Annotation-Free Character Detection in Historical Vietnamese Stele Images

Anna Scius-Bertrand[1,2] (✉), Michael Jungo[1], Beat Wolf[1], Andreas Fischer[1,3], and Marc Bui[2]

[1] iCoSys, University of Applied Sciences and Arts Western Switzerland
{firstname.lastname}@hefr.ch
[2] Ecole Pratique des Hautes Etudes, PSL, Paris, France
{firstname.lastname}@ephe.sorbonne.fr
[3] DIVA, University of Fribourg, Switzerland

Abstract. Images of Historical Vietnamese stone engravings provide historians with a unique opportunity to study the past of the country. However, due to the large heterogeneity of thousands of images regarding both the text foreground and the stone background, it is difficult to use automatic document analysis methods for supporting manual examination, especially with a view to the labeling effort needed for training machine learning systems. In this paper, we present a method for finding the location of Chu Nom characters in the main text of the steles without the need of any human annotation. Using self-calibration, fully convolutional object detection methods trained on printed characters are successfully adapted to the handwritten image collection. The achieved detection results are promising for subsequent document analysis tasks, such as keyword spotting or transcription.

Keywords: Object Detection · Self-Calibration · FCOS · YOLO · Historical Vietnamese Steles · Chu Nom Characters.

1 Introduction

Centuries-old stone engravings in man-sized steles in Vietnam carry invaluable information for historians, who are interested to learn more about the life of common people in the villages [14]. Over the past decades, thousands of steles have been copied to paper, then photographed, thus creating a comprehensive collection of digital documents [15,16] that facilitate the examination of the engravings, as currently being done in the ERC project VIETNAMICA.[4]. However, due to the sheer size of the document collection, which contains tens of thousands of images, manual inspection is difficult and time-consuming.

Automatic document image analysis would be of tremendous help for the historians. Even if an automatic transcription of the engraved Chu Nom characters is not immediately feasible, knowing more about the general layout structure,

[4] https://vietnamica.hypotheses.org

number and size of the characters, or spotting particular keywords, can be helpful for browsing and exploring the image collection.

Over the past decade, methods for historical document analysis have seen a strong progress [6], driven in large part by the advent of robust convolutional approaches to image enhancement [21], layout analysis [2], keyword spotting [22], and automatic transcription [3], to name just a few. However, unlike earlier more heuristic methods to solve these problems, one of the main constraints is the need for human-annotated samples to drive the machine learning process.

In the case of historical Vietnamese steles, a large amount of labeled samples would be needed to cover all the different layouts, character shapes, text resolutions, stone backgrounds, fissures, ornaments, etc. that are different from one stele to another. For obtaining an impression of the variety of steles, we refer the reader to Figure 7 in the experimental section. Annotations are difficult to obtain, especially for rare Nom characters that require expert knowledge.

The ideal properties of a document analysis method in this context would be that it avoids preprocessing, does not rely on an explicit segmentation prior to recognition, and does not require any human annotation. In this paper, we explore the feasibility of using fully convolutional object detection methods to achieve these goals, aiming at transferring knowledge from printed Nom characters directly to the stone engravings.

Such an approach is encouraged by the findings of Yang et al. [27], showing that object detection can be used successfully to detect Chinese characters in historical documents. Furthermore, specifically targeting Nom characters, Nguyen et al. [13] report strong results for character extraction and recognition in historical manuscripts using convolutional neural networks, i.e. a U-Net architecture [19] to perform semantic segmentation, followed by a watershed segmentation of character regions, and finally a coarse-to-fine recognition with different convolutional models. An interesting aspect of their work, which relates to our goals, is that they were able to train the U-Net in large parts with printed characters, which were then fine-tuned on real manuscript pages. In previous work, a U-Net architecture has also been applied by Scius-Bertrand et al. [20] in the context of Vietnamese steles, together with a seam carving algorithm, to first detect pixels belonging to the main text and then segmenting them reliably into columns. However, further denoising, detection, and recognition algorithms would be needed to locate individual characters.

In the present paper, we investigate object detection methods trained on printed Nom characters to find the location of main text characters on the steles directly. Without using human annotations, we control the behavior of the convolutional neural networks by means of unsupervised layout analysis and a self-calibration step, for allowing the network to take the different stele backgrounds into account.[5] The proposed approach is studied for two object detection methods, namely YOLO [17] and FCOS [24], on a dataset of 2,036 stele images

[5] Readers interested in the source code and the dataset are referred to our GitHub repository https://github.com/asciusb/annotationfree.

containing hundreds of thousands of characters. 65 stele images have been manually annotated for the purpose of performance evaluation.

