Linear Object Detection in Document Images using Multiple Object Tracking

Philippe Bernet, Joseph Chazalon, Edwin Carlinet, Alexandre Bourquelot, and Elodie Puybareau

EPITA Research Lab (LRE), Le Kremlin-Bicêtre, France
{firstname.lastname}@epita.fr

Abstract. Linear objects convey substantial information about document structure, but are challenging to detect accurately because of degradation (curved, erased) or decoration (doubled, dashed). Many approaches can recover some vector representation, but only one closed-source technique introduced in 1994, based on Kalman filters (a particular case of Multiple Object Tracking algorithm), can perform a pixel-accurate instance segmentation of linear objects and enable to selectively remove them from the original image. We aim at re-popularizing this approach and propose: 1. a framework for accurate instance segmentation of linear objects in document images using Multiple Object Tracking (MOT); 2. document image datasets and metrics which enable both vector- and pixel-based evaluation of linear object detection; 3. performance measures of MOT approaches against modern segment detectors; 4. performance measures of various tracking strategies, exhibiting alternatives to the original Kalman filters approach; and 5. an open-source implementation of a detector which can discriminate instances of curved, erased, dashed, intersecting and/or overlapping linear objects.

Keywords: Line segment detection · Benchmark · Open source

1 Introduction

The detection of linear structures is often cast as a boundary detection problem in Computer Vision (CV), focusing on local gradient maxima separating homogeneous regions to reveal edges in natural images. Document images, however, often exhibit very contrasted and thin strokes which carry main information, containing printings and writings, overlaid on some homogeneous "background". It is therefore common to observe a local maximum and a local minimum in a short range along the direction normal to such stroke. As a result, methods based on region contrast tend to fail on document images, rejecting text and strokes as noise or produce double-detections around linear structures.

In this work, we are interested in enabling a fast and accurate pre-processing which can detect and eventually remove decorated, degraded or overlapping linear objects in document images. Figure 2 illustrates the two major outputs such

method needs to produce: a vector output containing the simplified representation of all start and end coordinates of approximated line segments; and a pixel-accurate instance segmentation where each pixel is assigned zero (background), one, or more (intersections) linear object label(s), enabling their removal while preserving the integrity of other shapes (i.e. without creating gaps).

Our work aims at reviving a technique proposed in 1994 [27,23,28] which introduced a way to leverage Kalman filters to detect and segment instances of linear objects in document images, for which no public implementation is available. We propose a re-construction of this method with the following new contributions: 1. we introduce a general framework for accurate instance segmentation of linear objects in document images using Multiple Object Tracking (MOT), a more general approach, which enables to separate the stages of the approach properly and eventually leverage deep image filters, and also to extend the method with new trackers; 2. we introduce datasets and metrics which enable both vector- and pixel-based evaluation of linear object detector on document images: from existing and newly-created contents we propose vector and pixel-accurate ground-truth for linear object and release it publicly; 3. we show the performance of MOT approaches against modern segment detectors on a reproducible benchmark using our public datasets and evaluation code; 4. we compare the performance of various tracking strategies, exhibiting alternatives to the original Kalman filters approach and introducing, in particular, a better parametrization of the Kalman filters; and 5. we release the first open-source implementation of a fast, accurate and extendable detector which can discriminate instances of curved, dashed, intersecting and/or overlapping linear objects.

This study is organized as follows. We review existing relevant methods for linear object detection in Section 2, detail the formalization of our proposed framework in Section 3, and present the evaluation protocol (including the presentation of datasets and metrics), as well as the results of our benchmarks for vector and pixel-accurate detection, in Section 4.

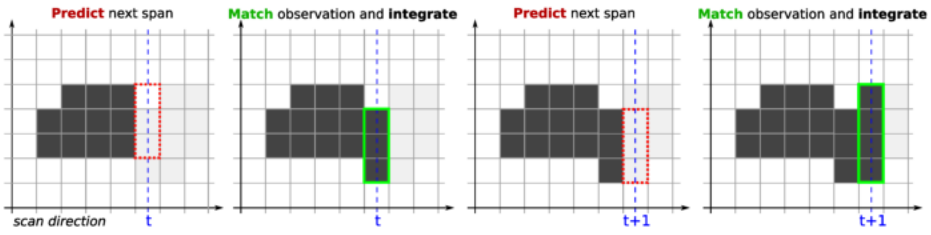

Fig. 1: Tracking process of linear objects. Images are scanned in a single left-to-right (resp. top-to-bottom) pass. At each step t, the tracker predicts the next *span* (red box) of a linear object from past (opaque) observations, then it matches candidate observations, selects the most probable one (green box), and adjust its internal state accordingly. Multiple Object Tracking (MOT) handles intersections, overlaps, gaps, and detects boundaries accurately.

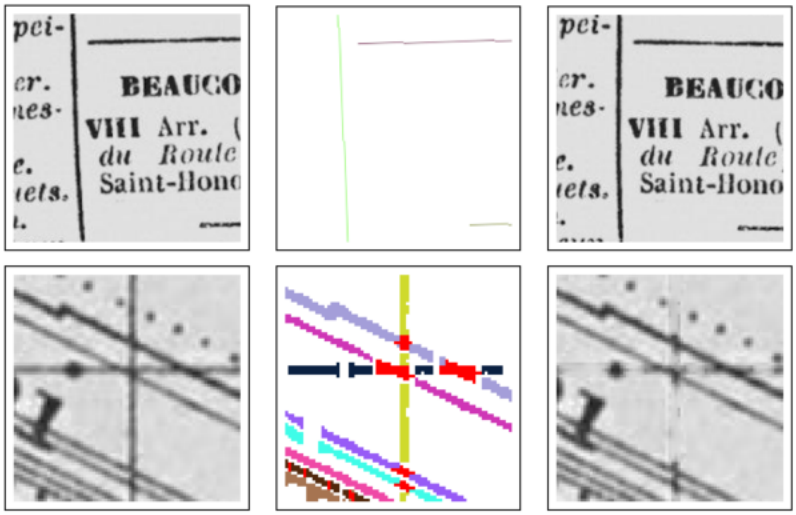

Fig. 2: Two possible applications. *First row* illustrates an orientation detection task, relying on locating endpoints of vertical lines. *Second row* illustrates a multi-stage information extraction task, relying on discriminating between building shapes and georeferencing lines (horizontal and vertical) in map images. *Left*: image inputs, *center*: detections, *right*: rotated or inpainted result.

