Work report on tableTranscriber: an automatic pipeline for astronomical tables transcription

Tristan Dot

June 2022

Here is a work report for *tableTranscriber*, an end-to-end method for handwritten tables transcription. Combining a table structure recognition method, based on an image segmentation model, and a Handwritten Text Recognition (HTR) algorithm, scans of handwritten tables can be effectively, and automatically transcribed into numerical tables. This work aims specifically at transcribing historical tables, which vary greatly in their layouts and writing style. Astronomical tables analysis constitutes the key application of *tableTranscriber*.

Nota bene: This document is not a scientific paper, but simply a technical work report concerning tableTranscriber, a table transcription method developed at Paris Observatory, as part of the ERC-funded ALFA research project[1].

1 Introduction: research context and contributions

In the last few years, while digital humanities grew in interest, more and more tools emerged in order to automatically segment, transcribe and analyze digitized historical documents.
History of astronomy constitutes a field with potential major applications for such automatic methods. Indeed, a great quantity of astronomical manuscripts are composed not only of text and diagrams, but of a very large amount of tables. During the middle ages and the modern period, the position of the Sun, the Moon, the planets and the stars were predicted thanks to mathematical tables. To compute these tables, astronomers used a combination of auxiliary calculation tables, that were based on trigonometric tables. All these tables constitute a unique heritage for the understanding of old hand calculation methods. Their quantitative study is essential both for the history of astronomy and for the history of mathematics, but their manual transcription is extremely time consuming. To study them in large scale, and to consequently open up the research in history of astronomy, we developed a new method for automatic table transcription: *tableTranscriber*.
Based on the historical document segmentation approach *docExtractor* [1], *tableTranscriber* combines a table structure recognition method and a HTR method in order to efficiently transcribe scans of historical tables. To go with our method and to better evaluate generalization, we introduce a new dataset for historical tables detection, composed of medieval astronomical tables: *DishasTables*. Moreover, we implemented *tableTranscriber* as a user-friendly web application, that can be easily used by all researchers in history of astronomy – leading to significant time savings in the analysis of tables corpora.

[1] https://alfa.hypotheses.org/

2 Related works

2.1 Table detection

Before the advent of deep learning, traditional approaches for table detection were based on heuristics, metadata, predefined tables templates [2, 3], or hand-crafted features combined with classifiers [4]. More recently, deep learning models of object detection and semantic segmentation brought significant performance improvements for such tasks. These models have been efficiently used for documents layouts general understanding [1, 5, 6], but also, more specifically, for table detection. In 2017, DeepDeSRT [7] employed the object detection network Faster R-CNN [8] and transfer learning to locate tables on images, and achieved particularly performing results. Kavasidis et al. [9] employed Conditional Random Fields (CRFs) on saliency maps extracted by a Fully Convolutional Network (FCN) [10] to detect tables, and different types of charts. More recently, Huang et al. [11] proposed a table detection network based on an adapted YOLOv3 [12] network. In a revealing way, nearly all participants of the ICDAR 2019 Table competition [13] used, for the table detection task, one of the following models: Faster R-CNN, YOLO, FCN, or U-Net [14].

2.2 Table structure recognition

The first tables recognition methods were based on PDF metadata, hand-crafted features and heuristic [15], which made strong assumptions concerning tables layouts. DeepDeSRT [7] was among the first deep learning approach to deal with table structure analysis. In this method, after a global table detection step, realized by a Faster R-CNN network, the tables rows and columns are detected thanks to a FCN network. The processed images are previously stretched, vertically and horizontally, to facilitate the tables rows and columns separation. Another method –closer to our own approach–, TableNet [16], is based on a single FCN network, divided in two branches, to detect tables, and each one of their columns. Recently, many works have preferred a graph-based formulation of the problem [17], as illustrated Qasim et al. [18], who used graph neural networks to model table-level associativity between words. In the same idea, Raja et al. [19] mixed, in a single network, a top-down table cells detection, performed thanks to a Mask R-CNN network [20], and a bottom-up structure recognition process, built on top of graphical model [18].