In the following, we describe the dataset, provide more details about object detection, introduce the proposed layout analysis and self-calibration methods, present the experimental results, and draw some conclusions.

2 Dataset of Historical Vietnamese Steles

The collection of stele inscriptions includes about 40,000 unique copies collected from 1910 to 1954 by the French School of the Far East (EFEO) and, since 1995, by the Han-Nom Institute [14]. The majority of the steles are from the 17th to 19th century. The steles were mainly located in the north of Vietnam, where production and trade made the farmers rich enough to have steles, such as in the rice plains and along river banks. In terms of content, 90% of the steles are at the village or quarter level. They are often internal village or family documents. The steles are a mine of information on the social, cultural, economic, religious and linguistic life of the villages.

Reading this corpus involves many challenges. As the steles are often outside, they have suffered damage due to the weather or the deterioration of the stone (impact, cracks, etc.). The layout of the steles is also very diverse. The borders are more or less wide and more or less ornate. The title can be engraved or can appear in relief. The text columns are irregular and can be divided into two columns. All these elements make automation more complex.

To test our system, we had access to 2,036 photographs of stele stamps used in the VIETNAMICA project. A stamping is a faithful reproduction of a stele thanks to a roll of ink passed over the whole of a sheet glued together with banana juice. The engraved characters are white and those in relief are black. The only preprocessing in this paper consists of inverting the color of the steles, such that the characters of the main text appear in black. For the training data we have generated synthetic documents inspired by the steles, namely text in Chu Nom organised in columns. For printing Nom characters, we rely on a font, which was kindly provided by the Vietnamese Nom Preservation Foundation.[6] The annotated test set of 65 stele images contains on average 375 characters per stele, with a minimum of 20 characters and a maximum of 883.

3 Convolutional Character Detection

3.1 You Only Look Once (YOLO)

YOLO [17] (You Only Look Once) was the pioneer of one-stage object detection by showing competitive results with a single convolutional neural network. Its simplicity and efficiency made it one of the most popular object detection models and inspired further improvements of YOLO itself, by addressing its shortcomings, as well as other one-stage object detectors.

[6] http://www.nomfoundation.org

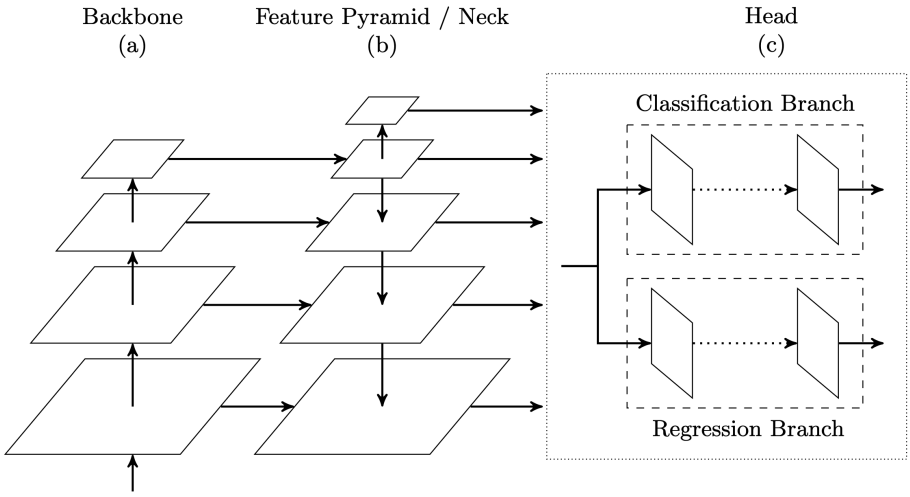


Fig. 1: One-stage object detection models are comprised of three major parts: (a) the backbone extracts features from the input image, (b) a feature pyramid combines the extracted feature maps to enrich the features across different scales, which are then processed individually by the head (c), where two separate branches are used for the classification and the bounding box regression respectively. The specific implementation for all three parts can be freely chosen and some models may include additional modifications, but the overall architecture remains.