2 State of the Art in Linear Object Detection

The two complementary goals addressed in this work were traditionally tackled by distinct method categories. Among these categories, some produce vector outputs (with end coordinates, but also sometimes sparse or dense coordinates of points laying on the object), and some others provide some pixel labeling of linear objects. Such pixel labeling is often limited to a *semantic segmentation* [19], i.e. a binary "foreground" vs "background" pixel-wise classification, and, as we will see, only tracking-based approaches can propose an *instance segmentation* [19], i.e. effectively assign each pixel to a set of objects. Due to the lack of datasets for linear object detection in document images, we are particularly interested, in this current work, in techniques which have light requirements on training data, and leave for future research the implementation of a larger benchmark. Linear object detection is very related to the problem of *wireframe parsing*, well studied in CV, for which several datasets were proposed [11,16], but their applicability to document images, which exhibit distinct structural and textural properties, were not specifically studied. We review a selection of approaches which pioneered key ideas in each category of the taxonomy we propose here.

Pixel-wise edge classifiers. The Sobel operator [32] is a good example of an early approach which predicts, considering a limited spatial context, the probability or score that a particular image location belongs to an edge. This large family includes all hourglass-shaped deep *semantic segmentation* networks, notably

any approach designed around some U-Net variant [31,24] like the HED [36] and BDCN [15] edge detectors. Such approaches can make pixel-accurate predictions but cannot discriminate object instances. To enable their use for vectorization, Xue et al. [38] proposed to predict an Attraction Field Map which eases the extraction of 1-pixel-thin edges after a post-processing stage. EDTER [29] pushes the *semantic segmentation* approach as far as possible with a transformer encoder, enabling to capture a larger context. While all these techniques can produce visually convincing results, they cannot assign a pixel to more than one object label and require a post-processing to produce some vector output. Furthermore, their training requirements and computational costs forces us to remove them from our comparison, but they could be integrated as an image pre-processing in the global framework we propose.

Hough Transform-based detectors. The Hough transform [17] is a traditional technique to detect lines in images. It can be viewed as a kind of meta-template matching technique: line evidence L with (ρ, θ) polar parameters is supported by pixel observations (x_i, y_i) according to the constraint that $(L) : \rho = x_i \cos\theta + y_i \sin\theta$. In practice, such technique relies on a binarization pre-processing to identify "foreground" pixels which can support a set of (ρ, θ) values in polar space. Line detection is performed by identifying maximal clusters in this space, which limits the accuracy the detection to a very coarse vectorization, while relying on a high algorithmic complexity. Several variants [26] were proposed to overcome those limitations: the Randomized Hough Transform [21] and Probabilistic Hough Transform [20] lowered the computational cost thanks to some random sampling of pixels, and the Progressive Probabilistic Hough Transform [13] added the ability to predict start and end line segment coordinates, turning the approach into an effective linear object detector. This method can tolerate gaps and overlaps to some extents, and is highly popular thanks to its implementation in the OpenCV library [3]. However, it is very sensitive to noise, to object length, rather inaccurate with slightly curved objects, and cannot assign pixels to linear object instances.

Region growing tracers. The Canny edge detector [5] is the most famous example of this category. It is based on the following steps: a gradient computation on a smoothed image, the detection of local gradient maximums, the elimination of non-maximal edge pixels in the gradient's direction, and the filtering of edge pixels (based on gradient's magnitude) in the edge direction. All region growing approaches improve this early algorithm, probably introduced by Burns et al. [4], which effectively consists in finding salient edge stubs and tracing from there initial position. LSD [14] was the first approach which took linear object detection to the next level in natural images. Fast and accurate, it is based on a sampling of gradient maximums which are connected only if the gradient flow between these candidates is significantly strong compared to a background noise model, according to the *Helmholtz principle* proposed by [12]. EDLine [2] and AG3line [39] improved the routing scheme to connect distant gradient maximums using Least Square fitting and a better sampling, accelerating and enhancing

the whole process. CannyLines [25] proposed a parameter-free detection of local gradient maximums, and reintegrated the *Helmholtz principle* in the routing. Ultimately, ELSED [33] further improved this pipeline to propose the fastest detector to date, with a leading performance among learning-less methods. These recent approaches are very fast and accurate, require no training stage, and can handle intersections. However, they do not detect all pixels which belong to a stroke, and have limited tolerance to gaps and overlaps. Finally, these methods require a careful tuning to be used on document images as the integrated gradient computation step tends to be problematic for thin linear objects, and leads to double detections or filters objects (confusing them with noise).

Deep linear object detectors. Region Proposal Networks, and their ability to be trained end-to-end using RoI pooling, where introduced in the Faster R-CNN architecture [30]. This 2-stage architecture was adapted to linear object detection by the L-CNN approach [40]. L-CNN uses a junction heatmap to generate line segment proposals, which are fed to classifier using a Line of Interest pooling, eventually producing vector information containing start and end coordinates. HAWP [38] accelerated the proposal stage by replacing the joint detection stage with the previously-introduced Attraction Field Map [38]. F-Clip [10] proposed a similar, faster approach using a single-stage network which directly predicts center, length and orientation for each detected line segment. These approaches produce very solid vector results, but they are still not capable to assign pixels to a particular object instance. Furthermore, their important computation and training data requirements forces us to remove them from our current study.

Vertex sequence generators. Recently, the progress of decoders enabled the direct generation of vector data from an image input, using a sequence generator on top of some feature extractor. Polygon-RNN [7,1] uses a CNN as feature extractor, and an RNN decoder which generates sequences of vertex point coordinates. LETR [37] uses a full encoder/decoder transformer architecture and reaches the best wireframe parsing accuracy to date. However, once again, such architectures are currently limited to vector predictions, and their computation and training data requirements make them unsuitable for our current comparison.

Linear object trackers. A neglected direction, with several key advantages and which opens interesting perspectives, was introduced in 1994 in a short series of papers [27,28,23]. This approach leverages the power of Kalman filters [18], which were very successful for sensors denoising, to stabilize the detection of linear objects over the course of two image scans (horizontal and vertical). By tracking individual object candidates, eventually connecting or dropping them, this approach proposed a lightweight solution, with few tunable parameters, which can *segment instances* of linear objects by assigning pixels to all the objects they belong to, but also deal with noise, curved objects, gaps and noise. However, no comparison against other approaches, nor public implementation, were disclosed. This called for a revival and a comparison against the fastest methods to date.

