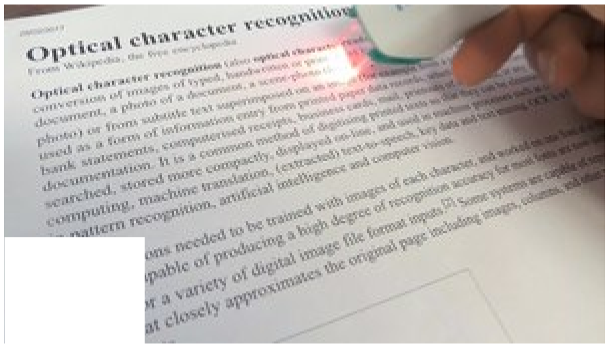
WIKIPEDIA

Optical character recognition

Optical character recognition or **optical character reader** (**OCR**) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast).[1]


Video of the process of scanning and real-time optical character recognition (OCR) with a portable scanner.

Widely used as a form of information entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs.[2] Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

Contents

History

Early optical character recognition may be traced to technologies involving telegraphy and creating reading devices for the blind.[3] In 1914, Emanuel Goldberg developed a machine that read characters and converted them into standard telegraph code. Concurrently, Edmund Fournier d'Albe developed the Optophone, a handheld scanner that when moved across a printed page, produced tones that corresponded to specific letters or characters.[4]

In the late 1920s and into the 1930s Emanuel Goldberg developed what he called a "Statistical Machine" for searching microfilm archives using an optical code recognition system. In 1931 he was granted USA Patent number 1,838,389 for the invention. The patent was acquired by IBM.

With the advent of smart-phones and smartglasses, OCR can be used in internet connected mobile device applications that extract text captured using the device's camera. These devices that do not have OCR functionality built into the operating system will typically use an OCR API to extract the text from the image file captured and provided by the device.[5][6] The OCR API returns the extracted text, along with information about the location of the detected text in the original image back to the device app for further processing (such as text-to-speech) or display.

Blind and visually impaired users

In 1974, Ray Kurzweil started the company Kurzweil Computer Products, Inc. and continued development of omni-font OCR, which could recognize text printed in virtually any font (Kurzweil is often credited with inventing omni-font OCR, but it was in use by companies, including CompuScan, in the late 1960s and 1970s[3][7]). Kurzweil decided that the best application of this technology would be to create a reading machine for the blind, which would allow blind people to have a computer read text to them out loud. This device required the invention of two enabling technologies – the CCD flatbed scanner and the text-to-speech synthesizer. On January 13, 1976, the successful finished product was unveiled during a widely reported news conference headed by Kurzweil and the leaders of the National Federation of the Blind. In 1978, Kurzweil Computer Products began selling a commercial version of the optical character recognition computer program. LexisNexis was one of the first customers, and bought the program to upload legal paper and news documents onto its nascent online databases. Two years later, Kurzweil sold his company to Xerox, which had an interest in further commercializing paper-to-computer text conversion. Xerox eventually spun it off as Scansoft, which merged with Nuance Communications. The research group headed by A. G. Ramakrishnan at the Medical intelligence and language engineering lab, Indian Institute of Science, has developed PrintToBraille tool, an open source GUI front end[8] that can be used by any OCR to convert scanned images of printed books to Braille books.

In the 2000s, OCR was made available online as a service (WebOCR), in a cloud computing environment, and in mobile applications like real-time translation of foreign-language signs on a smartphone.

Various commercial and open source OCR systems are available for most common writing systems, including Latin, Cyrillic, Arabic, Hebrew, Indic, Bengali (Bangla), Devanagari, Tamil, Chinese, Japanese, and Korean characters.

Applications

OCR engines have been developed into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, check OCR, legal billing document OCR.

They can be used for:

- Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt
- Automatic number plate recognition
- In airports, for passport recognition and information extraction
- Automatic insurance documents key information extraction
- Extracting business card information into a contact list[9]
- More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg
- Make electronic images of printed documents searchable, e.g. Google Books
- Converting handwriting in real time to control a computer (pen computing)
- Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR.[10][11][12] The purpose can also be to test the robustness of CAPTCHA anti-bot systems.
- Assistive technology for blind and visually impaired users

Types

- Optical character recognition (OCR) – targets typewritten text, one glyph or character at a time.
- Optical word recognition – targets typewritten text, one word at a time (for languages that use a space as a word divider). (Usually just called "OCR".)
- Intelligent character recognition (ICR) – also targets handwritten printscript or cursive text one glyph or character at a time, usually involving machine learning.
- Intelligent word recognition (IWR) – also targets handwritten printscript or cursive text, one word at a time. This is especially useful for languages where glyphs are not separated in cursive script.

