack of historical tables corpuses available to the community. By their time periods, their layouts, their typography, language and content, our dataset's tables differ widely from the tables of the cTDaR 2019 historical dataset, presented in Sec. 2.1.

Fig. 2 shows some examples of the DISHAS table detection dataset.

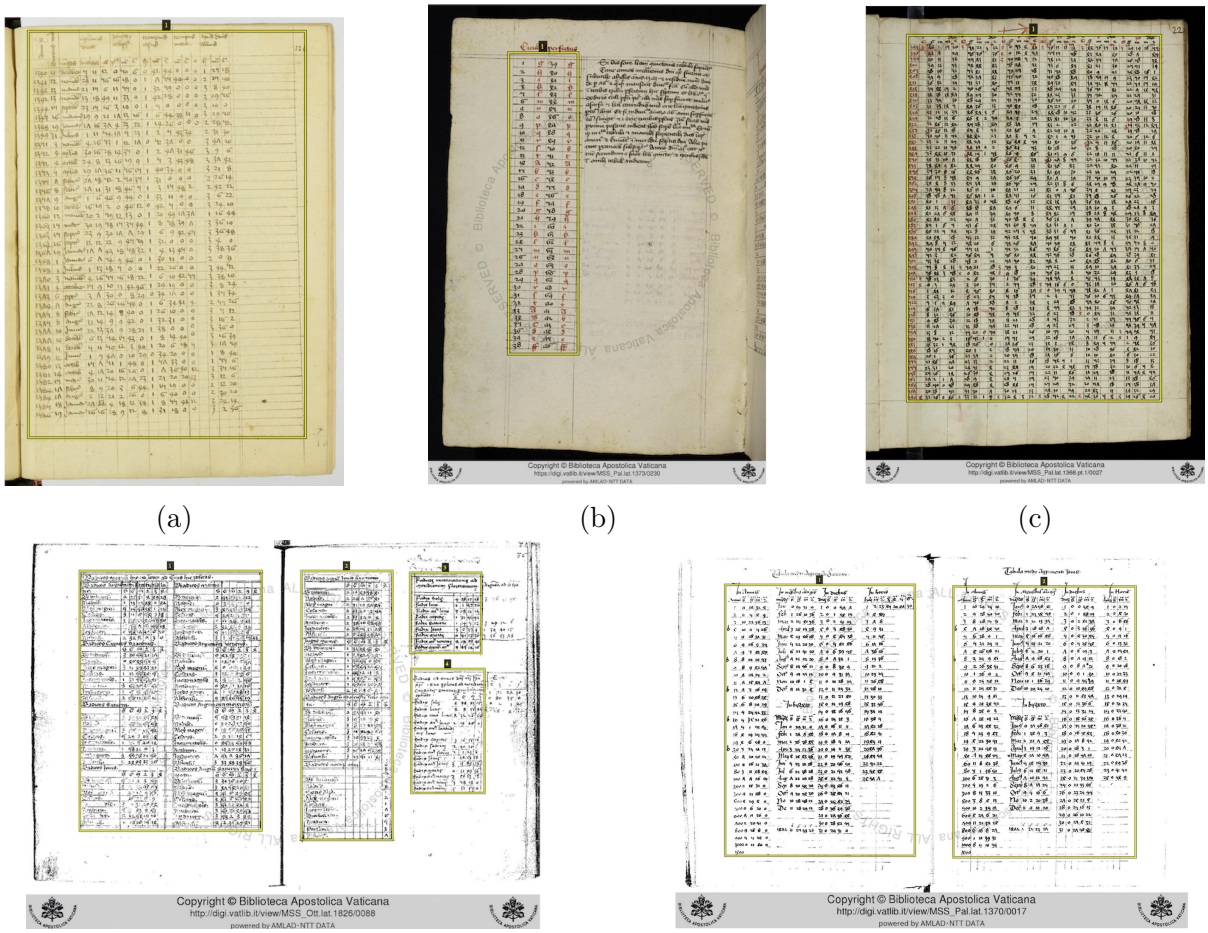


Figure 2: Example documents of DISHAS table detection dataset

But how such tables can be detected ? Using a synthetic table dataset, and an adapted neural network, as presented in next part.

3 Approach

Our approach, for table detection and structure recognition, is based on two main elements: a large scale synthetic table dataset, adapted from the *SynDoc* dataset [1], and a network, to perform image segmentation, i.e. pixel-wise classification.

We have explored two distinct networks architectures:

- a fully convolutional network, which simultaneously detects cells, tables and rows / columns separators (Sec. 3.2.1)
- Two independent networks: a first classification network detects cells and tables, and a second regression network detects tables separators (in Annex, Sec. A.2)

We mainly studied, implemented and tested the first architecture, composed of a single FCN. Nevertheless, the second approach could also be promising.

Our global pipeline for table structure recognition can be summed up in three points:

1. generation of a synthetic table dataset, as various as possible
2. training of the network(s) to detect: tables, cells' content, and rows and columns separations
3. based on the cells and separations detections, table structure inference: the row ranges and column ranges of cells are determined

We are now going to detail our synthetic generation process, and our neural network architecture.

N.B.: contrary to graph-based table structure recognition approaches, such as [27, 28, 29, 30], which generally detect cells and then infer their adjacency matrices thanks to an adapted graph neural network, our method is highly dependant on rows and columns separations detections. This is the case of other approaches in the literature, such as [9, 21, 24]. The winner team of cTDaR 2019 structure recognition challenge also based their method on such a line detection approach.

3.1 Synthetic tables

A single real-world dataset exists for table detection and recognition: the cTDaR 2019 archival dataset (Sec. 2.1). Given its limited size and its lack of diversity, it could not be used as a general training dataset for table segmentation tasks. Even if transfer learning could have been used to generalize the networks results to other dataset (such as DISHAS dataset), as shown by [9, 21, 24], it would have required a large number of manual annotations, which would have been extremely time consuming.

To deal with those issues, we decided to work on a large synthetic dataset, adapted from *SynDoc* dataset [1]. Synthetic data generation has already been used in table detection [27, 36]. The limit of such an approach lies in its tricky generalization to real world data.