3 MOT Framework for Linear Object Detection

As previously mentioned, we extend the original approach of [27,28,23], which performs 2 scans over an image, plus a post-processing, to detect linear objects. We will describe the horizontal scan only: the vertical one consists in the same process applied on the transposed image. During the horizontal scan, the image is read column by column in a single left-to-right pass; each column being considered as a *1-dimensional scene* containing linear object *spans*, i.e. slices of dark pixels in the direction normal to linear object (Figure 1). Those spans are tracked scene by scene, and linked together to retrieve objects with pixel accuracy. To ensure accurate linking, and also to tolerate gaps, overlaps and intersections, each coherent sequence of *spans* is modeled as a *Kalman filter* which stores information about past *spans*, and can predict the attributes (position, thickness, luminance...) of the next most probable one. By matching such *observations* with *predictions*, and then correcting the internal parameters of the *filter*, observations are aggregated in a self-correcting model instance for each linear object. Once the horizontal and vertical scans are complete, object deduplication is required to merge double detections close to 45 degrees. Finally, using pixel-accurate information about each object instance, several outputs can be generated, and in particular: a mapping which stores for each pixel the associated object(s), and a simplified vector representation containing first and last span coordinates only.

Framework overview. We propose to abstract this original approach into a more general Multiple Objects Tracking (MOT) framework. This enables us to explicit each stage of such process, and introduce variants. Linear object detection using 2-pass MOT can be detailed as follows.

Pre-processing. In Section 4, we will report results with grayscale images only, as this is the simplest possible input for this framework. However, some preprocessing may be used to enhance linear object detection. In the case of uneven background contrast, we obtained good results using a *black top hat*, and to be able to process very noisy images, or images with rich textures, it may be possible to train a semantic segmentation network which would produce some edge probability map.

Processing. The horizontal and vertical scans, which can be performed in parallel, aim at initializing, updating and returning a set of *trackers*, a generalization of the "filters" specific to Kalman's model. Like in the original approach, each detected instance is tracked by a unique *tracker* instance. *Trackers* are structures which possess an internal State S containing a variable amount of information, according to the variant considered; two key methods "predict" and "integrate" which will be described hereafter; and an internal list of *spans* which compose the linear object. The State and the two methods can be customized to derive alternate tracker implementations (we provide some examples later in this section). For each *scene* t (column or line) read during the scan, the following steps are performed. Steps 1 and 2 can be performed in parallel, as well as steps 4, 5 and 6.

1. Extract the set of observations O_t^j, $j \in [0, nobs_t[$. Observations represent linear object *spans*, and contain information about their *position* in the scene, *luminance* and *thickness*. At this step, some observations may be rejected because there size is over a certain threshold. This threshold is a parameter of the method. The algorithm and the associated illustration in fig. 3 detail our implementation this step based on our interpretation of the original approach [27,28,23].
2. Predict the most probable next observation X_t^i for each tracker i, using its current internal State S_{t-1}. Predictions have the same structure and attributes as observations.
3. Match extracted observations O_t^j with predicted ones X_t^i. Matching is a two-step process performed for each tracker i. First, candidate observations $O_t^{i,j}$ are selected based on a distance threshold, and *slope*, *thickness* and *luminance* compatibility (they must be inferior to 3 times the standard deviation of each parameter, computed over a window of past observations). Second, the closest observation is matched: $\hat{O}_t^i = \arg\min |O_t^{i,j}(\text{position}) - X_t^i(\text{position})|$. The same observation can be matched by multiple trackers when lines are crossing. Unmatched observations $\bar{O}_t$ are kept until step 5.
4. Integrate new observations $\hat{O}_t^i$ *into trackers' States* S_t^i, considering (in the more general case) current State S_{t-1}^i, scene t, matched observation $\hat{O}_t^i$ and prediction X_t^i. This enables each tracker to adapt its internal model to the particular object being detected.
5. Initialize new trackers from unmatched observations $\bar{O}_t$. New trackers are added to the active pool of trackers to consider at each scene t.
6. Stop trackers of lost objects. When a tracker does not match any observation for too many t, it is removed from the pool of active trackers. The exact threshold depends on the current size of the object plus an absolute thresholds for acceptable gap size. Those two thresholds are parameters of our method.

Post-processing. *Deduplication* is required because 45°-oriented segments may be detected twice. Duplications are removed by comparing and discarding object instances with high overlap. An optional *attribute filtering* may then be performed, that consists in filtering objects according to their length, thickness or angle. This filtering must be performed after the main processing stage to avoid missing intersection and overlaps with other objects, would these be linear or not: this makes possible to handle interactions between handwritten strokes and line segments, for instance. Finally, *outputs* can be generated by decoding the values stored in each *tracker* object.

Tracker variants. Each tracker variant features a different internal State structures (which needs to be initialized), and specific prediction and integration functions. However, in the context of Linear Object Detection, we keep the original model of the IRISA team [27,28,23]: all observations have luminance, position and thickness attributes, and States have a similar structure, generally adding a *slope* attribute to capture position change. In this work, we consider the follow-

Observation extraction algorithm

1: **for** $n = 0$ to max_n **do**
2: **if** $Image_t[n] < l_{mm}$ **then**
3: $s \leftarrow 0$; **do** { s ← s + 1 } **while** $(Image_t[n+s] < l_{mm})$;
4: $l_{min} \leftarrow$ min of $Image_t[n \dots n+s]$
5: $l_{max} \leftarrow$ max of $Image_t[n \dots n+s]$
6: contrast $\leftarrow l_{max} - l_{min}$
7: $l_{stab} \leftarrow l_{min} + r * \text{contrast}$
8: $n_i \leftarrow n$; **while** $(Image_t[n_i] > l_{stab})$ { $n_i \leftarrow n_i + 1$ }
9: $n_f \leftarrow n$; **while** $(Image_t[n_f] < l_{stab})$ { $n_f \leftarrow n_f - 1$ }
10: $obs_{pos} \leftarrow (n_i + n_f)$ / 2
11: $obs_{thick} \leftarrow n_f$ - n_i + 1
12: $obs_{lum} \leftarrow$ mean of $Image_t[n_i \dots n_f]$