2.3 HTR

Off-line HTR methods, as a sub-branch of Optical Character Recognition (OCR) methods, were initially based on language models and Hidden Markov Models (HMMs) combined with Gaussian Mixtures. Since a few years, the most common approaches in HTR are based on what are called Convolutional Recurrent Neural Networks (or CRNN, which consist of a Recurrent Neural Networks, such as a LSTM, on top of CNN) combined with a Connectionist Temporal Classification (CTC) loss, as proposed by Shi et. al. in 2015 [21]. More recently, some authors got rid of the CTC loss, and stacked a RNN with attention mechanisms on top of a CNN in order to perform in the wild OCR [22, 23]. Concerning, more specifically, HTR on historical documents, various international projects [1] have had a leading role in the creation of new, large datasets, thought by and for researchers in humanities. Such historical datasets were used in the last HTR competitions organized by the ICDAR and ICFHR conferences (as summarized by Sanchez et. al. [24]).

[1] Such as: READ [https://readcoop.eu/], and Transkriptorium [http://www.transkriptorium.com/].

3 Dataset

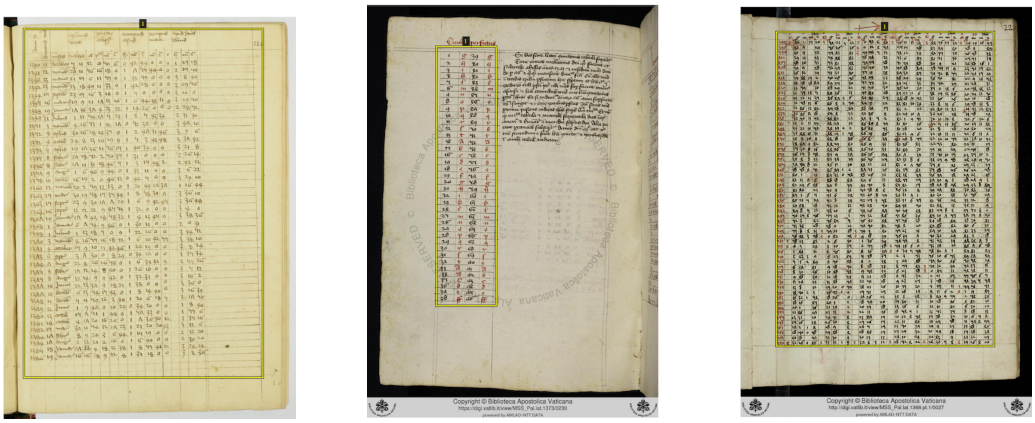
3.1 Context

tableTranscriber, though applicable to any handwritten tables (with Latin script and Hindu Arabic numerals), has been specifically developed for medieval astronomical tables. More precisely, in Paris Observatory, the ALFA team [1] is dedicated to the study of Alfonsine Astronomy, which flourished in Europe from the second half of the 13th century to the middle of the 16th century. Alfonsine tables relied on the tradition of Ptolemy's theory and tables (1st c. CE), based on spherical trigonometry modified over the centuries in the Muslim world, from which the Latin scholars inherited computational astronomy during the 12th and 13th centuries. These tables constituted a major European scientific achievement. For more than two centuries in Europe, they were the main tool for astronomical computations, allowing the emergence of thinkers like Regiomontanus or Copernicus.
To study astronomical tables, quantitatively and at large scale, the ALFA team has developed an online platform dedicated to their numerical edition and analysis: the DISHAS (Digital Information System for the History of Astral Sciences)[2] project. *tableTranscriber* aims at scaling up this quantitative analysis process, by automatically transcribing astronomical tables.