SynDoc dataset is thought to simulate a large variety of historical-like documents, randomly mixing various pages backgrounds, pages layouts, visual degradations and elements types [1]. It uses a large database of illustrations and fonts downloaded from the web to increase its diversity. Its textual elements can be generated in different layouts, including a table layout. But the generated tables are extremely basic, and cannot cope with the complexity of real-world tables. We had to enhance this table generation process.

3.1.1 Table generation

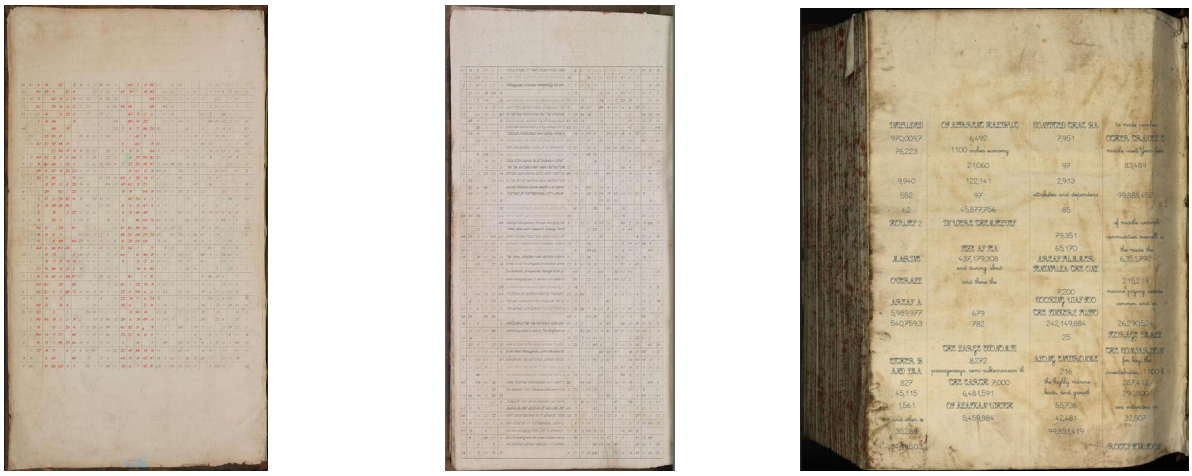
We adapted *SynDoc* to tables segmentation issues. The resulting dataset, *SynDoc_tables*, is generated as follows.

First, a document is created, with a very high probability of containing a table. If so, then:

1. A table layout is randomly chosen chosen among two types: *compact* and *loose*
2. A sub-category layout is composed, according to the presence or absence of the following elements:
   - row / column separators
   - empty rows / columns
   - colored columns
   - multiple spanning cells
   - legends
3. If double page layout: the table can, or not, spread among the two pages
4. Some cells content is randomly deleted
5. Random opacity levels are chosen for the following elements: rows / columns separators, textual elements

Very diverse tables are consequently generated. The *compact* layout is thought to mimic the DISHAS tables layouts (Sec. 2.2). *Compact* tables are mainly composed of numbers, and present very tight rows and columns. The *loose* layout is closer to modern tables layouts, such as those of the cTDaR 2019 competition (Sec. 2.1): cells are bigger, and can contain words, or even multi-lines sentences.

Some *SynDoc_tables* examples can be seen on Fig. 3. We generated 10K *SynDoc_tables* elements in order to train our network.



(a) (b) (c)

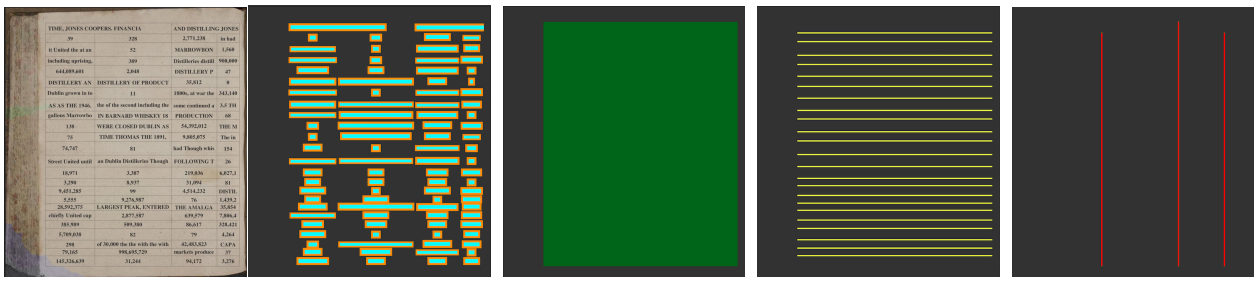
Figure 3: Example synthetic tables: (a) *compact* with coloured columns, (b) *compact* with legends, (c) *loose* without row separators

3.1.2 Table labeling

The table labeling depends on the network architecture. In the single network case, all the labels are binary. In the Two networks case, signed distances to row and column separations are introduced. See Annex, Sec. A.2, for an illustration of the signed distance labeling.

Binary labeling:

In the single network case, the elements are discreetly labeled. Concretely, a label is defined for: tables, cells content (with borders), row separations, and column separations. Fig. 4 illustrates a typical table labeling in this case.



(a) (b) (c) (d) (e)

Figure 4: Tables components labeling, single network case: (a) input *loose* table with multiple rows cells, (b) cell content (with border) label, (c) table label, (d) row separations label, (e) columns separations label

3.2 Segmentation network

Two possible networks were considered, with their own advantages and disadvantages: a single fully convolutional network, and two independent networks. For an explanation of the two networks approach, please see Annex, Sec. A.2.

3.2.1 Single network approach

Because of the very close proximity between some elements we want to detect (such as rows and columns separations), and of their high number of occurrences, object detection networks such as Faster R-CNN [10], don't seem adapted. Fine-grained images segmentations are usually obtained by FCN-Xs architectures, based on fully convolutional networks, skip connections, and fractionally strided convolutions. Such a FCN choice was made by [9, 21] to detect tables' columns.

Similar to [21], we use a single fully convolution network to simultaneously detect the tables and some of their semantic components. Such a parallel detection could be positive: the rows and columns separators segmentation could help the network understand what a table is, and favor accurate tables detections. This hypothesis is discussed in Sec. 5.2.

Such as [1, 8], we used a U-Net [18] architecture with a ResNet [37] backbone encoder. We used the same ResNet-18 encoder as [1], with 2-strided 3x3 convolutional layer instead of maxpooling. This favorizes small text lines detection.