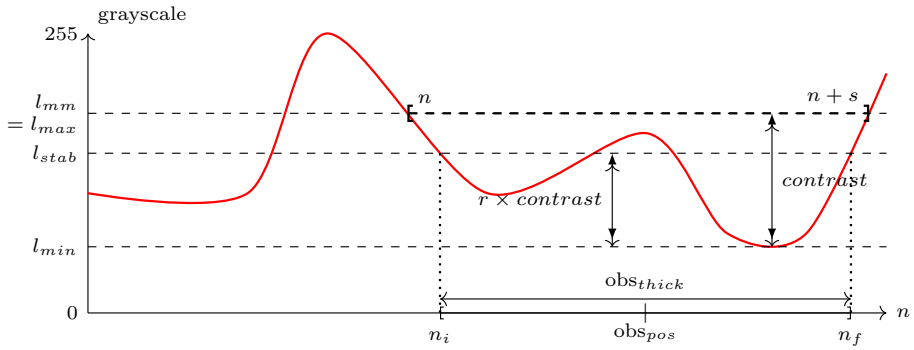

Fig. 3: Algorithm and illustration of the observation extraction process. The red curve represents the luminosity profile of a line (or a column) of pixels. The algorithm looks for a range of pixels $[n \dots n+s]$ whose value do not exceed l_{mm} (L.3) and computes its contrast (L.4-6). Then, this range is refined to extract the largest sub-range whose contrast does not exceed r (a parameter of the method set to 1 in all our application but kept from the original method) times *contrast* (L.8-9) and gives the *observation* with features computed on lines 10-12.

ing variants. To simplify the following equations, we refer to $t = 0$ as the time of the first observation of a tracker, and consider only one tracker at a time.

Last Observation. A naive baseline approach which uses as prediction the last matched observation ($X_t = \hat{O}_{t-1}$). Updating its States simply consists in storing the last matched observation in place of the previous one.

Simple Moving Average (SMA). Another baseline approach which stores the k last matched observations and extrapolates the prediction based on them. This tracker uses a *slope* attribute. Prediction for the attribute a is the average of the previous observations $X_t(a) = (\sum_{j=t-k}^{t-1} \hat{O}_j)/k$, except for the position, which is computed using the last observed position and the Exponential Moving Average (EMA) of the slope. Integration of new

observations consists in adding them to the buffer. We used a buffer of size $k = \min(t, 30)$ in our experiments.

Exponential Moving Average (EMA). A last baseline approach very similar to the previous one, which requires only to store the last matched observation $\hat{O}_{t-1}$ and the last prediction X_{t-1}. The prediction of some attribute a is computed by: $X_t(a) = \alpha * \hat{O}_{t-1}(a) + (1 - \alpha) * X_{t-1}(a)$ where α tunes importance of the new observation. The prediction of the position is computed using the last observed position and the EMA of the slope. We used a value of $\alpha = 2/(\min(t, 16) + 1)$ in our experiments.

Double Exponential [22]. This tracker uses a double exponential smoothing algorithm which is faster than Kalman filters and simpler to implement. $X_t = (2+\frac{\alpha}{1-\alpha})\cdot S_{X_{t-1}} - (1+\frac{\alpha}{1-\alpha})\cdot S_{X[2]_{t-1}}$ where $S_{X_t} = \alpha\cdot O_t + (1-\alpha)\cdot S_{X_{t-1}}$, $S_{X[2]t} = \alpha\cdot S_{X_t} + (1-\alpha)\cdot S_{X[2]_{t-1}}$ and $\alpha \in [0, 1]$ (set to 0.6 in our experiments) a smoothing parameter.

1€ Filter [6]. This tracker features a more sophisticated approach, also based on an exponential filter, which can deal with uneven signal sampling. It adjusts its low-pass filtering stage according to signal's derivative, and has only two parameters to configure: a minimum cut-off frequency that we set to 1 in our experiments, and a β parameter that we set to 0.007. We refer the reader to the original publication for the details of this approach.

Kalman Filter [18,35], IRISA variant [27,28,23]. This tracker is based on our implementation of the approach proposed by the IRISA team. The hidden State is composed of $n = 4$ attributes $S \in \mathcal{R}^n : \left[\text{pos. slope tick. lum.}\right]^\mathsf{T}$, and the process is governed by the following equation: $S_t = AS_{t-1} + w_{k-1}$ where $A \in \mathcal{R}^{n,n}$ is the transition model and w_{k-1} some process noise. Process States S can be projected to observation space $O \in \mathcal{R}^m : \left[\text{pos. tick. lum.}\right]^\mathsf{T}$ according to the measurement equation $O_t = HS_t + v_k$ where H is a simple projection matrix discarding the slope and v_k the measurement noise. By progressively refining the estimate of the covariance matrix Q (resp. R) of w (resp. v), the Kalman filter recursively converges toward a reliable estimate of the hidden State S and its internal error covariance matrix P.
The prediction step consists in (1) projecting the State ahead according to a noise-free model: $S^-_{t+1} = AS_t$; and (2) projecting the error covariance ahead: $P^-_t = AP_{t-1}A^\mathsf{T} + Q$ where Q is the process noise covariance matrix.
The integration step consists in (1) computing the Kalman gain: $K_t = P^-_t H^\mathsf{T}(HP^-_t H^\mathsf{T} + R)^{-1}$ where R is the measurement noise covariance matrix; (2) updating estimate with measurement $\hat{O}_t$: $S_t = AS^-_t + K_t(\hat{O}_t - HS^-_t)$ where H relates the State to the measurement; and (3) updating the error covariance: $P_t = (I - K_t H)P^-_t$.
The following initialization choices are made, according to our experiments and the original publications [27,28,23]. We initialize each State with the values of the first observation, with a slope of 0. We use $P_0 = I_4$ as initial value for the error covariance matrix, I_n being the identity matrix in $\mathcal{R}^n$. Q,

R, H and A assumed to be constant with the following values:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, Q = I_4 \cdot 10^{-5}, \text{ and } R = \begin{bmatrix} 1 & 1 & 4 \end{bmatrix}^\intercal.$$

4 Experiments

We report here the results obtained by comparing the performance of *training-less* approaches on two tasks: a pure *vectorization task*, where only two endpoint coordinates are required for each line segment, and an *instance segmentation task*, where pixel-accurate labeling of each line segment instance is required. This separation is due to the limitations of some approaches which can only generate vector output, as well as the limitations of existing datasets.
While the methods studied are *training-less*, they are not *parameter-free*, and such parameters require to be tuned to achieve the best performance. In order to ensure an unbiased evaluation, we manually tuned (because grid-search and other optimization techniques were not usable here) each approach on the *train set* of each dataset, then evaluate their performance on the *test set*.

