Qu'est-ce qu'est l'*Handwritten Text Recognition* ou, traduit en français, la reconnaissance du texte écrit à la main ? L'*Handwritten Text Recognition* (HTR) est l'un des meilleurs approches aujourd'hui à prédire du texte à partir d'une image numérique. Pour expliquer comment l'HTR se réalise, il faut exposer en premier temps qu'est-ce qu'une image numérique. En sachant à quoi ressemble la saisie d'un logiciel HTR, qui fait la prédiction du texte sur l'image, on comprend mieux le défi dont s'occupaient des chercheurs pendant près d'un siècle.

1 Les objectifs de l'HTR

1.1 Le contour

L'un des objectifs de l'HTR est d'imiter l'œil humain et reconnaître les lignes et les points d'une écriture sur une image numérisée du texte. Une image numérique se compose d'un quadrillage des carrés qui s'appellent des pixels. Venant de l'anglais, le mot pixel veut dire *picture element* et il décrit l'élément le plus minimal d'une image numérique. Les pixels sont stockés dans un format *bitmap* ou BMP et chaque pixel peut compter 1 bit, 4 bits, 8 bits, ou 24 bits selon l'encodage de l'image. Quelque soit l'encodage, chaque pixel n'a qu'une seule couleur. Vu ensemble, un groupe de pixels peut donner l'impression des courbes d'un objet ou d'un caractère. L'exemple de figure 1 pourrait donc montrer la diagonale de la lettre « A » écrite en crayon rouge.

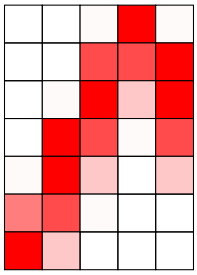


FIGURE 1 – Une couleur per pixel

Un scanner encode l'image en décomposant une image en les unités de pixel et en donnant à chaque pixel un tableau de trois entiers. (cf. Figure 2) Le premier entier qui se trouve dans la donnée tripartite d'un pixel déclare le degré de la couleur rouge à rendre dans la carré. Le deuxième entier déclare le degré de la couleur vert et le troisième de la couleur bleu. En visionnant de loin un groupe de ces carrés, l'œil humain arrive à distinguer les formes. L'HTR vise à reproduire le même résultat et reconnaître les points contiguës d'un caractère.

255,255,255	255,255,255	255,250,250	255,0,0	255,250,250
255,255,255	255,255,255	255,75,75	255,75,75	255,0,0
255,255,255	255,250,250	255,0,0	255,200,200	255,0,0
255,255,255	255,0,0	255,75,75	255,250,250	255,75,75
255,250,250	255,0,0	255,200,200	255,255,255	255,200,200
255,125,125	255,75,75	255,250,250	255,255,255	255,255,255
255,0,0	255,200,200	255,255,255	255,255,255	255,255,255

FIGURE 2 – Donnée tripartite portant sur le degré du rouge, du vert, et du bleu

1.2 Le masque

L'autre objectif de l'HTR est de distinguer les limites et regrouper les composants d'un caractère. Même si un modèle HTR arrivait à reconnaître toutes les lignes de la lettre A, il n'arriverait pas pourtant à reconnaître que les lignes se constituent un caractère sans les rassembler et classifier comme un objet contiguë. Il faut donc tracer la limite d'un caractère dans une entité qui s'appelle un « masque ». [1]

1. Denis COQUENET, Clément CHATELAIN et Thierry PAQUET. « Handwritten Text Recognition : From Isolated Text Lines to Whole Documents ». In : *ORASIS 2021*. Saint Ferréol, France : Centre National de la Recherche Scientifique [CNRS], sept. 2021. URL : `https://hal.archives-ouvertes.fr/hal-03339648` (visité le 04/08/2022).

1.3 Allier les deux objectifs

Un modèle HTR doit donc réaliser ces deux objectifs : identifier les contours d'un caractère et se rende compte que ces points contiguës constituent un objet contiguë. Cet objet contiguë permet au modèle reconnaître du texte. La reconnaissance du texte n'est pas de magie. Elle compte sur la constitution des objets ou des *masques* à partir des pixels. Le modèle HTR traite ces objets comme des matrices, qui s'expliquera prochainement, et peut donc réaliser des calculs qui lui permet prédire du texte.

2 Les origines de l'HTR

L'*Handwritten Text Recognition* a évolué à partir de l'*Optical Character Recognition*. Les origines de cette dernière méthode atteignent jusqu'au dix-neuvième siècle. Mais les progrès les plus importants à la technologie éventuelle dont le projet *Gallic(orpor)a* profite ont eu lieu il y a plus qu'un demi-siècle aux États-Unis. Le développement des logiciels OCR a été réalisé dans le cadre de l'amélioration du traitement en masse des données numérisées. Dans les années soixante, ce besoin de gérer et traiter des quantités vastes de données s'est fait senti largement dans les grandes sociétés, surtout les banques. Mais la numérisation des journaux et le traitement en masse des lettres à la poste étaient aussi quelqu'uns des contextes importants du développement de la reconnaissance du texte sur l'image.