In our case, multi label segmentation should be performed. Indeed, some pixel can be simultaneously classified to different classes, such as: *table* and *text*, or *table* and *column separator*. Consequently, we have added several final branches to the network to take into account multi labeling. The network is optimized with a sum of binary cross-entropy losses.

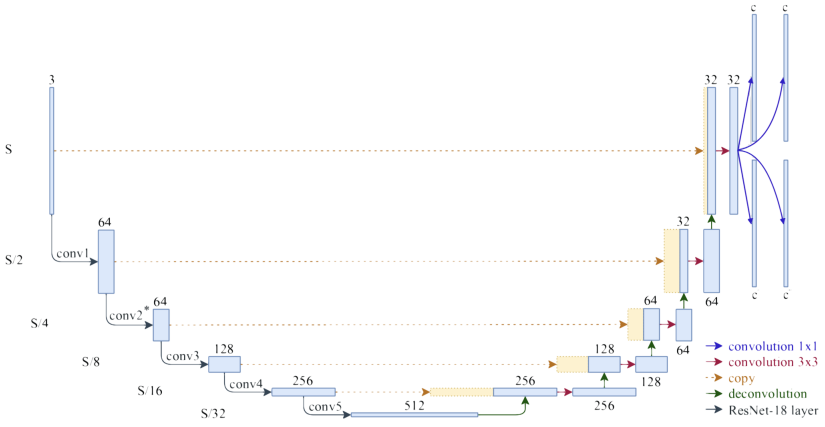


Figure 5: Single network architecture: U-Net adapted to deal with multi-label segmentation

Fig. 5 shows an illustration of our network. One branch is dedicated to illustration and text (including text cell) segmentation. A second one is dedicated to tables segmentation, another one to horizontal table separators segmentation, and a last one to vertical table separators segmentation. An example of the ground truth image fed to each branch can be seen on Fig.

4. Similar to [1], detected connected components with low area are filtered before output, depending on class-specific ratio thresholds.

Comments:

Other possible network architectures could be considered. For instance, a single branch network could be used, simultaneously segmenting illustrations, text, tables borders and tables separators, and avoiding the multi label segmentation problem. Other architectures have been tested, but they all have defects, and lead to less performing results, as presented in Sec. 5

But such a network has a big flaw: the binary classification of row and column separators will only allow them to be detected when they are visually present, as lines, on the images. A large majority of tables, in the DISHAS and the cTDaR datasets, effectively have separators drawn as lines. But some tables simply have white spaces between two adjacent rows. This is a big issue, that our second network, which chains two independent networks, aims to correct. Please see Annex, Sec. A.2 for a description of the corresponding labeling, and architecture. Even if implemented, this second approach has not been quantitatively tested yet. In all the following experiments, only the single network approach has been tested.

4 Experiments and results

Different experiments have been set up to evaluate the performances of our approach. Some of those experiments can be directly useful to the Paris Observatory ALFA team, to help them with automatic analysis of large corpus of manuscripts.

4.1 Implementation details

Similarly to [1], input images are resized so that their larger side is 1280 pixels. Per-channel standardization is performed, and on-the-fly data augmentation is applied: contrast variation, image rotation and transposition, Gaussian noise, random scaling. We process one sample per batch and use Instance Normalization [38] with a momentum of 0.1 instead of Batch Normalization. We use ImageNet [39] pretrained weights for the encoder, and Xavier initialization [40] for the other convolutional layers. We train for 50 epochs with Adam optimizer [41] with a weight decay of 10^{-6}. The learning rate is initially set to 0.01 and divided by 2 all 10 epochs.

4.2 Evaluation metrics

To quantitatively evaluate table and cells detection on DISHAS annotated dataset, we implemented the mean Average Precision (mAP) metric, as defined in the PASCAL VOC 2012 competition [42]. This definition considers an Intersection over Union (IoU) threshold of 0.5 for a detect object to be considered as a True Positive. Another definition of mAP exists, coming from the COCO competition [43], with an AP averaging over multiple IoU thresholds, from 0.5 to 0.95 with a step size of 0.05. Nevertheless, the PASCAL VOC 2012 definition leaves more room for small irregularities in our annotations, and we decided to choose it. We adapted an

open implementation found online[1]. mAP appeared to be a more pertinent evaluation metric than IoU only, considered the importance of layouts detection in our case.

For the cTDaR 2019 competition [2], another evaluation metrics was considered, both for table detection and table structure recognition: weighted average F1 (WAvg. F1), calculated from F1 scores with IoU thresholds of 0.6, 0.7, 0.8 and 0.9. The WAvg. F1 value is defined as:

$$WAvg.F1 = \frac{\sum_{i=1}^{4} IoU_i \cdot F1IoU_i}{\sum_{i=1}^{4} IoU_i}$$

The cTDaR 2019 measurement tool is available online [2].

4.3 Table presence binary classification

A first necessary experiment consists in the binary classification of a large number of DISHAS manuscripts pages into two categories: *page with table*, and *page without table*. A page is considered as a *page with table* when at least one table area is detected on it by our network (after the post processing step).

The test dataset is composed of 7450 pages, coming from 17 different medieval manuscripts, which were binary annotated as containing a table, or not, by members of the DISHAS team.

Our precision-recall curve can be seen on Fig. 6. The average precision is of 0.96. It appears quite satisfactory, but could certainly be increased a bit.

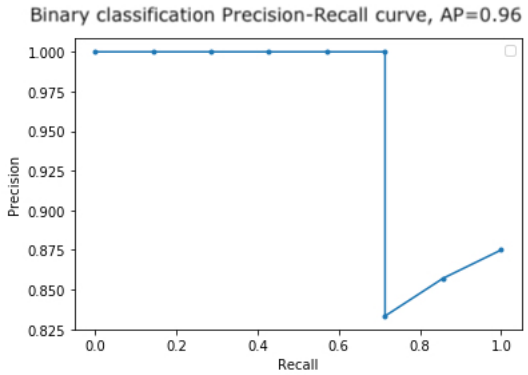


Figure 6: Precision-recall curve, for table presence binary classification

False positives could be reduced by diversifying illustrations and text pages in the synthetic dataset (addition of lines without table presence). False negatives could also be reduced by diversifying still a bit the synthetic table generation process. Some fine tuning on real DISHAS tables could also largely increase those results.