OCR is generally an "offline" process, which analyses a static document. Handwriting movement analysis can be used as input to handwriting recognition.[13] Instead of merely using the shapes of glyphs and words, this technique is able to capture motions, such as the order in which segments are drawn, the direction, and the pattern of putting the pen down and lifting it. This additional information can make the end-to-end process more accurate. This technology is also known as "on-line character recognition", "dynamic character recognition", "real-time character recognition", and "intelligent character recognition".

Techniques

Pre-processing

OCR software often "pre-processes" images to improve the chances of successful recognition. Techniques include:[14]

- De-skew – If the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counterclockwise in order to make lines of text perfectly horizontal or vertical.
- Despeckle – remove positive and negative spots, smoothing edges
- Binarisation – Convert an image from color or greyscale to black-and-white (called a "binary image" because there are two colors). The task of binarisation is performed as a simple way of separating the text (or any other desired image component) from the background.[15] The task of binarisation itself is necessary since most commercial recognition algorithms work only on binary images since it proves to be simpler to do so.[16] In addition, the effectiveness of the binarisation step influences to a significant extent the quality of the character recognition stage and the careful decisions are made in the choice of the binarisation employed for a given input image type; since the quality of the binarisation method employed to obtain the binary result depends on the type of the input image (scanned document, scene text image, historical degraded document etc.).[17][18]
- Line removal – Cleans up non-glyph boxes and lines
- Layout analysis or "zoning" – Identifies columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-column layouts and tables.
- Line and word detection – Establishes baseline for word and character shapes, separates words if necessary.
- Script recognition – In multilingual documents, the script may change at the level of the words and hence, identification of the script is necessary, before the right OCR can be invoked to handle the specific script.[19]

- Character isolation or "segmentation" – For per-character OCR, multiple characters that are connected due to image artifacts must be separated; single characters that are broken into multiple pieces due to artifacts must be connected.
- Normalize aspect ratio and scale[20]

Segmentation of fixed-pitch fonts is accomplished relatively simply by aligning the image to a uniform grid based on where vertical grid lines will least often intersect black areas. For proportional fonts, more sophisticated techniques are needed because whitespace between letters can sometimes be greater than that between words, and vertical lines can intersect more than one character.[21]

Character recognition

There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters.[22]

Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation". This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.

Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. The extraction features reduces the dimensionality of the representation and makes the recognition process computationally efficient. These features are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes. General techniques of feature detection in computer vision are applicable to this type of OCR, which is commonly seen in "intelligent" handwriting recognition and indeed most modern OCR software.[23] Nearest neighbour classifiers such as the k-nearest neighbors algorithm are used to compare image features with stored glyph features and choose the nearest match.[24]

Software such as Cuneiform and Tesseract use a two-pass approach to character recognition. The second pass is known as "adaptive recognition" and uses the letter shapes recognized with high confidence on the first pass to recognize better the remaining letters on the second pass. This is advantageous for unusual fonts or low-quality scans where the font is distorted (e.g. blurred or faded).[21]

The OCR result can be stored in the standardized ALTO format, a dedicated XML schema maintained by the United States Library of Congress.

For a list of optical character recognition software see Comparison of optical character recognition software.

Post-processing

OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document.[14] This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy.[21]

The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

"Near-neighbor analysis" can make use of co-occurrence frequencies to correct errors, by noting that certain words are often seen together.[25] For example, "Washington, D.C." is generally far more common in English than "Washington DOC".

Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy.

The Levenshtein Distance algorithm has also been used in OCR post-processing to further optimize results from an OCR API.[26]

Application-specific optimizations

In recent years, the major OCR technology providers began to tweak OCR systems to deal more efficiently with specific types of input. Beyond an application-specific lexicon, better performance may be had by taking into account business rules, standard expression, or rich information contained in color images. This strategy is called "Application-Oriented OCR" or "Customized OCR", and has been applied to OCR of license plates, invoices, screenshots, ID cards, driver licenses, and automobile manufacturing.

The New York Times has adapted the OCR technology into a proprietary tool they entitle, *Document Helper*, that enables their interactive news team to accelerate the processing of documents that need to be reviewed. They note that it enables them to process what amounts to as many as 5,400 pages per hour in preparation for reporters to review the contents.[27]

Workarounds

There are several techniques for solving the problem of character recognition by means other than improved OCR algorithms.