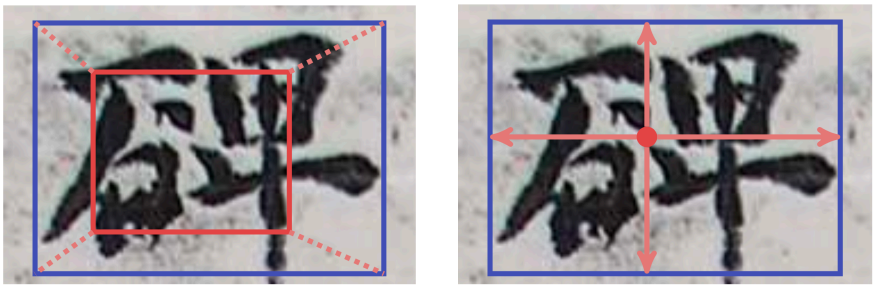
The model has been improved iteratively with rather small changes and in the latest version, YOLOv4 [1], the focus put on optimizing the current architecture by comparing various existing building blocks in order to find the best possible configuration. As a result, YOLOv4 consists of the following: A variation of the Cross Stage Partial Network (CSPNet)[25] is used as the backbone, a modified Path Aggregation Network (PAN)[12] as the neck and the existing anchor based head of YOLOv3 [5]. Figure 1 illustrates these system components.

We have decided to use YOLOv5 [8]. Contrary to its name, it is not the successor of YOLOv4, but rather a port of YOLOv3 to PyTorch, which has been improved independently from YOLOv4, but then also adopted most improvements introduced by YOLOv4.

3.2 Fully Convolutional One-Stage Object Detection (FCOS)

FCOS [24] is a fully convolutional one-stage and anchor-free object detector. The primary difference of the underlying method compared to YOLO is the omission of anchors, which serve as initial bounding boxes that are further refined in order to get tightly fitting bounding boxes around the objects (Figure 2a). In contrast

to this, FCOS uses a single point to span up a bounding box by predicting the distances towards each of the four sides (Figure 2b).



(a) YOLO (b) FCOS

Fig. 2: Difference in bounding box prediction: (a) YOLO refines a given anchor box (b) FCOS predicts the distances to all four sides from a point

The backbone is most commonly any variation of ResNet [7], but any desired CNN can be used in its place. We have decided to use EfficientNet [23] because it has shown to provide strong results with great performance. One key change to the EfficientNet model was replacing Batch Normalization with Group Normalization [26] because of having to use smaller batch sizes, due to needing larger input images for this task, where Batch Normalization is not as effective.

A particularly important aspect of the architecture for the bounding box localization in FCOS is the Feature Pyramid Network (FPN)[9], which generates multiple feature maps at different scales, five to be exact, where later feature maps flow back into earlier stages. This feature pyramid results in semantically stronger feature maps across all levels, making the detection more robust to various scales.

Another aspect that made one-stage and anchor-free object detectors more viable, was the selection of loss functions. The Focal Loss [10] is used as the classification loss to address the imbalance between foreground and background. As one-stage detectors create dense predictions, i.e. they predict every single point in a feature map, there are disproportionally more points that are considered background than foreground. Having such a massive imbalance, the model may easily be swayed towards predicting background when even the smallest uncertainty exists, as it is the much more likely outcome.

$$\mathcal{L}_{cls} = FL(p, y) = \begin{cases} -\alpha(1-p)^{\gamma}\log(p) & \text{if } y = 1 \text{ (foreground)} \\ -(1-\alpha)p^{\gamma}\log(1-p) & \text{otherwise (background)} \end{cases} \quad (1)$$

The Focal Loss (Equation 1) is an extension of the cross entropy loss by adding the factors $(1-p)^{\gamma}$ and p^{γ} respectively, which are designed to reduce the loss

for easily classified examples, putting more emphasis on misclassified ones, when $\gamma > 0$. Setting $\gamma = 0$ makes the Focal Loss equivalent to the α-balanced cross entropy loss. $\alpha \in [0, 1]$ is a weighting factor to also address the class imbalance, but only to balance the importance of positive/negative examples, whereas the Focal Loss also addresses the importance between easy/hard examples.

For the bounding box regression, the Generalized Intersection over Union Loss (GIoU)[18] is used. The regular Intersection over Union (IoU) is not well suited as a loss, since non-overlapping boxes always have an IoU of zero, regardless of how far apart they are, making the optimization infeasible. This shortcoming is addressed by the GIoU with the introduction of an additional term:

$$\mathcal{L}_{reg} = 1 - GIoU = 1 - \left(\frac{|A \cap B|}{|A \cup B|} - \frac{|C \setminus (A \cup B)|}{|C|} \right) \tag{2}$$

where A and B are two bounding boxes and C is the smallest enclosing bounding box, i.e. the smallest bounding box to contain both A and B. The goal is to have an enclosing bounding box C with the smallest possible area. In other words, the empty area between the A and B should be minimized.