[1] https://github.com/Cartucho/mAP

[2] https://github.com/cndplab-founder/ctdar$_{m}easurement_{t}ool$

4.4 Table detection

Table detection was performed on two datasets, with two indepent metrics: mAP for DISHAS dataset, and WAvg. F1 for the cTDaR 2019 dataset.

4.4.1 DISHAS dataset

We performed table detection on the 310 elements of the *DISHAS table detection dataset*, presented in Sec. 2.2. We obtained, without fine-tuning, a mAP of 0.90. The synthetic table dataset appears to mimic well the DISHAS dataset. Fig. 7 shows the corresponding precision-recall curve.
As complementary information, Tab. 4 shows the table AP score, for IoU thresholds greater or equal than 0.5. This score is stil very good for an IoU threshold of 0.8: it is a satisfactory point, table areas are gloablly well detected.

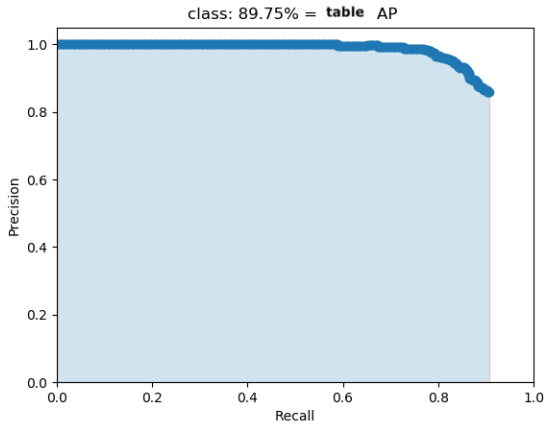


Figure 7: Precision-recall curve, table detection without fine tuning

Table 2: Table AP for various IoU thresholds, without fine tuning

IoU threshold	0.5	0.6	0.7	0.8	0.9
Table AP	0.90	0.87	0.84	0.80	0.60

4.4.2 cTDaR 2019

For the cTDaR 2019 competition (Track A, archival) [2], Tab. 3 sums-up our results, with or without fine-tuning, on the 600 images annotated for table detection.

If our results without fine tuning appear encouraging (given the wide variety of test tables), our fine-tuned results are not amazing. It appears that our network very quickly overfits on the small training dataset given by the cTDaR competition, despite data augmentation of the input data. We should work on increasing this fine-tuned score.

Table 3: Results for the archival dataset, Track A, cTDaR 2019

Rank	Team	WAvg.F1
1	TableRadar	**0.94**
2	NLPR-PAL	0.93
3	Lenovo Ocean	0.91
4	TJNU202-2	0.85
5	Applica-robots	0.82
6	*Ours (fine-tuned)*	*0.73*
7	Cinners-table	0.64
8	Table Fan	0.63
9	*Ours*	*0.59*
10	ABC Fintech	0.58
11	Clova AI	0.58
12	HCL IDORAN	0.35
13	AIRL	0.17

4.5 Table cells detection

Another important element of table structure recognition is a correct tables cells detection. Indeed, cells detection is necessary to infer their row and column ranges in the table. A small part of the DISHAS dataset has been annotated for such a task evaluation (see *DISHAS cell detection dataset*, in Sec. 2.2). The cTDaR 2019 dataset also presents, in its Track B (structure recognition), some rich annotations for cells coordinates and corresponding row and column ranges. But those annotations depend on the cells borders, and not on the text cell convex hull. Our network, which for the moment detects cells text, is not adapted for such a task. We have therefore not been able to evaluate it on the cTDaR 2019 dataset. It constitutes an important 'to do' thing.

DISHAS dataset:

We obtained, without fine-tuning on the real DISHAS datast, a cell average precision of 0.69. Fig. 8 shows the corresponding precision-recall curve.

The scores are less good for cell detection than for table detection (Sec. 4.4.1). This can be explained by: less precise ground truth annotations for small cells, and merged detection of neighboring text cells when they are very close in tables.
As complementary information, Tab. 4 shows the cells AP score, for various IoU thresholds around 0.5. It appears that cells AP is very bad for an IoU threshold greater than 0.7. This can be explained by our not very precise ground truth annotations on small cells.

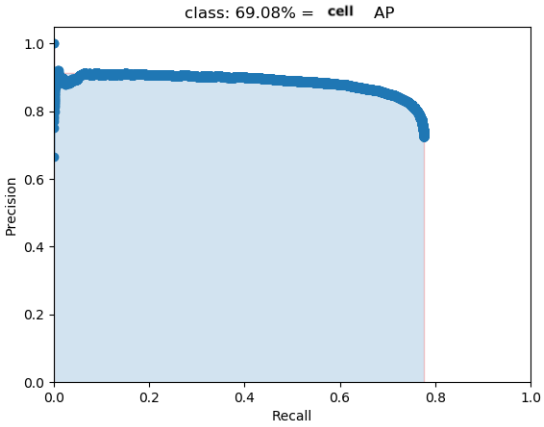


Figure 8: Precision-recall curve, cells detection

Table 4: Cells AP for various IoU thresholds, without fine tuning

IoU threshold	0.3	0.4	0.5	0.6	0.7
Cells AP	0.88	0.83	0.69	0.37	0.08

4.6 Table structure recognition

The cTDaR 2019 dataset (Sec. 2.1) presents rich annotations in term of tables structures but, as explained before, is based on cells borders detection, that our network does not output for the moment. We will overcome this issue in near future, as discussed in Sec. 6, but we cannot evaluate structure recognition on this dataset for now.

Concerning the *DISHAS cell detection dataset* (Sec. 2.2), an important addition should be made in order to evaluate our capacities in table structure recognition: each cell row and column index and spanning should be added. For the moment, we are reduced to qualitative analysis on this dataset.

Being reduced to qualitative results, we decided to create a complete pipeline transforming images of tables into HTML tables. HTML tables can be easely shared with the members of the ALFA project. Each cell of the resulting HTML table shows a cropped image of the corresponding original table cell. The qualitative results are encouraging, even if some neighboring cells always tend to merge.

Fig. 9 shows an illustration of such a conversion, from an image of table to an HTML table of cropped cells images. For other qualitative examples, please check this webpage[1].