Forcing better input

Special fonts like OCR-A, OCR-B, or MICR fonts, with precisely specified sizing, spacing, and distinctive character shapes, allow a higher accuracy rate during transcription in bank check processing. Ironically however, several prominent OCR engines were designed to capture text in popular fonts such as Arial or Times New Roman, and are incapable of capturing text in these fonts that are specialized and much different from popularly used fonts. As Google Tesseract can be trained to recognize new fonts, it can recognize OCR-A, OCR-B and MICR fonts[28].

"Comb fields" are pre-printed boxes that encourage humans to write more legibly – one glyph per box.[25] These are often printed in a "dropout color" which can be easily removed by the OCR system.[25]

Palm OS used a special set of glyphs, known as "Graffiti" which are similar to printed English characters but simplified or modified for easier recognition on the platform's computationally limited hardware. Users would need to learn how to write these special glyphs.

Zone-based OCR restricts the image to a specific part of a document. This is often referred to as "Template OCR".

Crowdsourcing

Crowdsourcing humans to perform the character recognition can quickly process images like computer-driven OCR, but with higher accuracy for recognizing images than is obtained with computers. Practical systems include the Amazon Mechanical Turk and reCAPTCHA. The National Library of Finland has developed an online interface for users to correct

OCRed texts in the standardized ALTO format.[29] Crowd sourcing has also been used not to perform character recognition directly but to invite software developers to develop image processing algorithms, for example, through the use of rank-order tournaments.[30]

Accuracy

Commissioned by the U.S. Department of Energy (DOE), the Information Science Research Institute (ISRI) had the mission to foster the improvement of automated technologies for understanding machine printed documents, and it conducted the most authoritative of the *Annual Test of OCR Accuracy* from 1992 to 1996.[31]

Recognition of Latin-script, typewritten text is still not 100% accurate even where clear imaging is available. One study based on recognition of 19th- and early 20th-century newspaper pages concluded that character-by-character OCR accuracy for commercial OCR software varied from 81% to 99%;[32] total accuracy can be achieved by human review or Data Dictionary Authentication. Other areas—including recognition of hand printing, cursive handwriting, and printed text in other scripts (especially those East Asian language characters which have many strokes for a single character)—are still the subject of active research. The MNIST database is commonly used for testing systems' ability to recognise handwritten digits.

Accuracy rates can be measured in several ways, and how they are measured can greatly affect the reported accuracy rate. For example, if word context (basically a lexicon of words) is not used to correct software finding non-existent words, a character error rate of 1% (99% accuracy) may result in an error rate of 5% (95% accuracy) or worse if the measurement is based on whether each whole word was recognized with no incorrect letters.[33]

An example of the difficulties inherent in digitizing old text is the inability of OCR to differentiate between the "long s" and "f" characters.[34]

Web-based OCR systems for recognizing hand-printed text on the fly have become well known as commercial products in recent years (see Tablet PC history). Accuracy rates of 80% to 90% on neat, clean hand-printed characters can be achieved by pen computing software, but that accuracy rate still translates to dozens of errors per page, making the technology useful only in very limited applications.

Recognition of cursive text is an active area of research, with recognition rates even lower than that of hand-printed text. Higher rates of recognition of general cursive script will likely not be possible without the use of contextual or grammatical information. For example, recognizing entire words from a dictionary is easier than trying to parse individual characters from script. Reading the *Amount* line of a cheque (which is always a written-out number) is an example where using a smaller dictionary can increase recognition rates greatly. The shapes of individual cursive characters themselves simply do not contain enough information to accurately (greater than 98%) recognize all handwritten cursive script.

Most programs allow users to set "confidence rates". This means that if the software does not achieve their desired level of accuracy, a user can be notified for manual review.

Unicode

Characters to support OCR were added to the Unicode Standard in June 1993, with the release of version 1.1.

Some of these characters are mapped from fonts specific to MICR, OCR-A or OCR-B.

Optical Character Recognition[1][2]
Official Unicode Consortium code chart (https://www.unicode.org/charts/PDF/U2440.pdf) (PDF)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+244x	⑀	⑁	⑂	⑃	⑄	⑅	⑆	⑇	⑈	⑉	⑊					
U+245x																

Notes

1. ^ As of Unicode version 12.0
2. ^ Grey areas indicate non-assigned code points

See also

- AI effect
- Applications of artificial intelligence
- Computational linguistics
- Digital library
- Digital mailroom
- Digital pen
- Institutional repository
- Legibility
- List of emerging technologies
- Live ink character recognition solution
- Magnetic ink character recognition
- Music OCR
- OCR in Indian Languages
- Optical mark recognition
- Outline of artificial intelligence
- Sketch recognition
- Speech recognition
- Voice recording
- Comparison of optical character recognition software
- Tesseract OCR engine