Similar to the classification imbalance, multiple points may be used as the starting point for the bounding box regression, where some of them can be harmful for the model. In order to alleviate that problem, the Adaptive Training Sample Selection (ATSS)[28] has been employed, where only the $k = 9$ best candidates are selected per FPN level for each ground truth bounding box, based on the centerness, because starting points closer to the center of the bounding box are favorable and produce the best results.

4 Self-Calibration

In this section, we introduce the proposed calibration method, which allows to adapt a character detection system from printed characters to the handwritten image collection. An overview of the method is provided first, followed by a description of the individual components.

4.1 Method overview

Figure 3 provides an overview of the self-calibration. First, a comprehensive Chu Nom font (see Section 2) and a generic background are used to create an initial training dataset. Afterwards, a convolutional character detection system (see Section 3) is trained and applied to the entire dataset of Vietnamese steles. The character detection results, which are far from being perfect before calibration, are then processed by a layout analysis method that aims to form consistent main text columns. If a main text area can be identified, it is filled with a homogeneous non-text area to create a synthetic background image. The obtained background images are then provided as input to the page synthesis for creating more realistic pages, together with an estimated number of columns and characters.

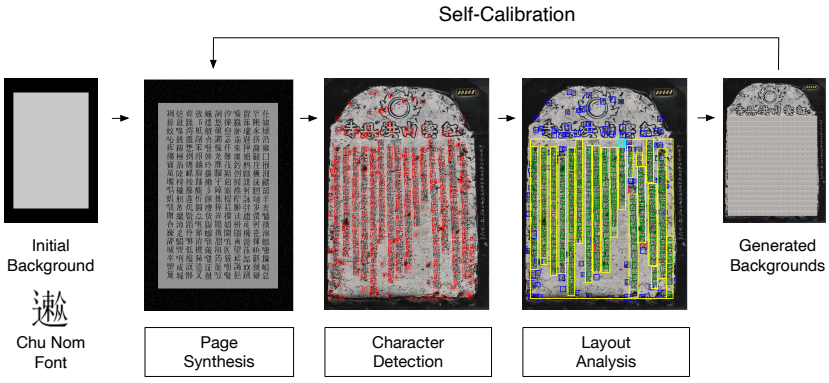


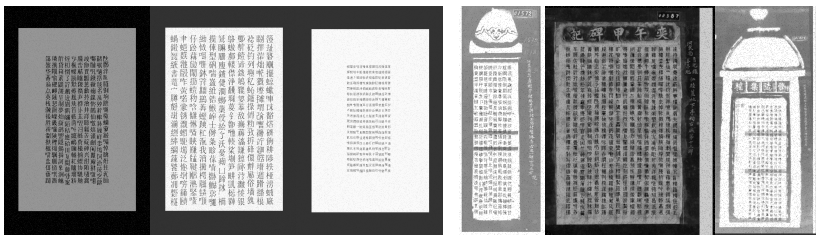
Fig. 3: Method overview

4.2 Page synthesis

In order to generate synthetic training pages, random Chu Nom characters are printed in columns on a page background. The characters are selected either uniformly from the entire set of characters in the font, or according to a specific Chu Nom text, respecting the character frequencies observed in the text.

For the initial page synthesis, a simple background with gray text background and black border is used. The number of columns and characters per column are varied randomly to obtain different character sizes. When placing characters in a column, the width of the characters is fixed and the padding between the characters is varied randomly. For the second page synthesis, the layout analysis method generates hundreds of stele backgrounds. Characters are printed on the main text area according to the estimated number of columns and characters per column.

After placing the characters on the background, several random image distortions are applied to support the training of the character detection system. Distortions include blur, changes in brightness, translation, as well as salt and pepper noise.



(a) Iteration 1 (b) Iteration 2

Fig. 4: Page synthesis

Figure 4 illustrates generated training samples during the first iteration with the initial background and during the second iteration with different stele backgrounds. Note that even though the main text area is not always well detected, the generated stele backgrounds allow the character detection system to include relevant background patterns.

4.3 Character detection

The generated training samples contain bounding box coordinates of each character as well as class labels, such that character detection can be conjointly trained for localization and classification (see Section 3).

However, due to computational constraints, we have clustered the tens of thousands of Chu Nom characters into 500 categories, in order to allow the neural networks to train with reasonable memory requirements.

Fig. 5: Character clusters

For clustering, we create binary images of the printed characters with a width of 50 pixels and extract 100 grid features $(x_1, \ldots, x_{100})$ using a regular 10×10 grid, where x_i is the number of black pixels in the respective cell. Afterwards, k-means clustering is applied for forming k character classes. For the case $k = 1$, all characters belong to the same class and the character detection system is optimized for localization only. Figure 5 illustrates seven exemplary clusters for $k = 500$.