[1] http://imagine.enpc.fr/ dott/notebooks/HtmlTranslator.html

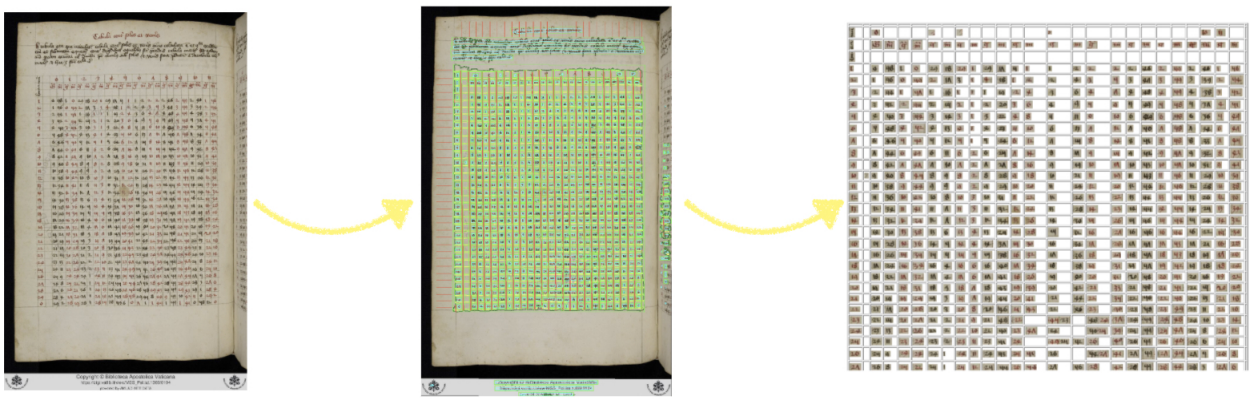


Figure 9: Conversion pipeline, from image of table to HTML table of images

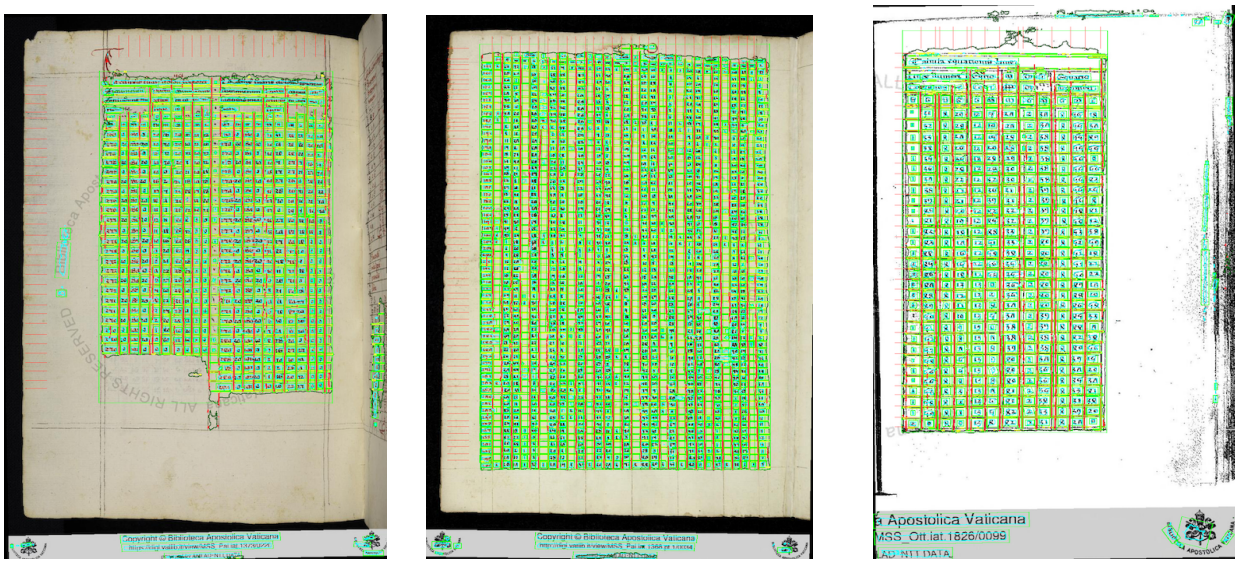
Technically, our conversion pipeline is the following one:

1. Detection of: tables, cells, horizontal separators and vertical separators
2. Rotation of the image according to the average angle of all the vertical separators
3. After a small post processing (deletion of too close detected separators), projection of the column separators on the x-axis, and of the row separators on the y-axis: creation of the table grid. See Fig. 10 for an illustration of the grid detection on various images.
4. Indexation of each detected cell depending on its maximum and minimum coordinates, and of the table grid.
5. Output of the HTML file

Comments:

One of the dilemna with such a method is to choose between the detected grid structure and the detected cells: if a cell spreads among multiple rows or columns, should we cut it, or should we trust its detected spanning ?

Besides, this grid detection method could be a solution to easily detect cells borders (and not cells content) on various datasets, such as cTDaR dataset. The big issue with such a method is the suppression of multiple spanning cells. But such cells seem to be, statistically, very rare in the cTDaR dataset, and almost non-existent in the *DISHAS* dataset.



(a) (b) (c)

Figure 10: Grid detection on various input images. Vertical separators are labeled in red, horizontal separators are labeled in yellow. After a post processing step, the projections of the separators on the x-axis and the y-axis (i.e. the grid definition) can be seen as red dashes in the margins of the images.

5 Ablation study

In this section, we are showing the quantitative impacts, on tables and cells detection, of each of our contributions, both in the synthetic tables generation process and in the network architecture choice.

5.1 Synthetic tables

Scores depending on the training dataset can be seen in Tab. 5.

A *SynDoc_table* training presents a very considerable increase for each score, compared to the original *SynDoc* dataset. A typical case of partial *DISHAS* table detection, after learning on the original *SynDoc* dataset, can be seen on Fig. 11a. The original *SynDoc* dataset didn't present a sufficient variety of tables layouts to cope with the complexity of historical tables.