References

1. OnDemand, HPE Haven. "OCR Document" (https://dev.havenondemand.com/apis/ocrdocument#overview).
2. OnDemand, HPE Haven. "undefined" (https://dev.havenondemand.com/docs/ImageFormats.html).
3. Schantz, Herbert F. (1982). *The history of OCR, optical character recognition*. [Manchester Center, Vt.]: Recognition Technologies Users Association. ISBN 9780943072012.
4. d'Albe, E. E. F. (July 1, 1914). "On a Type-Reading Optophone". *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*. **90** (619): 373–375. Bibcode:1914RSPSA..90..373D (http://adsabs.harvard.edu/abs/1914RSPSA..90..373D). doi:10.1098/rspa.1914.0061 (https://doi.org/10.1098%2Frspa.1914.0061).
5. "Extracting text from images using OCR on Android" (https://community.havenondemand.com/t5/Blog/Extracting-text-from-images-using-OCR-on-Android/ba-p/1883). June 27, 2015.
6. "[Tutorial] OCR on Google Glass" (https://community.havenondemand.com/t5/Blog/Tutorial-OCR-on-Google-Glass/ba-p/1164). October 23, 2014.
7. "The History of OCR". *Data Processing Magazine*. **12**: 46. 1970.
8. PrintToBraille Tool. "ocr-gui-frontend" (https://web.archive.org/web/20141225115650/https://code.google.com/p/ocr-gui-frontend/). MILE Lab, Dept of EE, IISc. Archived from the original (https://code.google.com/p/ocr-gui-frontend/) on December 25, 2014. Retrieved December 7, 2014.
9. "[javascript] Using OCR and Entity Extraction for LinkedIn Company Lookup" (https://community.havenondemand.com/t5/Blog/javascript-Using-OCR-and-Entity-Extraction-for-LinkedIn-Company/ba-p/460). July 22, 2014.
10. "How To Crack Captchas" (http://www.andrewt.net/blog/how-to-crack-captchas/). andrewt.net. June 28, 2006. Retrieved June 16, 2013.
11. "Breaking a Visual CAPTCHA" (http://www.cs.sfu.ca/~mori/research/gimpy/). Cs.sfu.ca. December 10, 2002. Retrieved June 16, 2013.