Dans les années mille neuf cent soixante, plus en plus de sociétés voulaient adopter le traitement des données assisté par ordinateur. Ce traitement a exigé l'encodage des données dans un format directement exploitable par ordinateur. En gros il y avait deux moyens de la saisie des données. En premier temps, il y avait la carte perforée dans laquelle une machine ferait des trous qu'un ordinateur savait lire. En deuxième temps, il y avait la bande magnétique sur laquelle une machine imprimerait des bytes, la plus petite unité exploitable par ordinateur.

L'*Optical Character Recognition* comptait sur ce dernier moyen ainsi qu'une autre technologie, la numérisation des images par scanner. L'atout de l'OCR est qu'en principe sa saisie de données se fait automatiquement. À l'époque, les autres moyens d'encoder les données textuelles exigeaient la saisie manuelle des données. Dans le cas des cartes perforées, un individu lirait un document et taperait son texte à la perforatrice à clavier. Le *Magnetic Tape/Selectric Typewriter* (MT/ST), développé par l'entreprise américaine IBM en 1964, comptait sur la même saisie manuelle des données mais il les a encodé sur une bande magnétique au lieu d'une carte perforée.

En 1971, le chercheur Ben R. Schneider a comparé l'OCR et ces deux autres moyens d'encodage du texte.[2] Schneider a déterminé que l'OCR n'avait pas en-

2. Ben R. Schneider. « The Production of Machine-Readable Text : Some of the Variables ». In : *Computers and the Humanities* 6.1 (sept. 1971), p. 39-47. ISSN : 0010-4817, 1572-8412. DOI : 10.1007/BF02402324. URL : http://link.springer.com/10.1007/BF02402324 (visité le 02/08/2022).

core surpassé l'appareil MT/ST d'IBM parce que, malgré sa saisie automatique de données, il exigeait toujours beaucoup de correction à la main. Contrairement aux deux autres méthodes, puisque les données d'entrée du traitement OCR n'étaient pas créées par un humain mais par un scanner, l'OCR n'a pas permis de la correction lors de la saisie. L'appareil MT/ST avait donc une exactitude supérieure à l'OCR à l'époque.

En outre, un logiciel OCR était limité par son jeu de données de polices, puisqu'il prédit du texte en faisant un comparaison entre un caractère qu'il a mémorisé et le motif qu'il a reconnu sur l'image. Même aujourd'hui l'OCR se distingue de son successeur l'HTR par le fait qu'il compte sur une base de données des polices. Contraire à l'OCR dans les années soixante-dix, l'appareil MT/ST pourrait encoder des documents des diverses polices en comptant sur le discernement d'un être humain lors de la saisie des données.

En 1977, ayant vu le progrès de la technologie, Gian Piero Zarri a résumé l'état de l'OCR. Il a remarqué que la reconnaissance du texte sur les manuscrits n'était pas encore faisable.

> Rappelons que la reconnaissance optique des caractères permet la lecture directe du texte par un « scanner » qui se charge d'effectuer le transfert sur bande magnétique ; le texte doit être composé avec des caractères de type « imprimerie » ou « machine à écrire », car la lecture de caractères « manuels » ne semble pas encore actuellement complètement sortie de la phase expérimentale. [3]

Il y a deux ans avant l'appréciation de Zarri, le chercheur américain Ray Kurzweil a produit son lecture *Kurzweil* ou le *Kurzweil Reading Machine* (KRM). L'appareil a avancé l'OCR en pouvant reconnaître plusieurs polices. [4] Cependant, comme dit Zarri, l'OCR restait sous la dépendance de la reconnaissance des polices et donc ne pouvait pas encore parvenir à la prédiction du texte écrit, ce qui a poussé le développement de l'*Handwritten Text Recognition*.

2.1 La binarisation

Afin de reconnaître la police d'un caractère, les logiciels OCR du XXe siècle avaient besoin d'un processus préliminaire qui s'appelle la binarisation. Cette étape est toujours nécessaire pour l'HTR puisque le logiciel, quelque soit sa manière de reconnaître des motifs dans une image, a besoin de relever d'un tableau de pixels les contours d'un caractère. La binarisation trie les pixels d'une image en deux classes : l'arrière-plan (*background* en anglais) et le premier plan (*foreground* en anglais).

3. Gian Piero Zarri. « Quelques aspects techniques de l'exploitation informatique des documents textuels : saisie des données et problèmes de sortie ». In : *Publications de l'École Française de Rome* 31.1 (1977), p. 399-413. URL : `https://www.persee.fr/doc/efr_0000-0000_1977_act_31_1_2286` (visité le 01/08/2022).