Table 5: Table AP, Cell AP and WAvg.F1 for various training datasets, on our single network (Sec. 3.2.1)

	Table AP (DISHAS)	Cell AP (DISHAS)	WAvg.F1 (cTDaR)
SynDoc_table	**0.90**	**0.69**	**0.59**
SynDoc_table w/o legends	0.89	0.66	0.58
Original *SynDoc*	0.36	0.12	0.27

5.2 Apports of the structure detection

In this section, we studied the impact of structure detection (rows and columns separators detection) on table detection, and cells detection. Such a study is important to compare our Single Network approach (Sec. 3.2.1), and the Two Networks approach (Annex, Sec. A.2). Indeed, in the Single Network case, the tables, structures and cells are detected at the same time, while only the tables and text are detected simultaneously in the Two Networks case. We therefore compared the two following networks, trained on the *Syndoc_tables* dataset:

- 4 branches: architecture presented in Sec. 3.2.1, which detects tables, cells and separators.
- 2 branches: the segmentation network presented in Sec. A.2 (left network on Fig. 13), which only detects tables and cells.

Results can be seen on Tab. 6, Tab. 7 and Tab. 8. It appears that the simultaneous detection of separators, tables and cells increases the tables and cells detection scores. It could constitute an argument in favor of a Single Network approach (Sec. 3.2.1), and against a Two Networks architecture (Sec. A.2).

Table 6: Table AP, Cell AP and WAvg.F1 for various networks architecture, trained on *Syndoc_tables* dataset

	Table AP (DISHAS)	Cell AP (DISHAS)	WAvg.F1 (cTDaR)	Separators detection
4 branches	**0.90**	**0.69**	**0.59**	Yes
2 branches	0.77	0.63	0.57	No

Table 7: Table AP for various IoU thresholds, on the DISAHS dataset, depending on the network

	AP @ 0.5	AP @ 0.6	AP @ 0.7	AP @ 0.8	AP @ 0.9
4 branches	**0.90**	**0.87**	**0.84**	**0.80**	**0.60**
2 branches	0.77	0.72	0.67	0.58	0.31

Table 8: Cells AP for various IoU thresholds, on the DISAHS dataset, depending on the network

	AP @ 0.3	AP @ 0.4	AP @ 0.5	AP @ 0.6	AP @ 0.7
4 branches	**0.88**	**0.83**	**0.69**	**0.37**	**0.08**
2 branches	0.79	0.75	0.63	0.36	0.08

5.3 Network architecture

Our Single Network architecture, composed of 4 final branches, has been presented in Sec. 3.2.1. But others network architectures could have been considered. Among them, we have evaluated the following ones, which seemed particularly relevant:

– 1 branch: segmentation in one branche of: illustrations, text, rows and columns separator, tables interstitial pixels (others than cells and separators).
Pros: avoids the multi-label issue, simple.
Cons: very tight separators labels: impossible structure recognition. Bad tables interstitial pixels detection: fragmentation of the global table detection.

– 3 branches: one branch for text and illustrations segmentation, one branch for binary table segmentation, one branch for binary separators segmentation.
Pros: elegant: one branch for table detection, one for table semantic segmentation.
Cons: very tight separators labels: impossible structure recognition.

– 4 branches: architecture presented in Sec. 3.2.1.
Pros: avoids the tight separators labels. Independent segmentation of vertical and horizontal separators, useful in structure extraction (Sec. 4.6).
Cons: not very elegant: a lot of branches.

– 5 branches: each branch is dedicated to a binary element segmentation: text, illustrations, tables, vertical separators, horizontal separators.
Pros: simple: each element got it own branch. Avoid tight separation labels. Independent vertical and horizontal separators segmentation, useful in structure extraction (Sec. 4.6).
Cons: another new branch added...

Scores depending on the neural architecture, after learning on *SynDoc_ table* dataset (as presented in Sec. 3.1), can be seen on Tab. 9. The 'Separators detection' column corresponds to the ability for the network to effectively detect rows and columns separators. The *1 branch* and *3 branches* architectures could give less good scores than other architectures because, among other things, they are not able to detect the tables separators, as discussed in Sec. 5.2. The *4 branches* and *5 branches* architectures approximately give the same results, and are the only ones to correctly handle rows and columns separations detections. We decided to keep the *4 branches* architecture because of its less important number of branches.

Table 9: Table AP, Cell AP and WAvg.F1 for various networks architecture, with our current dataset

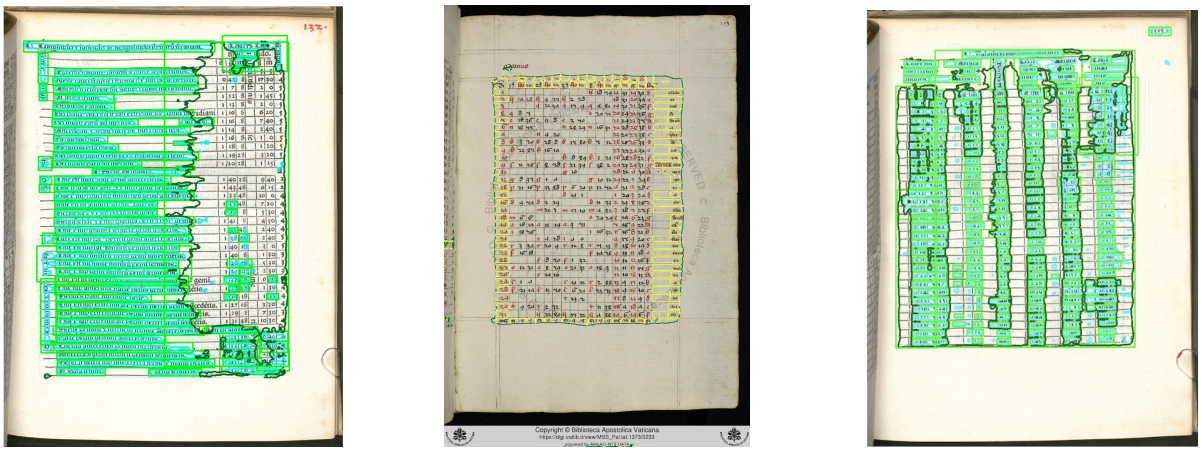
	Table AP (DISHAS)	Cell AP (DISHAS)	WAvg.F1 (cTDaR)	Separators detection
4 branches	**0.90**	**0.69**	**0.59**	Yes
5 branches	0.89	0.69	0.59	Yes
3 branches	0.87	0.66	0.55	No
1 branch	0.54	0.65	0.22	No

Some of these networks failures can be explained qualitatively.