4.1 Vectorization Task

Dataset. To our knowledge, no dataset for line segment detection in vector format exists for document images. We introduce here a small new, public dataset which contains endpoints annotations for line segments in 195 images of 19^{th} trade directories. In these documents, line segment detection can be used for image deskewing. The train (resp. test) set is composed of 5 (resp. 190) images, containing on average 4.3 line segments to detect each. Images samples are available in the extra material.

Metric. Our evaluation protocol is a slightly modified version of the one proposed by Cho *et al.* [9], and is defined as follows. Let $P = \{P_1, \cdots, P_n\}$ and $T = \{T_1, \cdots, T_m\}$ be respectively the set of *predicted* and *targets* line segment (LS). Let L_{ij} be the projection of P_i over T_j. We note $|X|$, the length of the LS X. A predicted LS P_i matches the *target* T_j if: 1. the prediction overlaps the target over more than 80% ($|L_{ij}|/|P_i| \geq 0.8$), 2. the perpendicular distance d_{ij} between T_j' center and P_i is less than 20 pixels, and 3. the orientations of P_i and P_j differ at most by 5°. For each P_i, we associate the closest LS (in terms of d_{ij}) among the set of *matching* LSs from the targets. We note $A = \{A_{ij}\} \in (P \times T)$ the mapping that contains $A_{ij} = (P_i, T_j)$ whenever P_i matches T_j. Our protocol allows a *prediction* to match a single *target*, but a *target* may be matched by many *predictions*. This allows for *target* fragmentation (1-to-many relation). We then compute the *precision* and *recall* scores as follows:

$$precision = \frac{\sum_{(i,j)\in A} |L_{i,j}|}{\sum_i |P_i|} \qquad recall = \frac{|\bigcup_{(i,j)\in A} L_{i,j}|}{\sum_j |T_j|}$$

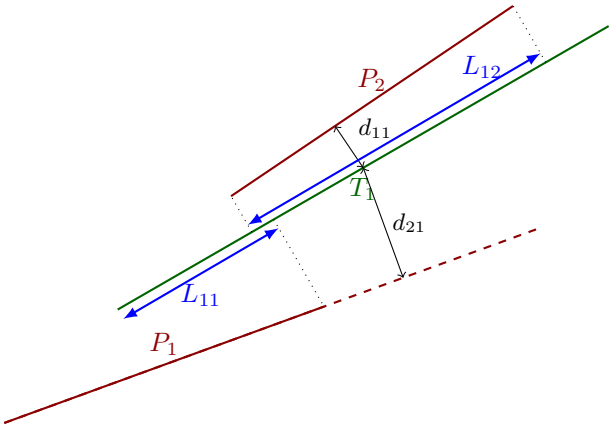

Fig. 4: Matching and scoring process. P_1 and P_2 are associated to T_1 because angles are compatible and overlaps L_{11} and L_{22} are large enough. In this example, *precision* is $\frac{|L_{11}|+|L_{21}|}{|P_1|+|P_2|}$ and is < 1 because P_1 does not fully match the ground-truth. *Recall* is $\frac{|L_{11} \cup L_{21}|}{|T_1|}$ and also is < 1 because T_1 is not fully matched.

Roughly said, *precision* stands for the quality of the *predictions* to match and cover the ground-truth, while *recall* assess the coverage of the *targets* that were matched, as illustrated in fig. 4.
Our protocol differs from the original one [9] which allows *prediction* to match many *targets* (many-to-many relation). This case can happen when segments are close to each other, and results in a *precision* score that can exceed *1.0*. Our variant ensure precision remains in the $[0, 1]$ range. Even with this modification, these metrics still have limitations: duplication of detections and fragmentation of targets are not penalized. We thus propose an updated *precision* as:

$$precision_2 = \frac{1}{\sum_i |P_i|} \times \sum_{(i,j) \in A} \frac{|L_{i,j}|}{|A_{(*,j)}|}$$

with $|A_{(*,j)}| = |\{A_{kj} \in A\}|$ being the number of matches with target T_j (number of fragments). From *precision* (resp. $precision_2$) and *recall*, the F-score (resp. F-score$_2$) is then computed and used as the final evaluation metric.

Results. We chose to only compare Hough based and Region growing algorithms featuring a rapid, public implementation and requiring no training. In table 1, we show that the original Kalman strategy performs the best on this dataset. Nevertheless, the other tracking strategies reach almost the same level of performance (the first five are within a 2% range). The differences between the F-score and F-score$_2$ columns are explained by many very short detections that match a ground truth line and are more penalized with F-score$_2$. In table 2, state-of-the-art detectors are compared: MOT-based (using *Kalman* tracker), Edlines [2], Hough (from OpenCV) [20], CannyLines [25], LSD [14], LSD II with a filtering on

Table 1: Vectorization performance and compute time of various MOT strategies on the *trade directories* dataset. F-score and F-score$_2$ are computed per-page and averaged on the dataset (standard deviation is shown between brackets).

	Time (ms)	F-Score Train	F-Score Test	F-score$_2$ Train	F-score$_2$ Test
Last observation	**616**	**95.2 (±7.5)**	90.0 (±24.1)	**92.7 (±13.0)**	87.2 (±24.7)
SMA	652	**95.2 (±7.5)**	90.0 (±24.0)	**92.7 (±13.0)**	87.4 (±24.7)
EMA	617	92.6 (±8.6)	89.7 (±24.3)	88.3 (±16.9)	86.5 (±24.8)
Double exp. [22]	623	94.6 (±7.2)	87.3 (±25.6)	85.6 (±15.9)	81.7 (±26.4)
One euro [6]	627	**95.2 (±7.5)**	**90.1 (±24.0)**	90.8 (±17.2)	87.2 (±24.7)
Kalman [27]	633	**95.2 (±7.5)**	**90.1 (±24.0)**	**92.7 (±13.0)**	**87.6 (±24.6)**

Table 2: Vectorization performance and compute time of various line segment detection approaches on the *trade directories* dataset. Even on this rather simple dataset, these results show the superiority of MOT-based approaches for line segment detection in document images.