4. Gregory Goodrich. « Kurzweil Reading Machine : A Partial Evaluation of Its Optical Character Recognition Error Rate ». In : *Journal of Visual Impairment and Blindness* (12 jan. 1979).

2.1.1 Le niveau de gris

Normalement pour binariser une image, on veut remplacer la valeur composée d'un pixel, c'est-à-dire les trois degrés du rouge, du vert, et du bleu, avec la valeur simple d'un décimal. Ce décimal veut représenter l'échelle des gris dans le pixel. En anglais, ce traitement s'appelle le *grayscale* ou le niveau de gris en français. Aujourd'hui les langages de programmation ont souvent des librairies qui fournissent des méthodes pour rendre une image en niveau de gris automatiquement. Mais dans la Figure 3, nous donnons un calcul simple qui sert à montrer en exemple le traitement niveau de gris. On commence toujours avec la donnée tripartite du pixel, qu'on veut remplacer par une seule valeur. Représentons chaque partie de la donnée du pixel par les variables p :

$$p_1, p_2, p_3$$

En premier temps, on prend la moyenne des degrés de la couleur, c'est-à-dire la moyenne de p ou $\bar{p}$. La Figure 4a montre le résultat de ce calcul superposé à l'image originale.

$$\sum_{i=1}^{3} p_i = p_1, p_2, p_3$$

Ensuite, on récupère $\bar{p}$ et la divise par la valeur maximale du p, qui est 255 parce que le degré du rouge, vert, ou bleu d'un pixel ne monte qu'à 255. La Figure 4b montre le résultat de ce calcul.

Donnée tripartite du pixel p_1, p_2, p_3	Moyenne du pixel $\sum_{i=1}^{3} p_i$	Valeur niveau de gris du pixel $\frac{\sum_{i=1}^{3} p_i}{\max\{p_i\}}$
255,000,000	$\bar{p} = \frac{255+0+0}{3} = 85$	$\frac{\bar{p}}{255} = 0.33$
255,075,075	$\bar{p} = \frac{255+75+75}{3} = 135$	$\frac{\bar{p}}{255} = 0.53$
255,125,125	$\bar{p} = \frac{255+125+125}{3} = 168.33$	$\frac{\bar{p}}{255} = 0.66$
255,200,200	$\bar{p} = \frac{255+200+200}{3} = 218.33$	$\frac{\bar{p}}{255} = 0.86$
255,250,250	$\bar{p} = \frac{255+220+220}{3} = 251.67$	$\frac{\bar{p}}{255} = 0.99$
255,255,255	$\bar{p} = \frac{255+255+255}{3} = 255$	$\frac{\bar{p}}{255} = 1.00$

FIGURE 3 – Évaluation des pixels

2.1.2 Le seuil

Il y a plusieurs techniques de binarisation mais toute a besoin d'un seuil (*threshold* en anglais). Le seuil nous permet à trier les pixels en les deux classes : le *background* et le *foreground*. Nos yeux font cette étape facilement, mais un

ordinateur a besoin d'un algorithme. L'un des algorithmes le plus courant pour définir le seuil a été élaboré en 1979 par le chercheur Nobuyuki Otsu. [5]

La méthode Otsu de seuillage reste toujours courante dans l'HTR [6] et elle fait partie de la librairie Python `numpy`. [7] Elle examine la variance entre les deux classes (*background* et *foreground*) pour déterminer un seuil idéal pour le jeu de données. Visualisées dans un histogramme, comme on voit dans la Figure 4, les données d'un jeu de pixels devraient se lever dans deux sommets et, idéalement, une baisse profonde devrait les diviser. Cette variance veut dire qu'il y a dans l'image une distinction importante entre les contours d'un caractère et l'arrière-plan de l'image. La méthode de seuillage examine la variance entre ces deux classes, le contour et l'arrière-plan, pour générer un seuil adapté aux données.

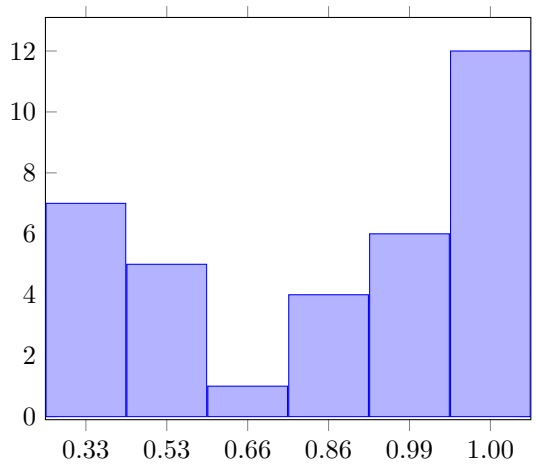


FIGURE 4 – Histogramme des valeurs niveau de gris des pixels

5. Nobuyuki OTSU. « A Threshold Selection Method from Gray-Level Histograms ». In : *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (jan. 1979), p. 62-66.