3.2 Dataset presentation

To better evaluate generalization, to fill a lack of diversity in previous tables datasets, and because of the scientific interest of Alfonsine Tables, we introduce a new dataset for historical tables detection: *DishasTables*. This new table detection dataset is composed of ∼310 images of folios containing tables, coming from 10 Alfonsine manuscripts. Polygons surrounding tables were annotated using VGG Image Annotator [25]. Tables regions are constituted by cells (empty or containing numbers) structured in a grid layout.
Although dating from various centuries[3], the tables in *DishasTables* share many common characteristics in terms of layouts. They are composed of many (several hundreds) numerical cells, and present very tight rows and columns. Sometimes, entire columns are written in colored ink (to help with astronomical 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, as Fig. 1 can give an idea. 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 [13], which is currently the only public dataset dealing with historical table analysis. It constitutes, consequently, a valuable contribution.

[1] https://alfa.hypotheses.org/members
[2] https://dishas.obspm.fr/
[3] To get a look at the diversity of tables studied in DISHAS, please visit: https://dishas.obspm.fr/js/mirador-alfa/index.html



(a) (b) (c)

Figure 1: Examples from DishasTables dataset, with ground truth indicated in yellow lines.

4 Approach

Our global approach combines a segmentation model, trained on a synthetic tables dataset, and a HTR model, trained on images labeled manually, in order to perform table structure recognition and automatic transcription.

4.1 Table segmentation

4.1.1 Synthetic tables dataset

A single real-world dataset exists for historical table detection and structure recognition: the cTDaR 2019 archival dataset [13]. Given its limited size and its lack of tables diversity, it cannot be used as a general training dataset for tables structure recognition. To cope with this lack, we decided to generate a large synthetic tables dataset, adapted from the *SynDoc* dataset (introduced by Monnier et. al. in 2020 [1]).
SynDoc simulates a large variety of historical-like documents, created by randomly mixing various pages backgrounds, pages layouts, visual degradations and elements types. It uses a large database of illustrations and fonts downloaded from the web to increase its diversity. Its textual elements are 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 therefore enhanced this table generation process, to create realistic and varied synthetic tables – some of which are thought to imitate, from a layout point of view, medieval astronomical tables.

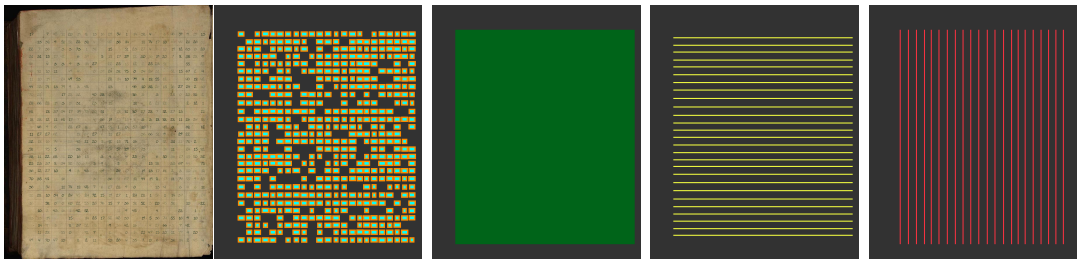
Table generation
The generation of our synthetic tables dataset, *SynTables*, is divided into various randomized steps. First, a synthetic document is created (as described by [1]), with a high (boosted) probability of containing a table element. If it is the case, the table layout can be *compact*, or *loose*, and can contain, or not, one of the following specific features: rows and columns separators, empty rows or columns, colored rows or columns, multiple spanning cells, and textual legends. Additionally, to increase the diversity of the tables, some cells contents are randomly deleted, random opacity levels are fixed for the rows and columns separators, as well as for the cells contents. If the synthesized document is a double page document, the table can alternatively

spread, or not, among the two pages.
Very diverse tables are then generated. The *compact* layout is thought to mimic the *DishasTables* layouts (Sec. 3): *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. [13]): cells are bigger, and can contain numbers or words.
Some *SynTables* examples can be seen on Fig. 2. We generated 10K *SynTables* elements, with their corresponding labeling, in order to train our segmentation network.