	Time (ms)	F-Score Train	F-Score Test	F-score$_2$ Train	F-score$_2$ Test
MOT (Kalman)	633	**95.2 (±7.5)**	**90.1 (±24.0)**	**92.7 (±13.0)**	**87.6 (±24.6)**
AG3Line [39]	434	66.2 (±23.8)	72.5 (±35.4)	25.9 (±9.9)	24.2 (±13.7)
CannyLines [25]	551	81.2 (±22.7)	84.4 (±24.2)	39.0 (±13.5)	34.2 (±14.0)
EDLines [2]	314	83.2 (±23.6)	87.4 (±24.0)	35.5 (±8.3)	30.5 (±12.3)
ELSED [33]	**264**	91.1 (±11.3)	87.0 (±26.6)	45.3 (±9.6)	35.2 (±13.7)
Hough [13]	419	80.5 (±14.5)	64.8 (±30.0)	23.5 (±9.2)	18.2 (±10.1)
LSD [14]	2338	18.7 (±10.3)	12.5 (±8.5)	1.6 (±1.5)	0.5 (±0.6)
LSD II	2206	76.7 (±28.6)	53.3 (±43.7)	47.6 (±24.7)	20.7 (±17.9)

segment lengths, ELSED [33], and AG3Line [39]. All these techniques performs much lower than the MOT-based approach because of thick lines being detected twice (especially with the F-score$_2$ metric where splits are penalized). The two previous tables exhibit a large standard deviation of the performance because of some bad quality pages (noise, page distortions...) where most methods fail to detect lines. This behavior is shown on fig. 5 where the F-score$_2$ distributions of the dataset samples are compared. It shows some outliers at the beginning where detectors get a null score for some documents.

4.2 Instance Segmentation Task

Dataset. To demonstrate the feasibility of line segment instance segmentation, we adapted the dataset of the ICDAR 2013 music score competition for staff removal [34] to add information about staff line instances. This was, to our knowledge, the only dataset for document images which features some form of line segmentation annotation we could leverage easily. As the original competition

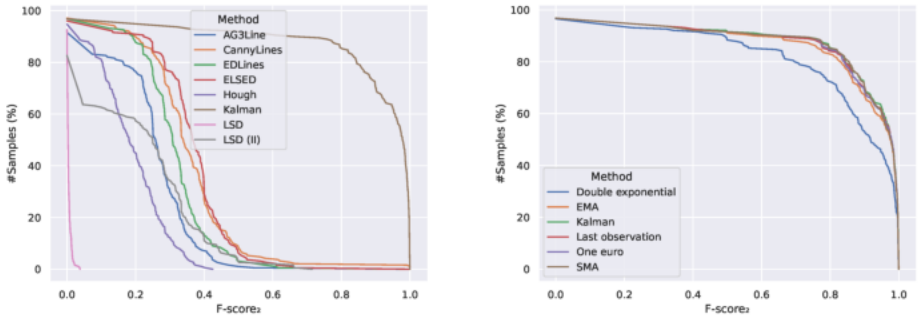

Fig. 5: Distribution of the scores obtained by traditional methods (*left*) and tracking systems (*right*) on the *trade directories* dataset. The distribution $F(\alpha)$ counts the number of pages that gets a F-score$_2$ higher than α.

dataset contain only binary information, i.e. for a given pixel whether it belong to any staff line, we annotated the 2000 binary images from the competition test set to assign to each pixel some unique identifier according to the particular staff line it belongs to, effectively enabling to evaluate the instance segmentation performance of the MOT-based line segment detectors. However, this enriched dataset contains only binary images with horizontal, non-intersecting staff lines. To enable hyperparameter calibration, we randomly selected 5 images to use as a train set, and kept the 1995 others in the test set.

Metrics. We use the COCO Panoptic Quality (PQ) metric [19] to measure instance segmentation quality. The metric is based pairings between proposed and ground-truth regions which overlap over more than 50%. The Intersection over Union (IoU) is used to score the pairings and enables to compute an instance segmentation quality ("COCO PQ") that we report here. Even if COCO PQ has become the standard metric for instance segmentation evaluation, it has two main limits: 1. it does not measure fragmentation as it considers only 1-to-1 matches; 2. IoU is somehow unstable on thin and long objects. For the sake of completeness, we also report results from the binary segmentation metric used in the original ICDAR 2013 competition [34]. The problem is seen as binary classification of pixels from which an F-Score is computed. This evaluation is not really relevant in our case because it does not include any instance information, but is shows that MOT-based techniques can be used for binary detection.

Results. Table 3 shows the performance of the various trackers to identify staff lines. All trackers successfully retrieve the staff lines. The score difference is explained by the way the trackers handle line intersections (when a staff line is hidden behind an object). The *one-euro* and *last observation* tracking strategies perform almost equally and are the ones that handle the best such cases.

Qualitative evaluation on the municipal atlases of Paris. Finally, to assess the capabilities of MOT-based line segment detectors, we processed a selection of

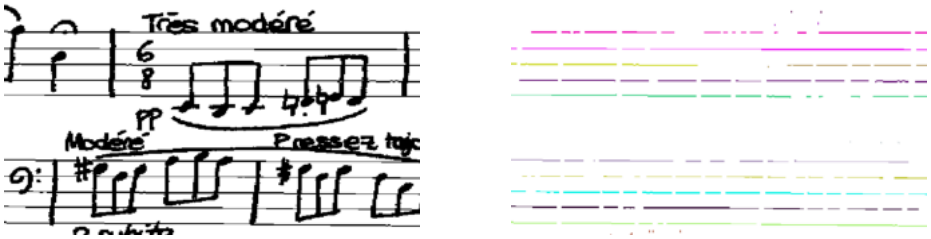

Fig. 6: Excerpt of the original image of staff lines (700 x 400) and its instance segmentation performed by the Kalman Filter predictor in 60ms.

Table 3: Instance segmentation performance and compute time of MOT-based approaches on the *music sheets* dataset adapted from ICDAR 2013 music score competition [34]. For completeness, we also report the winner of the binary segmentation contest, though such method only performs a semantic segmentation.