4.4 Layout analysis

The goal of layout analysis is to identify the main text region of a stele, the number of columns, and the number of characters per column based on the character detection results. If these estimates can be done with a high confidence, a homogeneous text background pattern is identified and used to fill the main text region. The generated stele background images are then used for page synthesis.

Avoiding the need for manual annotation, layout analysis relies on unsupervised clustering. It aims to form consistent columns based on two basic assumptions:

1. Within the same stele the main text characters have a similar size.
2. Within the same column characters are relatively close together.

Algorithm 1 Column Detection

Require:
$\mathcal{D}$: set of detected characters
$(\epsilon = 2, k = 3)$: DBSCAN parameters
$\tau_s = 0.3$: maximum size deviation
$\tau_x = 0.1$: maximum horizontal deviation
Ensure:
$\mathcal{C}$: set of main text columns
1: $m \leftarrow$ character with the median area in $\mathcal{D}$
2: $\mathcal{D} \leftarrow \mathcal{D} - \{c \in \mathcal{D} : \frac{|c.width - m.width|}{m.width} > \tau_s \vee \frac{|c.height - m.height|}{m.height} > \tau_s\}$
3: $\mathcal{C} \leftarrow DBSCAN(\mathcal{D}, \epsilon, k, d_{m,\tau_x}(.,.))$

We use DBSCAN [4] to perform the column clustering. This algorithms groups together points that have at least k neighbors within distance ϵ. Algorithm 1 details the proposed column detection method. First, the median character m is determined among all detected bounding boxes with respect to its area (line 1). Afterwards, characters whose width or height deviate more than τ_s from the median character are discarded, following the assumption that main text characters have a similar size (line 2). Finally, column clusters are formed using DBSCAN with parameters ϵ and k and a special column metric d_{m,τ_x} (line 3). The column metric is defined as

$$d_{m,\tau_x}(c_1, c_2) = \begin{cases} \infty, & \text{if } \frac{|c_1.x - c_2.x|}{m.width} > \tau_x \\ \frac{|c_1.y - c_2.y|}{m.height}, & \text{otherwise} \end{cases}$$

where $c.x$ and $c.y$ are the bounding box center coordinates of character c. That is, characters can only be assigned to the same column if their horizontal position does not deviate more than τ_x.

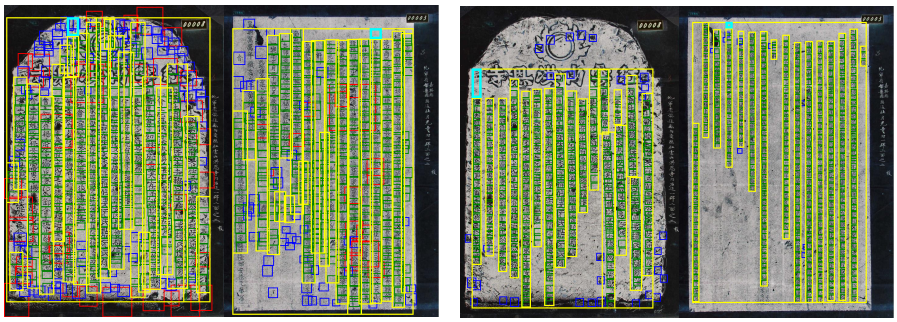
Note that because the self-calibration does not have access to annotated training samples, reasonable defaults have to be chosen for the parameters. With the suggested defaults in Algorithm 1, main text characters must not deviate more than $\tau_s = 30\%$ in width or height from the median character. They are grouped into columns if their horizontal deviation is less than $\tau_x = 10\%$ and there are at least $k = 3$ other characters above or below within the range of $\epsilon = 2$ character heights, following the assumption that characters in the same column are relatively close together.

By using the median character as a reference, the proposed column detection method is scale-invariant. This is important for the dataset of Vietnamese steles, where the character size varies significantly from one stele to another.

The final step of layout analysis is to identify a homogeneous non-text patch and use it to fill the main text region, i.e. the bounding box of all detected columns, such that the engraved characters are erased and an empty background is obtained. For finding such a patch, we systematically scan areas above, below, and between columns, convert them to grayscale, and choose the area with minimum standard deviation of its grayscale values.

The empty background image is then provided to the page synthesis method for the next self-calibration step, together with the bounding box of the main text, the number of detected columns, and the maximum number of characters per column. In order to increase the quality of the results, background images are used only if there are at least 100 characters in the main text.