6. Imran-Ahmed SIDDIQI, Florence CLOPPET et Nicole VINCENT. « Writing Property Descriptors : A Proposal for Typological Groupings ». In : *Gazette du livre médiéval* 56.1 (2011), p. 42-57. DOI : `10.3406/galim.2011.1981`. URL : `https://www.persee.fr/doc/galim_0753-5015_2011_num_56_1_1981` (visité le 02/08/2022).

7. *Numpy : NumPy Is the Fundamental Package for Array Computing with Python.* Version 1.23.1. URL : `https://www.numpy.org` (visité le 04/08/2022).

255	255	251.67	85	251.67
255	255	135	135	85
255	251.67	85	218.33	85
255	85	135	251.67	135
251.67	85	218.33	255	218.33
168.33	135	251.67	255	255
85	218.33	255	255	255

(a) La moyenne de chaque pixel

1.00	1.00	0.98	0.33	1.98
1.00	1.00	0.53	0.53	0.33
1.00	0.98	0.33	0.86	0.33
1.00	0.33	0.53	0.98	0.53
0.98	0.33	0.86	1.00	0.86
0.66	0.53	0.98	1.00	1.00
0.33	0.86	1.00	1.00	1.00

(b) L'image en niveau de gris

0	0	0	1	0
0	0	1	1	1
0	0	1	0	1
0	1	1	0	1
0	1	0	0	0
1	1	0	0	0
1	0	0	0	0

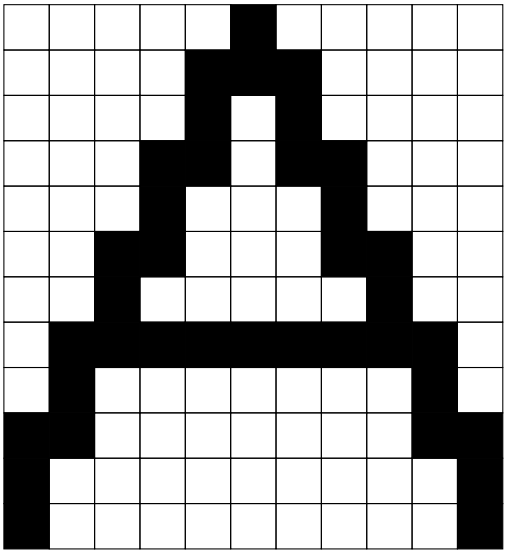
(c) L'image binarisée

FIGURE 5 – La binarisation

Le seuil sert à transformer la valeur numérique de chaque pixel soit en 0, soit en 1. Les pixels dont la valeur tombe au-dessous du seuil prennent la valeur 0 ; les autres, étant déterminés de faire partie du contour du caractère, prennent la valeur 1. La Figure 4c montre un exemple du résultat de ce triage. Ainsi, les logiciels OCR et HTR peuvent analyser les valeurs identiques et contiguës dans les données de l'image. Mais le logiciel n'est pas encore prêt à percevoir l'occurence d'un caractère.

2.1.3 Le masque

Ayant relevé sur l'image binarisée les points contiguës, c'est-à-dire les lignes, le logiciel doit ensuite reconnaître les objets qui s'en constituent. Les composants d'un caractère doivent s'encadrer dans un masque que le logiciel OCR ou HTR rendra comme une matrice binaire. (cf. Figure 5b) Grâce aux méthodes de segmentation, un logiciel produira les masques des glyphes, dont les composants qu'il a reconnus. En plus, le logiciel remarquera quels glyphes font partie de quels segments et quelles lignes de texte. Les processus de segmentation comptent sur la reconnaissance des espaces qui désunissent les groupes de points contiguës.



(a) Le masque de la lettre *A*, sur l'image binarisée

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

(b) La matrice du masque
de la lettre A

2.2 L'intelligence artificielle

Pourquoi transforme-t-on les pixels d'un caractère en une matrice, telle que celle dans la Figure 5b ? À la base, l'ordinateur est une calculatrice. Le format calculable de la matrice permet au logiciel OCR ou HTR traiter les composants d'une image numérisée. En outre, les découvertes majeures de l'intelligence artificielle ont donné aux logiciels OCR et HTR le pouvoir à prédire le texte représenté par les matrices. En faisant des calculs probabilistes, grâce aux statistiques qu'ils savoir apprennent et mettre à jour lors de leur entraînement, les logiciels soutenus par l'IA peuvent livrer les résultats de la reconnaissance du texte plus exacts que jamais.