(a) (b) (c)

Figure 2: Some SynTables examples: (a) & (b) *compact* tables, (c) *loose* table.

Table labeling
Each table semantic element is linked to a specific label. Compared to *SynDoc* [1], we created three new specific tables labels: global tables areas, tables rows separators, and tables columns separators, which complement the tables cells content (with borders) label. Fig. 3 illustrates a typical table labeling, in the loose table layout case. For our 10K *SynTables* elements, we then generated $4 \times 10 = 40K$ corresponding labeling images.



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

Figure 3: Tables components labeling: (a) input loose table with multiple rows cells, (b) cell content (with border) label, (c) global table label, (d) row separations label, (e) columns separations label

4.1.2 Segmentation network

Because of the very close proximity between some of the elements we want to detect (especially concerning compact tables: cells, rows and columns separators are very close to each other), and of their high number of occurrences, object detection networks, such as Faster R-CNNs [8], are not the best choice for table structure recognition. A fine-grained image segmentation performed by a FCN architecture appears to be more suited for the task (as inspired by [7, 16], which use a FCN to detect tables' columns).
We decided to use a single fully convolutional network to simultaneously detect the tables and their semantic components (similarly to [16]). Our network is a U-Net [14] (as [1, 6]), with a ResNet [26] backbone encoder. We used the same ResNet-18 encoder as Monnier et. al. [1], with 2-strided 3x3 convolutional layer instead of maxpooling, in order to perform efficiently small text lines detection.

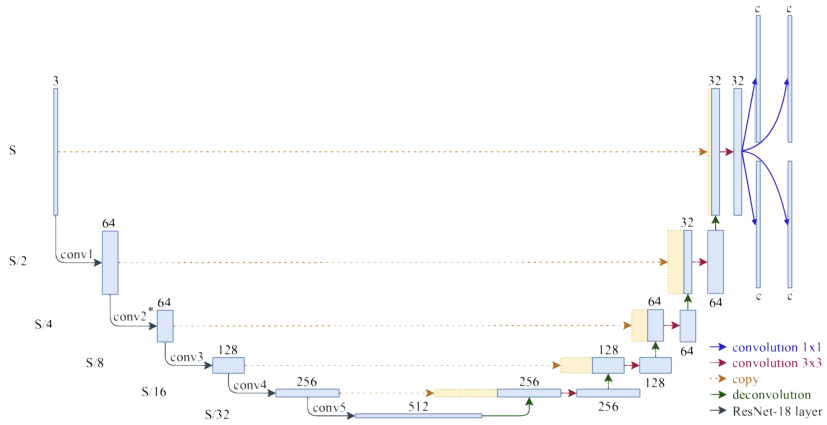


Figure 4: Our network architecture: U-Net with four final branches to perform multi-label segmentation.

To perform multi-label segmentation –since table pixels can belong to different classes simultaneously– we added four final branches to our network, as illustrated by Fig. 4. One branch is dedicated to illustration and text (including text cell) segmentation. A second one is dedicated to global tables segmentation, another one to rows separators segmentation, and a last one to columns separators segmentation. An example of the ground truth images fed to each branch of the network is illustrated in Fig. 3. The network is optimized according to a sum of binary cross-entropy losses (one binary cross-entropy loss per final branch).

Pre-processing, post-processing, training details:
Concerning pre-processing, input images are resized so that their larger side is 1280 pixels. After segmentation, detected connected components with small areas are filtered before output, depending on class-specific ratio thresholds – as inspired by Monnier et. al. [1].
During the training, 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 [27] with a momentum of 0.1 instead of Batch Normalization. We use ImageNet [28] pretrained weights for the encoder, and Xavier initialization [29] for the other convolutional layers. We train for 50 epochs with Adam optimizer [30] 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 HTR

4.2.1 Dataset

We constituted, manually and thanks to experts in medieval paleography, a dataset ∼1000 cropped cells with their corresponding annotations. The cells were extracted from three different manuscripts, from the same period, but written by distinct hands. These cells are only composed of numbers, corresponding to astronomical data calculations.