Figure 6 illustrates two layout analysis examples. Red boxes indicate characters that are too small or too big, blue boxes are outliers with respect to the column clustering, green boxes are part of a column, yellow boxes are drawn around columns and the main text area, and the cyan box indicates the homogeneous non-text patch. Note that although the character detection quality is rather low during the first iteration, the layout analysis still detects useful main text areas. Examples of resulting synthetic pages are illustrated in Figure 4.



(a) Iteration 1 (b) Iteration 2

Fig. 6: Layout analysis

5 Experimental Evaluation

To test the possibilities and limitations of annotation-free character detection, we have conducted a series of experiments on the dataset of 2,036 stele images with both YOLO and FCOS. In the following, we describe the experimental setup and evaluation measures in more detail before presenting the results.

5.1 Setup

The neural networks are trained on synthetic data and then applied to the 2,036 real stele images without manual interaction. The proposed self-calibration is performed without using human annotations, neither for fine-tuning the neural networks nor for optimizing meta-parameters. To test the performance of the method, a subset of 65 stele images has been manually annotated with character bounding boxes. This test set consists of two parts.

- **TEST-A.** In a first step, 10 stele images have been annotated that show some diversity in layout but contain main text characters that are relatively well-readable for the human eye.
- **TEST-B.** In a second step, to increase the difficulty, we have randomly selected 55 other samples[7]. Some of them are barely readable.

We consider several experimental setups that aim to investigate key aspects of the annotation-free character detection method, including the number of character classes C for object detection, the size of the dictionary D for page synthesis, the size of the synthetic training set, and the number of training epochs. Starting from a baseline setup, only one parameter is changed at a time.

- **Baseline-10K.** Object detection is only trained for bounding box regression, not for classification ($C = 1$), all characters of the font are used for page synthesis ($D = 26,969$), and the network is trained one epoch on 10,000 synthetic pages.
- **C-500.** Use k-means to cluster the characters into 500 classes ($C = 500$).
- **D-4855.** Consider a reduced set of frequent characters that appear in the famous Tale of Kieu ($D = 4,855$).
- **Baseline-30K.** Use a training set of 30,000 synthetic training pages.
- **Fully-Trained.** Train the object detection network until convergence, typically 10-15 epochs.

For each setup, we evaluate the character detection performance before layout analysis (**Raw detection**), after layout analysis (**Layout analysis**), and after self-calibration (**Calibrated**). The general configuration of the two object detection methods is described in Section 3. For YOLO, we consider a YOLOv5m model that has been pretrained on the COCO dataset [11]. For FCOS, we consider an EfficientNet-B4 backbone, which is trained from scratch using Adam optimization. Training one epoch on 30,000 synthetic pages with a batch size of 8 and two Titan RTX cards takes about 80 minutes.

5.2 Evaluation Measures

For evaluating the detection results, we first compute an optimal assignment between the detected character boxes and the ground truth annotations, using the intersection over union (IoU) $\frac{|A \cap B|}{|A \cup B|}$ as the matching cost and solving a linear sum assignment problem.

Several metrics are then computed from the optimal assignment, including the mean IoU, precision, recall, F_1 score, and the *character detection accuracy* (CDA), which we define as

$$CDA = \frac{N - S - D - I}{N} \tag{3}$$

[7] More specifically, three random selections have been performed. 11 samples have been selected among already transcribed steles, 22 samples from the dataset used in previous work [20], and 22 samples from the rest of the dataset.

where N is the number of characters in the ground truth, S is the number of substitution errors, D is the number of deletions (characters that are not retrieved), and I is the number of insertions (false positives of the detection system). Matching characters are considered as correct if their IoU is greater than 0.5, otherwise they are counted as substitution errors. Note that this definition is similar to the character recognition accuracy for optical character recognition. Because the number of insertion errors is not limited, it can also be negative.

5.3 Results

Table 1 reports character detection results for the initial experiment with 10'000 synthetic pages, training the detection systems for one epoch. Besides the raw detection output, we also evaluate the characters that are considered as main text by the layout analysis. In this weakly trained scenario, FCOS performs better than YOLO. While the increase in characters (C-500) or the focus on more frequent characters (D-4855) does not have a strong effect, the self-calibration clearly helps to adapt to the steles. The 87.1% character detection accuracy correspond to a remarkable increase of +35.8% when compared with the baseline. Layout analysis improves the results as long as the networks are not calibrated but slightly decreases the performance for calibrated models.