First, let's explain the issue with tight separation labels, and why models with *3 branches* and *1 branch* don't detect tables separators. When both rows and columns separators detection is realized by a single branch, under a single label, some important mislabeling happens. Indeed, in the case of *compact* tables, the rows and columns separators are so close to each other that the networks learns to predict the entire table area under the separator label. This is true even

when the global synthetic dataset is very diversified, and presents many *loose* tables. A typical case of such a misleading segmentation can be seen in Fig. 11b.

Then, in the *1 branch* architecture, another issue consists in a bad tables interstitial pixels (others than cells and separators) detection. It leads to very bad tables detection scores, as Tab. 9 shows it. Detected tables are very fragmented: the network loses the notion of what a table is, as a whole. A typical case of such a bad segmentation can be seen in Fig. 11c.



(a) (b) (c)

Figure 11: Examples of bad detection, due to unadapted synthetic datasets or network architecture:
(a): Partial table region detection, on a typical *DISHAS* table. The network was trained on the original *SynDoc* dataset.
(b) Mislabeling of separators for *compact* tables, when rows and columns separators are learnt by a single branch. It is the case for the *1 branch* and *3 branches* architectures.
(c) Bad table interstitial pixels detection, when a *1 branch* network architecture is chosen.

6 Discussion and future works

This internship is extended by a part-time 'Contrat à Durée Déterminée' next year, so the presented work is to be continued.

In the near future, the goal would be to obtain better score on the cTDaR 2019 benchmark dataset for archival table detection, and to implement a table structure recognition algorithm compatible with the competition format, to really test our algorithm, and compare it with the community. The adaptation of our grid detection pipeline, presented in Sec. 4.6, can constitute a good basis for a more generic table structure recognition algorithm.

The transformation of the *DISHAS cell detection dataset* into a true table structure recognition dataset (as discussed in Sec. 2.2) would also be a good point. It would be another contribution to the community, and would allow the implementation of state of the art algorithms in structure recognition (such as [30]) on the DISHAS dataset.

More importantly, a key question is: do we stay with the Single Network approach presented in Sec. 3.2.1 ? It simultaneously detects tables and separators, is elegant, and could use ground truths of real data for fine tuning, but is very dependant on the visible presence of separation lines. Or, do we focus on the training of Two independent Networks, as presented in Sec. A.2, which would give better results in the case on non visible table lines, but would necessitate too complicated ground truths for real data fine tuning ? We should certainly analyze the frequency of cells without separation lines in the available archival datasets to answer such a question.

Generally speaking, we need to increase our competition scores, and to affirm our neural network conceptual choice.

7 Conclusion

We have created a new dataset for historical tables analysis, the *DISHAS tables* dataset (Sec. 2.2), and adapted Monnier et al. [1] synthetic generation process to cope with the diversity of real world tables (Sec. 3.1). We created a fully convolutional architecture adapted to table detection and structure recognition tasks (Sec. 3.2.1), while reflecting on an alternative, two-networks-based approach (Sec. A.2). We obtained performing results on the *DISHAS* dataset, according to various experiments (Sec. 4.3, Sec. 4.4.1, Sec. 4.5), and can still improve our results on the *cTDaR* table detection competition (Sec. 4.4.2). We quantitatively evaluated the relevance of our synthetic generations, and network architectures choices (Sec. 5). Besides, we developed a table structure recognition method which already permits transformation of table images into HTML tables (Sec. 4.6), and can be useful to the DISHAS team. But we still need to quantitatively evaluate it, and compare it to the community. Some nice things have been done, and some things must still be done. Hopefully, this project will in fine contribute to research in history of astronomy.

Appendices

A Two networks approach

In this section, we present an alternative neural network approach from the Single Network approach presented in Sec. 3.2.1. It is base on two independent networks, trained on their own tasks. A first network is trained to detect tables and cells. And a second regression network is trained to learn the signed distance to the rows and columns separators. At test time, the second network uses as input images the table regions detected by the first network.

A.1 Signed distance labeling

:

In the Two networks case, tables and cells are still binary labeled, but row and column separations are not. A continuous approach is used: the signed distance to the separators are given as labels to the regression network. Fig. 12 illustrates a typical table labeling in this case.

(a) (b) (c) (d) (e)

Figure 12: Tables components labeling, Two networks case: (a) input *loose* table with multiple rows spanning cells, (b) cell content (with border) label, (c) table label, (d) signed distance to the row separations, (e) signed distance to the column separations

A.2 Two networks architecture

In this case, a first network segments the documents text cells, illustrations and tables. The detected tables are used as input images for a second network. This second network regresses the signed distance to each row separator and each column separator in a table. The row and column separators thus correspond to the 0-level-sets of the detected function.

Technically, the two networks are U-Nets, as presented in Fig. 5, but with two final branches each. The first segmentation network performs multi-label segmentation, and its two branches are necessary: a first branch detects text and illustration, a second branch detects tables. The second regression network simultaneously infers the signed distance vertically and horizontally thanks to its two branches. The segmentation network is optimized with a sum of binary cross-entropy losses. The regression network is optimized with a sum of L2 losses. Because

of their different losses spaces, these two networks could not be efficiently mixed into a single network. The two networks are independently trained, on their own inputs and ground truths. At test time, they are chained to each other, to output the final segmentations (after a 0-level set detection on the signed distances predictions). An illustration of this test-time chaining situation can be seen on Fig. 13.

Fig. 12 illustrates the different network ground truths. For a shared input image SubFig. 12a, the first segmentation network will use Fig. 12b and Fig. 12c as ground truths for its first and second branch, and the second regression network will use Fig. 12d and Fig. 12e as ground truths for its first and second branch.