4.2.2 Network

In order to process HTR on the cropped table cells, we implemented a CRNN [21]. Our method is based on the work and code of Virginie Loison and Xiwei Hu[1], adapted in order to easily transcribe numerical cells, by reducing the dictionary of transcribed signs to eleven elements (i.e. the ten numbers and the special blank symbol). Since astronomical tables are nearly only composed of numbers, such a reduction of the domain of translation seemed natural, and helped us to achieve very good HTR results with few annotations. Even though classification of images of numbers is one of the oldest tasks of CNNs, the use of a CRNN was in this case necessary, in order to transcribe numbers of unknown length.
A standard ResNet-18 [26] is used as feature extractor,linked to a LSTM [31] for labelling, with a CTC loss [32].

Pre-processing, post-processing, training details:
Concerning pre-processing, all images are converted to grey levels, normalized, contrast enhanced, and resized so that their width is 64 pixels. No further post-processing is implemented after the CTC decoding operation. During the training, on-the-fly random affine transformations are applied to the input data. We process 16 samples per batch and use Batch Normalization. We train the CRNN from scratch, for 400 epochs, with RMSprop optimizer. The learning rate is initially set to 0.001 and divided by 10 all 100 epochs.

4.3 Table transcription

Our global pipeline, for table structure recognition and then content transcription, corresponds to the following steps:

1. Input of a manuscript image.
2. Detection of the table(s), cells, rows separators and columns separators, as described in Sec. 4.1.
3. Rotation of the image according to the average angle of all the table(s) columns separators.
4. Post-processing cleaning: deletion of the separators which are too close from each other.
5. 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. 6 for an illustration of grid creations on various tables layouts.
6. Division of neighbouring merged cells according to the grid definition.
7. Indexation of each cell depending on its maximum and minimum coordinates, and on the table grid.

[1]Please see their GitHub: https://github.com/vloison/Handwritten_Text_Recognition

8. Transcription of the content of each cell, according to the HTR method described in Sec. 4.2.

9. Final digital table output, that can then be analyzed and manipulated in the wanted format (XML, CSV, HTML).

Fig. 5 shows an illustration of such a transcription, from a manuscript image containing an astronomical table, to the numerical, fully transcribed table.