	Time (ms)	Panoptic Quality Train	Panoptic Quality Test	F-Score (ICDAR'13) Train	F-Score (ICDAR'13) Test
Last observation	323	86.3 (± 5.2)	83.7 (± 11.1)	95.9 (± 2.1)	95.4 (± 2.7)
SMA	323	67.6 (± 17.0)	66.0 (± 17.6)	90.7 (± 6.5)	89.9 (± 7.4)
EMA	322	74.0 (± 14.9)	65.5 (± 18.4)	92.5 (± 4.7)	89.6 (± 7.7)
Double exp. [22]	**320**	55.4 (± 16.2)	51.7 (± 15.8)	87.3 (± 5.0)	83.8 (± 8.6)
One euro [6]	327	**87.2 (± 5.9)**	**85.1 (± 9.6)**	**95.9 (± 2.1)**	**95.7 (± 2.2)**
Kalman [27]	328	85.0 (± 7.1)	80.7 (± 15.6)	95.3 (± 2.5)	94.1 (± 5.6)
LRDE-bin [34]					**97.1**

images from the dataset of the ICDAR 2021 competition on historical map segmentation [8]. The challenges of this competition relied on an accurate detection of boundaries of building blocks, of map content on the sheet, and of georeferencing lines; all of them being mostly linear objects in the map images. We retained 15 images of average size 5454x3878 (21 Mpix) for our experiment: 3 for the training set, and 12 for the test set. Because no pixel-level ground truth exists for these images, we report qualitative results. The computation times of the trackers are similar: about 5-6s per 21 Mpix maps image. Time variations are due to the number of trackers to update during the process that depends on the observations integrated with the tracking strategy. From a quality standpoint (outputs available in extra material), the Kalman strategy outperforms all the other tracking strategies as shown in fig. 7. Indeed, these documents contain many overlaps and noise that create discontinuities when extracting observations. Kalman filters enable recovering the lines even if there are hidden behind some object. On the other hand, the other trackers with simpler prediction models (such as the *one-euro* or the *last observation* trackers) integrate wrong observations and lead to line segment fragmentation.

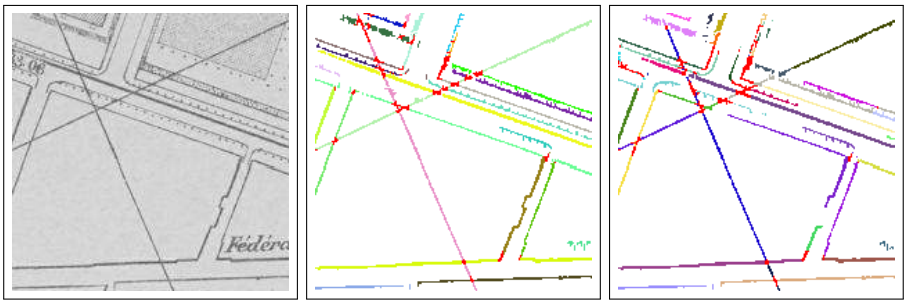

Fig. 7: Instance segmentation on map images. Left: original image, center: Kalman tracker, right: EMA tracker. With the same filtering parameters, the EMA tracker is more sensitive to gaps and overlaps and fragments objects.

5 Conclusion

Our goal was to implement a line segment detection method that was first proposed in the 1990s [27,28,23]. However, we generalized it within the Multiple Object Tracking framework that we proposed. We demonstrated the efficiency of the original proposal, which was based on Kalman filters, and also suggested some competitive alternatives. These approaches are highly robust to noise, overlapping contents, and gaps. They can produce an accurate instance segmentation of linear objects in document images at the pixel level. Results can be reproduced using open code and data available at `https://doi.org/10.5281/zenodo.7871318`.

Acknowledgements. This work is supported by the French National Research Agency under Grant ANR-18-CE38-0013 (SoDUCo project).

References

1. Acuna, D., Ling, H., Kar, A., Fidler, S.: Efficient interactive annotation of segmentation datasets with polygon-rnn++. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 859–868 (2018). https://doi.org/10.1109/cvpr.2018.00096, code available at `https://github.com/fidler-lab/polyrnn-pp`
2. Akinlar, C., Topal, C.: EDLines: A real-time line segment detector with a false detection control. Pattern Recognition Letters **32**(13), 1633–1642 (2011). https://doi.org/10.1016/j.patrec.2011.06.001, code available at `https://github.com/CihanTopal/ED_Lib`
3. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000), `https://opencv.org/`
4. Burns, J.B., Hanson, A.R., Riseman, E.M.: Extracting straight lines. IEEE Transactions on Pattern Analysis and Machine Intelligence **8**(4), 425–455 (1986). https://doi.org/10.1109/TPAMI.1986.4767808
5. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **8**(6), 679–698 (1986). https://doi.org/10.1109/TPAMI.1986.4767851
6. Casiez, G., Roussel, N., Vogel, D.: 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 2527–2530 (2012). https://doi.org/10.1145/2207676.2208639
7. Castrejon, L., Kundu, K., Urtasun, R., Fidler, S.: Annotating object instances with a polygon-rnn. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5230–5238 (2017). https://doi.org/10.1109/cvpr.2017.477
8. Chazalon, J., Carlinet, E., Chen, Y., Perret, J., Duménieu, B., Mallet, C., Géraud, T., Nguyen, V., Nguyen, N., Baloun, J., Lenc, L., , Král, P.: Icdar 2021 competition on historical map segmentation. In: Proceedings of the 16th International Conference on Document Analysis and Recognition (ICDAR'21). Lausanne, Switzerland (2021). https://doi.org/10.1007/978-3-030-86337-1_46
9. Cho, N.G., Yuille, A., Lee, S.W.: A Novel Linelet-Based Representation for Line Segment Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(5), 1195–1208 (May 2018). https://doi.org/10.1109/tpami.2017.2703841, code available at `https://github.com/NamgyuCho/Linelet-code-and-YorkUrban-LineSegment-DB`
10. Dai, X., Gong, H., Wu, S., Yuan, X., Yi, M.: Fully convolutional line parsing. Neurocomputing **506**, 1–11 (2022). https://doi.org/10.1016/j.neucom.2022.07.026, code available at `https://github.com/Delay-Xili/F-Clip`
11. Denis, P., Elder, J.H., Estrada, F.J.: Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) Computer Vision – ECCV 2008. pp. 197–210. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88688-4_15
12. Desolneux, A., Moisan, L., Morel, J.M.: From gestalt theory to image analysis: a probabilistic approach, vol. 34. Springer Science & Business Media (2007). https://doi.org/10.1007/978-0-387-74378-3
13. Galamhos, C., Matas, J., Kittler, J.: Progressive probabilistic Hough transform for line detection. In: Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 554–560. IEEE Comput. Soc, Fort