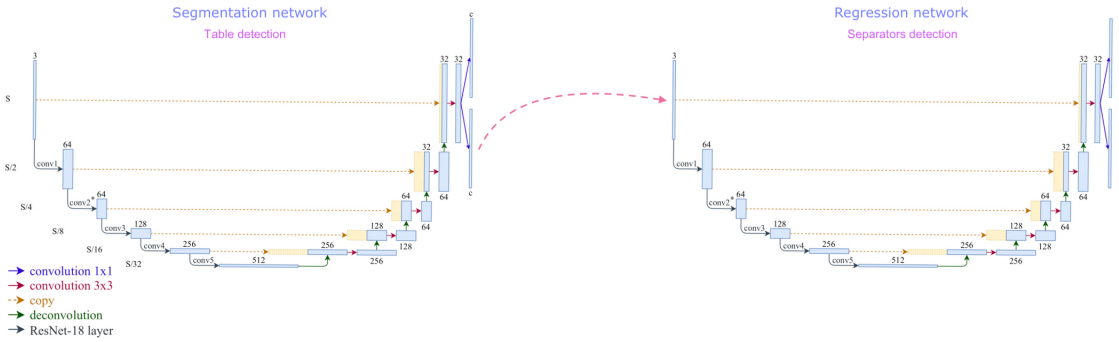


Figure 13: Two Networks approach: two U-Nets are independently trained, one to perform tables and cells segmentation, the other to perform signed distance to separators regression. At test time, regions detected as tables by the first segmentation network serve as input for the regression network, as symbolized by the pink arrow on the image.

Because it predicts signed distance functions on whole tables, this method should allow detection of rows and columns separators even when they are not visually drawn as lines, contrary to the Single Network approach. But this approach also has a big flaw: it is very complex to obtain signed distances ground truths from non-synthetic datasets. Consequently, fine-tuning this network on real-world datasets could be difficult.

N.B. In all the experiments, only the single network approach, as presented in Sec. 3.2.1 has been implemented. This two networks approach has only been studied qualitatively. Nevertheless, a comparison of tables and cells detection scores with or without simultaneous separators detection can be seen in Sec. 5.2. They tend to prove that a parallel detection of separators, cells and tables improve tables and cells detection scores.

References

[1] Tom Monnier and Mathieu Aubry. docextractor: An off-the-shelf historical document element extraction. In *ICFHR*, 2020.

[2] L. Gao, Y. Huang, H. Déjean, J. Meunier, Q. Yan, Y. Fang, F. Kleber, and E. Lang. Icdar 2019 competition on table detection and recognition (ctdar). In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1510–1515, 2019.

[3] Pallavi Pyreddy and W. Bruce Croft. Tintin: A system for retrieval in text tables. In *Proceedings of the Second ACM International Conference on Digital Libraries*, DL '97, page 193–200, New York, NY, USA, 1997. Association for Computing Machinery.

[4] F. Cesarini, S. Marinai, L. Sarti, and G. Soda. Trainable table location in document images. In *Object recognition supported by user interaction for service robots*, volume 3, pages 236–240 vol.3, 2002.

[5] Thotreingam Kasar, Philippine Barlas, Sebastien Adam, Clement Chatelain, and Thierry Paquet. Learning to detect tables in scanned document images using line information. pages 1185–1189, 08 2013.

[6] A. C. e. Silva. Learning rich hidden markov models in document analysis: Table location. In *2009 10th International Conference on Document Analysis and Recognition*, pages 843–847, 2009.

[7] Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C. Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural network, 2017.

[8] S. Ares Oliveira, B. Seguin, and F. Kaplan. dhsegment: A generic deep-learning approach for document segmentation. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 7–12, 2018.

[9] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1162–1167, 2017.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 06 2015.

[11] Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait. Table detection using deep learning. 09 2017.

[12] S. Siddiqui, M. Malik, S. Agne, A. Dengel, and S. Ahmed. Decnt: Deep deformable cnn for table detection. *IEEE Access*, 6:74151–74161, 2018.

[13] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017.

[14] I. Kavasidis, S. Palazzo, C. Spampinato, C. Pino, D. Giordano, D. Giuffrida, and P. Messina. A saliency-based convolutional neural network for table and chart detection in digitized documents, 2018.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[16] Y. Huang, Q. Yan, Y. Li, Y. Chen, X. Wang, L. Gao, and Z. Tang. A yolo-based table detection method. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 813–818, 2019.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[19] Yalin Wang, Ihsin Phillips, and Robert Haralick. Table structure understanding and its performance evaluation. *Pattern Recognition*, 37:1479–1497, 07 2004.

[20] Thomas Kieninger and Andreas Dengel. The t-recs table recognition and analysis system. volume 1655, pages 255–269, 11 1998.

[21] Shubham Paliwal, Vishwanath D, Rohit Rahul, Monika Sharma, and Lovekesh Vig. Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images, 2020.

[22] M. Göbel, T. Hassan, E. Oro, and G. Orsi. Icdar 2013 table competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1449–1453, 2013.

[23] C. Tensmeyer, V. I. Morariu, B. Price, S. Cohen, and T. Martinez. Deep splitting and merging for table structure decomposition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 114–121, 2019.

[24] Devashish Prasad, Ayan Gadpal, Kshitij Kapadni, Manish Visave, and Kavita Sultanpure. Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents, 2020.

[25] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation, 2019.

[26] N. L. Vine, M. Zeigenfuse, and M. Rowan. Extracting tables from documents using conditional generative adversarial networks and genetic algorithms. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[27] Shah Rukh Qasim, Hassan Mahmood, and Faisal Shafait. Rethinking table recognition using graph neural networks, 2019.

[28] Yiren Li, Zheng Huang, Junchi Yan, Yi Zhou, Fan Ye, and Xianhui Liu. Gfte: Graph-based financial table extraction, 2020.

[29] Zewen Chi, Heyan Huang, Heng-Da Xu, Houjin Yu, Wanxuan Yin, and Xian-Ling Mao. Complicated table structure recognition, 2019.

[30] C V Jawahar Sachin Raja, Ajoy Mondal. Table structure recognition using top-down and bottom-up cues, 2020.

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[32] Asif Shahab, Faisal Shafait, Thomas Kieninger, and Andreas Dengel. An open approach towards the benchmarking of table structure recognition systems. pages 113–120, 06 2010.

[33] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. Tablebank: A benchmark dataset for table detection and recognition, 2019.

[34] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: data, model, and evaluation, 2019.

[35] Abhishek Dutta and Andrew Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, New York, NY, USA, 2019. ACM.

[36] Yalin Wangt, I. T. Phillipst, and R. Haralick. Automatic table ground truth generation and a background-analysis-based table structure extraction method. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 528–532, 2001.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[38] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. 07 2016.

[39] 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*, pages 248–255, 2009.

[40] Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010.

[41] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[42] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[43] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.