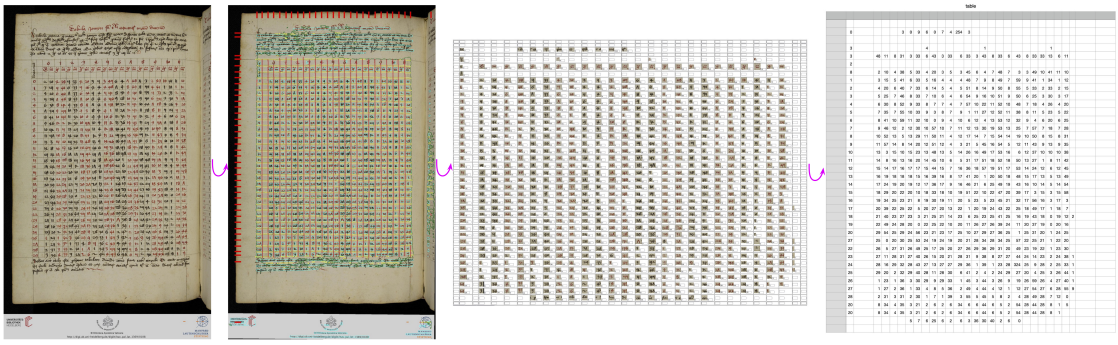
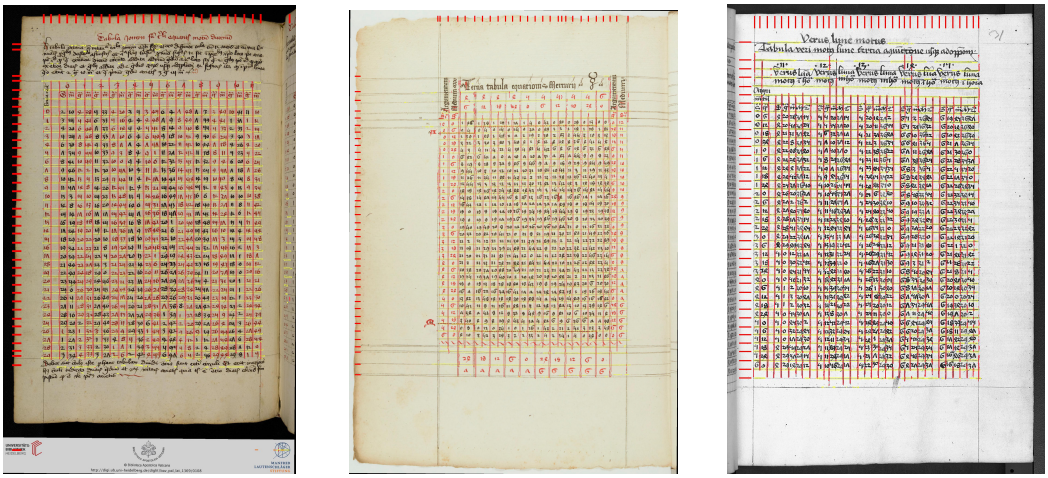


Figure 5: Conversion pipeline, from an image of table to a digital, fully transcribed table.



(a) (b) (c)

Figure 6: Grid detection on various input images. Columns separators are labeled in red, rows 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 Experiments

Our method, *tableTranscriber*, has been evaluated, quantitatively and qualitatively, on medieval astronomical tables. We first tested its ability to detect the presence of tables on manuscript

pages, before evaluating its table detection performance on the *DishasTables* dataset. Concerning the table structure recognition and content transcription tasks, we evaluated them qualitatively, with researchers in history of astronomy, and developed a web application –that can be used by researchers– in order to automatically transcribe large astronomical tables.

5.1 Evaluation

Table presence detection
We evaluated the capacity, for our segmentation method (presented in Sec. 4.1), to detect the presence of tables on manuscript pages. To do so, we used an internal dataset composed of 7450 manuscript pages, coming from 17 different medieval manuscripts, which were binary annotated as containing, or not, (at least) a table. On this task, we obtained an average precision of 96%, without any fine tuning on real world images.

Table detection
Concerning the table area detection task, we evaluated the performance of our segmentation network on the *DishasTables* dataset (introduced in Sec. 3.2). We obtained, without any fine tuning, a mAP[1] of 0.90. As complementary information, Tab. 1 shows the table AP score on the *DishasTables* dataset, for IoU thresholds greater or equal than 0.5.

Table 1: Table AP for various IoU thresholds, without fine tuning, on the *DishasTables* dataset

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

5.2 *tableTranscriber* web application

We did not evaluate the table structure recognition and content transcription tasks of *tableTranscriber* quantitatively, but we extensively did so qualitatively, with the researchers of the ALFAM team. On a daily basis, the use of *tableTranscriber* allows to save ∼90% of time in the transcription of astronomical tables[2].
To ensure that any researcher involved in astronomical table analysis can benefit from this tool, we implemented *tableTranscriber* into a –user friendly– web application, created in Flask, as illustrated in Fig. 7. Researchers now just need to upload the images of tables they are working on, and *tableTranscriber* will automatically recognize their structures and transcribe their content (according to the process described in Sec. 4.3).

[1] We considered the mean Average Precision –mAP– metric as defined in the PASCAL VOC 2012 competition [33], with an IoU threshold of 0.5.
[2] For a table containing ∼600 cells, the transcription time is reduced from 45 to 5 minutes.