12. John Resig (January 23, 2009). "John Resig – OCR and Neural Nets in JavaScript" (http://ejohn.org/blog/ocr-and-neural-nets-in-javascript/). Ejohn.org. Retrieved June 16, 2013.
13. Tappert, C. C.; Suen, C. Y.; Wakahara, T. (1990). "The state of the art in online handwriting recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **12** (8): 787. doi:10.1109/34.57669 (https://doi.org/10.1109%2F34.57669).
14. "Optical Character Recognition (OCR) – How it works" (https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/). Nicomsoft.com. Retrieved June 16, 2013.
15. Sezgin, Mehmet; Sankur, Bulent (2004). "Survey over image thresholding techniques and quantitative performance evaluation" (http://webdocs.cs.ualberta.ca/~nray1/CMPUT605/track3_papers/Threshold_survey.pdf) (PDF). *Journal of Electronic Imaging*. **13** (1): 146. Bibcode:2004JEI....13..146S (http://adsabs.harvard.edu/abs/2004JEI....13..146S). doi:10.1117/1.1631315 (https://doi.org/10.1117%2F1.1631315). Retrieved May 2, 2015.
16. Gupta, Maya R.; Jacobson, Nathaniel P.; Garcia, Eric K. (2007). "OCR binarisation and image pre-processing for searching historical documents" (http://www.rfai.li.univ-tours.fr/fr/ressources/_dh/DOC/DocOCR/OCRbinarisation.pdf) (PDF). *Pattern Recognition*. **40** (2): 389. doi:10.1016/j.patcog.2006.04.043 (https://doi.org/10.1016%2Fj.patcog.2006.04.043). Retrieved May 2, 2015.
17. Trier, Oeivind Due; Jain, Anil K. (1995). "Goal-directed evaluation of binarisation methods" (http://heim.ifi.uio.no/inf386/trier2.pdf) (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **17** (12): 1191–1201. doi:10.1109/34.476511 (https://doi.org/10.1109%2F34.476511). Retrieved May 2, 2015.
18. Milyaev, Sergey; Barinova, Olga; Novikova, Tatiana; Kohli, Pushmeet; Lempitsky, Victor (2013). "Image binarisation for end-to-end text understanding in natural images" (http://research.microsoft.com/en-us/um/people/pkohli/papers/mbnlk_icdar2013.pdf) (PDF). *Document Analysis and Recognition (ICDAR) 2013*. 12th International Conference on. Retrieved May 2, 2015.
19. Pati, P.B.; Ramakrishnan, A.G. (May 29, 1987). "Word Level Multi-script Identification". *Pattern Recognition Letters*. **29** (9): 1218–1229. doi:10.1016/j.patrec.2008.01.027 (https://doi.org/10.1016%2Fj.patrec.2008.01.027).
20. "Basic OCR in OpenCV | Damiles" (http://blog.damiles.com/2008/11/20/basic-ocr-in-opencv.html). Blog.damiles.com. November 20, 2008. Retrieved June 16, 2013.
21. Ray Smith (2007). "An Overview of the Tesseract OCR Engine" (http://tesseract-ocr.googlecode.com/svn/trunk/doc/tesseracticdar2007.pdf) (PDF). Retrieved May 23, 2013.
22. "OCR Introduction" (http://www.dataid.com/aboutocr.htm). Dataid.com. Retrieved June 16, 2013.
23. "How OCR Software Works" (http://ocrwizard.com/ocr-software/how-ocr-software-works.html). OCRWizard. Retrieved June 16, 2013.
24. "The basic pattern recognition and classification with openCV | Damiles" (http://blog.damiles.com/2008/11/14/the-basic-patter-recognition-and-classification-with-opencv.html). Blog.damiles.com. November 14, 2008. Retrieved June 16, 2013.
25. "How does OCR document scanning work?" (http://www.explainthatstuff.com/how-ocr-works.html). Explain that Stuff. January 30, 2012. Retrieved June 16, 2013.
26. "How to optimize results from the OCR API when extracting text from an image? - Haven OnDemand Developer Community" (https://community.havenondemand.com/t5/Wiki/How-to-optimize-results-from-the-OCR-API-when-extracting-text/ta-p/1656).
27. Fehr, Tiff, *How We Sped Through 900 Pages of Cohen Documents in Under 10 Minutes (https://www.nytimes.com/2019/03/26/reader-center/times-documents-reporters-cohen.html?rref=collection%2Fsectioncollection%2Freader-center&action=click&contentCollection=reader-center&region=rank&module=package&version=highlights&contentPlacement=2&pgtype=sectionfront)*, Times Insider, The New York Times, March 26, 2019
28. "Train Your Tesseract" (http://trainyourtesseract.com/). *Train Your Tesseract*. September 20, 2018. Retrieved September 20, 2018.
29. "What is the point of an online interactive OCR text editor? - Fenno-Ugrica" (http://blogs.helsinki.fi/fennougrica/2014/02/21/ocr-text-editor/). February 21, 2014.

30. Riedl, C.; Zanibbi, R.; Hearst, M. A.; Zhu, S.; Menietti, M.; Crusan, J.; Metelsky, I.; Lakhani, K. (February 20, 2016). "Detecting Figures and Part Labels in Patents: Competition-Based Development of Image Processing Algorithms". *International Journal on Document Analysis and Recognition*. **19** (2): 155. arXiv:1410.6751 (https://arxiv.org/abs/1410.6751). doi:10.1007/s10032-016-0260-8 (https://doi.org/10.1007%2Fs10032-016-0260-8).
31. "Code and Data to evaluate OCR accuracy, originally from UNLV/ISRI" (https://code.google.com/p/isri-ocr-evaluation-tools/). Google Code Archive.
32. Holley, Rose (April 2009). "How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs" (http://www.dlib.org/dlib/march09/holley/03holley.html). D-Lib Magazine. Retrieved January 5, 2014.
33. Suen, C.Y.; Plamondon, R.; Tappert, A.; Thomassen, A.; Ward, J.R.; Yamamoto, K. (May 29, 1987). *Future Challenges in Handwriting and Computer Applications* (http://users.erols.com/rwservices/pens/biblio88.html#Suen88). 3rd International Symposium on Handwriting and Computer Applications, Montreal, May 29, 1987. Retrieved October 3, 2008.
34. Sarantos Kapidakis, Cezary Mazurek, Marcin Werla (2015). *Research and Advanced Technology for Digital Libraries* (https://books.google.com/?id=kEyGCgAAQBAJ&dq=OCR+and+long+s). Springer. p. 257. ISBN 9783319245928. Retrieved April 3, 2018.

External links

- Unicode OCR – Hex Range: 2440-245F (https://www.unicode.org/charts/PDF/U2440.pdf) Optical Character Recognition in Unicode
- Annotated bibliography of references to handwriting character recognition and pen computing (http://ruetersward.com/biblio.html)