Table 2 reports the outcome of detection systems that are trained more extensively on 30,000 synthetic pages. In this setup, YOLO achieves quite a remarkable performance of 85.3% after layout analysis even without calibration, i.e. only trained with printed characters on the simple gray background with black border. After calibration, both YOLO and FCOS achieve a character detection accuracy of 87.3%, which is similar to the best FCOS result in Table 1. The comparison with a fully trained network indicates that FCOS overfits with more training epochs. YOLO, on the other hand, does not overfit, which we assume is due to the positive effect of pretraining on COCO.

Table 3 shows the results for the more difficult test set TEST-B. Even though FCOS has a very low detection accuracy at the beginning (due to a large number of insertion errors; see Equation 3), the layout analysis can make sense of the results and produces useful stele backgrounds for self-calibration, which increases the final performance to 60.7%. For this more difficult test set, YOLO proves more robust and achieves a significantly better result of 75.3%.

In all cases, the raw output of the calibrated system, trained until convergence on the synthetic pages and adapted one epoch on the synthetic stele backgrounds emerges as the most stable configuration. Table 4 summarizes the different evaluation measures for TEST-A in this configuration, showing that the character detection accuracy (CDA) is closely related to the recall of the system.

Figure 7 illustrates exemplary detection results. In general, the system is able to detect characters of different sizes, even for styles that differ strongly from the printed characters used for training (see for example the fourth stele from the left in Figure 7). Errors are often observed at the border of the main text region, or due to variations in the layout, e.g. when a single column is continued with two smaller columns below.

Table 1: Character detection accuracy on the initial test dataset (TEST-A) using 10,000 synthetic training pages. The baseline is compared with an extended number of classes (C-500), a reduced number of Nom characters (D-4855), and a self-calibrated system. The best result is highlighted in bold font.

		Baseline-10K	C-500	D-4855	Calibrated
FCOS	Raw detection	51.3	34.2	56.7	**87.1**
	Layout analysis	55.2	53.6	59.5	85.0
YOLO	Raw detection	29.4	25.3	25.2	**35.8**
	Layout analysis	34.8	29.6	34.0	33.9

Table 2: Character detection accuracy on the initial test dataset (TEST-A) using 30,000 synthetic training pages. The baseline is compared with a fully trained and calibrated system.

		Baseline-30K	Fully-Trained	Calibrated
FCOS	Raw detection	52.7	48.8	**87.3**
	Layout analysis	59.6	52.9	84.7
YOLO	Raw detection	64.4	72.4	**87.3**
	Layout analysis	66.2	85.3	86.8

Table 3: Character detection accuracy on the extended test dataset (TEST-B) using 30,000 synthetic training pages. The baseline is compared with a fully trained and calibrated system.

		Baseline-30K	Fully-Trained	Calibrated
FCOS	Raw detection	1.1	3.4	**60.7**
	Layout analysis	24.9	22.9	59.9
YOLO	Raw detection	30.2	57.7	**75.3**
	Layout analysis	44.8	68.1	71.6

Table 4: Performance evaluation of the fully trained and calibrated system for TEST-A using 30,000 synthetic training pages.

	CDA	IoU	Precision	Recall	F1 Score
FCOS	87.3	65.1	98.9	87.3	92.6
YOLO	87.3	66.8	96.3	88.9	92.3

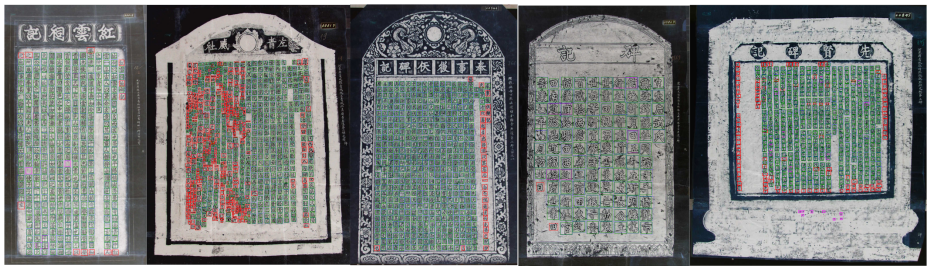


Fig. 7: Exemplary character detection results. The ground truth is marked in blue, detected characters in green, deletion errors (missing characters) in red, and insertion errors (false positives) in magenta.

6 Conclusions

The experimental results indicate a quite surprising ability of the convolutional object detection methods to transfer knowledge from a printed Nom font to the heterogeneous images of stone engravings. The proposed annotation-free approach is expected to be very useful for an initial exploration of the document collection, providing immediate access to hundreds of thousands of character images, predominant layouts, estimated number of characters per stele, etc. – meta-information that is obtained "for free" without manually labeling stele images beforehand.