Collins, CO, USA (1999). https://doi.org/10.1109/cvpr.1999.786993, code available at `https://github.com/rmenke/ppht`
14. Grompone von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: LSD: A Fast Line Segment Detector with a False Detection Control. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(4), 722–732 (Apr 2010). https://doi.org/10.1109/TPAMI.2008.300
15. He, J., Zhang, S., Yang, M., Shan, Y., Huang, T.: Bi-directional cascade network for perceptual edge detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3823–3832. IEEE (2019). https://doi.org/10.1109/CVPR.2019.00395
16. Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., Ma, Y.: Learning to parse wireframes in images of man-made environments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 626–635 (2018). https://doi.org/10.1109/cvpr.2018.00072
17. Illingworth, J., Kittler, J.: A survey of the hough transform. Computer vision, graphics, and image processing **44**(1), 87–116 (1988). https://doi.org/10.1016/0734-189x(88)90071-0
18. Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering **82**(1), 35–45 (Mar 1960). https://doi.org/10.1115/1.3662552
19. Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P.: Panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9404–9413 (2019). https://doi.org/10.1109/cvpr.2019.00963
20. Kiryati, N., Eldar, Y., Bruckstein, A.M.: A probabilistic Hough transform. Pattern recognition **24**(4), 303–316 (1991). https://doi.org/10.1016/0031-3203(91)90073-e
21. Kultanen, P., Xu, L., Oja, E.: Randomized hough transform (rht). In: Proceedings. 10th International Conference on Pattern Recognition. vol. 1, pp. 631–635. IEEE (1990). https://doi.org/10.1109/ICPR.1990.118177
22. LaViola, J.J.: Double exponential smoothing: an alternative to kalman filter-based predictive tracking. In: Proceedings of the workshop on Virtual environments 2003. pp. 199–206 (2003). https://doi.org/10.1145/769953.769976
23. Leplumey, I., Camillerapp, J., Queguiner, C.: Kalman filter contributions towards document segmentation. In: Proceedings of 3rd International Conference on Document Analysis and Recognition. vol. 2, pp. 765–769. IEEE (1995). https://doi.org/10.1109/icdar.1995.602015
24. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 936–944 (2017). https://doi.org/10.1109/CVPR.2017.106
25. Lu, X., Yao, J., Li, K., Li, L.: Cannylines: A parameter-free line segment detector. In: IEEE International Conference on Image Processing (ICIP). pp. 507–511. IEEE (2015). https://doi.org/10.1109/icip.2015.7350850, code available at `https://cvrs.whu.edu.cn/cannylines/`
26. Mukhopadhyay, P., Chaudhuri, B.B.: A survey of Hough Transform. Pattern Recognition **48**(3), 993–1010 (2015). https://doi.org/10.1016/j.patcog.2014.08.027
27. Poulain d'Andecy, V., Camillerapp, J., Leplumey, I.: Kalman filtering for segment detection: application to music scores analysis. In: Proceedings of 12th International Conference on Pattern Recognition. vol. 1, pp. 301–305 vol.1 (Oct 1994). https://doi.org/10.1109/ICPR.1994.576283
28. Poulain d'Andecy, V., Camillerapp, J., Leplumey, I.: Analyse de partitions musicales. Traitement du Signal **12**(6), 653–661 (1995)

29. Pu, M., Huang, Y., Liu, Y., Guan, Q., Ling, H.: EDTER: Edge detection with transformer. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1392–1402. IEEE (2022). https://doi.org/10.1109/CVPR52688.2022.00146
30. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 28. Curran Associates, Inc. (2015), https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf
31. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 234–241. Springer (2015). https://doi.org/10.1007/978-3-319-24574-4_28
32. Sobel, I., Feldman, G.: An isotropic 3x3 image gradient operator (2015). https://doi.org/10.13140/RG.2.1.1912.4965
33. Suárez, I., Buenaposada, J.M., Baumela, L.: Elsed: Enhanced line segment drawing. Pattern Recognition **127**, 108619 (2022). https://doi.org/10.1016/j.patcog.2022.108619, code available at https://github.com/iago-suarez/ELSED
34. Visani, M., Kieu, V., Fornés, A., Journet, N.: Icdar 2013 music scores competition: Staff removal. In: 2013 12th International Conference on Document Analysis and Recognition. pp. 1407–1411. IEEE (2013). https://doi.org/10.1109/icdar.2013.284
35. Welch, G., Bishop, G.: An Introduction to the Kalman Filter. techreport TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3175 (2006), https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf
36. Xie, S., Tu, Z.: Holistically-nested edge detection. In: IEEE International Conference on Computer Vision (ICCV). pp. 1395–1403 (2015). https://doi.org/10.1109/ICCV.2015.164
37. Xu, Y., Xu, W., Cheung, D., Tu, Z.: Line segment detection using transformers without edges. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4257–4266 (2021). https://doi.org/10.1109/cvpr46437.2021.00424, code available at https://github.com/mlpc-ucsd/LETR
38. Xue, N., Bai, S., Wang, F., Xia, G.S., Wu, T., Zhang, L.: Learning attraction field representation for robust line segment detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1595–1603 (2019). https://doi.org/10.1109/cvpr.2019.00169, code available at https://github.com/cherubicXN/afm_cvpr2019
39. Zhang, Y., Wei, D., Li, Y.: AG3line: Active grouping and geometry-gradient combined validation for fast line segment extraction. Pattern Recognition **113**, 107834 (2021). https://doi.org/10.1016/j.patcog.2021.107834, code available at https://github.com/weidong-whu/AG3line
40. Zhou, Y., Qi, H., Ma, Y.: End-to-end wireframe parsing. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 962–971 (2019). https://doi.org/10.1109/iccv.2019.00105, code available at https://github.com/zhou13/lcnn