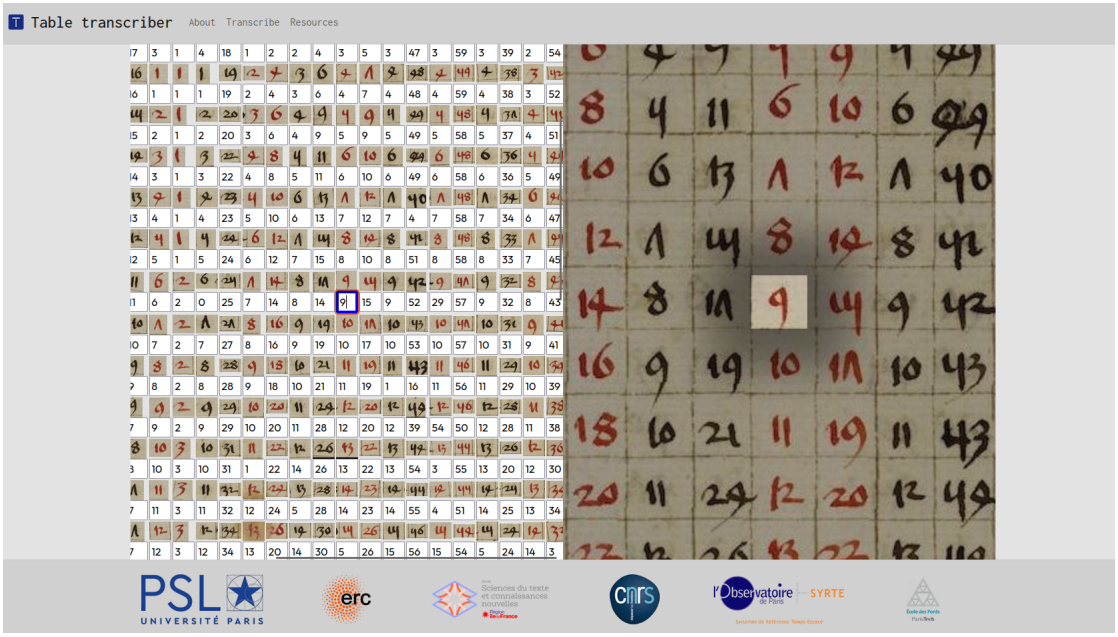


Figure 7: *tableTranscriber* web application screenshot

6 Conclusion

We adapted a table segmentation method, and linked it to a HTR network, in order to efficiently transcribe historical tables. Medieval astronomical tables, which are very dense and contain a great deal of numerical cells, constitute the key application of our method, and the main elements of our new dataset *DishasTables*. Concretely, and on a daily basis, *tableTranscriber* –implemeted as a user-friend web application– allows to save ∼90% of transcription time for researchers in history of astronomy.

tableTranscriber is, in our opinion, a good example of how machine learning can be used in humanities. By relying on recent computer vision methods, we developed a tool adapted to a precise and justified need (the analysis of astronomical tables), which is easy to use and flexible. The outputs of *tableTranscriber* can be exported in multiple formats (XML, CSV, HTML), before being corrected and analyzed. In the future, we would like to develop an even more interactive web interface, allowing users to correct potential structure recognition errors before the HTR transcription step. This interactivity, between researchers in humanities and machine learning methods –thanks to flexible visual interfaces–, is essential in order to create useful, efficient new tools in digital humanities.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

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

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

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

[20] 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.

[21] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. 2015.

[22] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. 2016.

[23] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. 2018.

[24] Joan Andreu SÃ¡nchez, VerÃ³nica Romero, Alejandro H. Toselli, Mauricio Villegas, and Enrique Vidal. A set of benchmarks for handwritten text recognition on historical documents. *Pattern Recognition*, 94:122–134, 2019.

[25] 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.

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

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

[28] 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.

[29] 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.

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

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[32] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. page 369–376, 2006.

[33] 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.