Regarding the performance of the system, there is clearly a margin of improvement. In addition to the suggested background calibration, it would be interesting to include a text foreground calibration as well, e.g. by means of confidently extracted character images in a self-training scenario, or by using neural networks to generate characters in a similar style. Furthermore, the detected character locations may be used for keyword spotting and, eventually, for automatic transcription. Finally, it might be rewarding to explore the concept for other scripts and languages.

Acknowledgements

This work has been supported by the Swiss Hasler Foundation (project 20008). It has also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 833933 - VIETNAMICA).

We would like to thank Bélinda Hakkar, Marine Scius-Bertrand, Jean-Michel Nafziger, René Boutin, Morgane Vannier, Delphine Mamie and Tobias Widmer for annotating bounding boxes during more than hundred hours to create the ground truth of the test set.

References

1. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv:2004.10934 (2020)
2. Borges Oliveira, D.A., Viana, M.P.: Fast CNN-based document layout analysis. In: Proc. Int. Conf. on Computer Vision Workshops (ICCVW). pp. 1173–1180 (2017)
3. Clanuwat, T., Lamb, A., Kitamoto, A.: KuroNet: Pre-modern Japanese Kuzushiji character recognition with deep learning. In: Proc. 15th Int. Conf. on Document Analysis and Recognition (ICDAR). pp. 607–614 (2019)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. pp. 226–231 (1996)
5. Farhadi, A., Redmon, J.: Yolov3: An incremental improvement. arXiv:1804.02767 (2018)
6. Fischer, A., Liwicki, M., Ingold, R. (eds.): Handwritten historical document analysis, recognition, and retrieval — State of the art and future trends. World Scientific (2020)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
8. Jocher, G., et al.: ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration. 10.5281/ZENODO.4418161 (2021)
9. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 2117–2125 (2017)
10. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proc. Int. Conf. on Computer Vision (ICCV). pp. 2980–2988 (2017)
11. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: Proc. 13th European Conf. on Computer Vision (ECCV). pp. 740–755 (2014)
12. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 8759–8768 (2018)
13. Nguyen, K.C., Nguyen, C.T., Nakagawa, M.: Nom document digitalization by deep convolution neural networks. Pattern Recognition Letters **133**, 8–16 (2020)
14. Papin, P.: Aperçu sur le programme "Publication de l'inventaire et du corpus complet des inscriptions sur stèles du Viêt-Nam". Bulletin de l'École Française d'Extrême-Orient **90**(1), 465–472 (2003)
15. Papin, P., Manh, T.K., Nguyên, N.V.: Corpus des inscriptions anciennes du Vietnam. EPHE, EFEO, Institut Han-Nôm (2005–2013)
16. Papin, P., Manh, T.K., Nguyên, N.V.: Catalogue des inscriptions du Viêt-Nam. EPHE, EFEO, Institut Han-Nôm (2007–2012)
17. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 779–788 (2016)
18. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 658–666 (2019)

19. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Proc. Int. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI). pp. 234–241 (2015)
20. Scius-Bertrand, A., Voegtlin, L., Alberti, M., Fischer, A., Bui, M.: Layout analysis and text column segmentation for historical Vietnamese steles. In: Proc. 5th Int. Workshop on Historical Document Imaging and Processing (HIP). pp. 84–89 (2019)
21. Stewart, S., Barrett, B.: Document image page segmentation and character recognition as semantic segmentation. In: Proc. 4th Int. Workshop on Historical Document Imaging and Processing (HIP). pp. 101–106 (2017)
22. Sudholt, S., Fink, G.A.: PHOCNet: A deep convolutional neural network for word spotting in handwritten documents. In: Proc. 15th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR). pp. 277–282 (2016)
23. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Int. Conf. on Machine Learning (ICML). pp. 6105–6114 (2019)
24. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: Proc. Int. Conf. on Computer Vision (ICCV). pp. 9627–9636 (2019)
25. Wang, C.Y., Liao, H.Y.M., Wu, Y.H., Chen, P.Y., Hsieh, J.W., Yeh, I.H.: CSPNet: A new backbone that can enhance learning capability of CNN. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 390–391 (2020)
26. Wu, Y., He, K.: Group normalization. In: Proc. European Conf. on Computer Vision (ECCV). pp. 3–19 (2018)
27. Yang, H., Jin, L., Huang, W., Yang, Z., Lai, S., Sun, J.: Dense and tight detection of Chinese characters in historical documents: Datasets and a recognition guided detector. IEEE Access **6**, 30174–30183 (2018)
28. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 9759–9768 (2020)