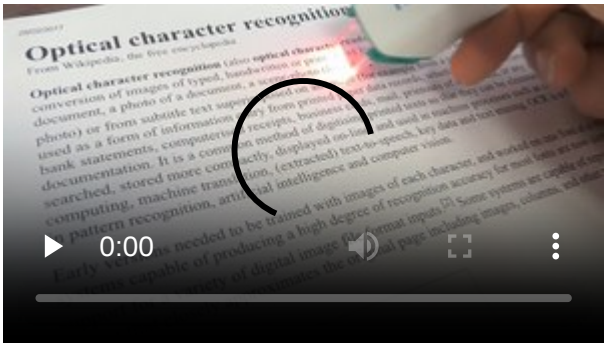
Optical character recognition

Optical character recognition or **optical character reader** (**OCR**) is the electronic or mechanical 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 data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printed 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 online, 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 accuracy for most fonts are now common, and with support for a variety of 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.

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.[4] 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.[5]

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 US Patent number 1,838,389 for the invention. The patent was acquired by IBM.

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][6]) Kurzweil used the technology to create a reading machine for blind people to have a computer read text to them out loud. The device included a CCD-type flatbed scanner and a text-to-speech synthesizer. On January 13, 1976, the 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 eventually spun it off as Scansoft, which merged with Nuance Communications.

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. With the advent of smartphones 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 built-in OCR functionality will typically use an OCR API to extract the text from the image file captured by the device.[7][8] 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.

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 software applications specializing in various subjects such as receipts, invoices, checks, and legal billing documents.

The software can be used for:

- Entering data for business documents, e.g. checks, passports, invoices, bank statements and receipts
- Automatic number-plate recognition
- Passport recognition and information extraction in airports
- Automatically extracting key information from insurance documents
- Traffic-sign recognition[9]
- Extracting business card information into a contact list[10]
- Creating textual versions of printed documents, e.g. book scanning for Project Gutenberg
- Making electronic images of printed documents searchable, e.g. Google Books
- Converting handwriting in real-time to control a computer (pen computing)
- Defeating or testing the robustness of CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR.[11][12][13]

- Assistive technology for blind and visually impaired users
- Writing instructions for vehicles by identifying CAD images in a database that are appropriate to the vehicle design as it changes in real time
- Making scanned documents searchable by converting them to PDFs

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. There are cloud based services which provide an online OCR API service. Handwriting movement analysis can be used as input to handwriting recognition.[14] Instead of merely using the shapes of glyphs and words, this technique is able to capture motion, 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 process more accurate. This technology is also known as "online 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:[15]

- De-skewing – 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.
- Despeckling – removal of positive and negative spots, smoothing edges
- Binarization – conversion of an image from color or greyscale to black-and-white (called a binary image because there are two colors). The task is performed as a simple way of separating the text (or any other desired image component) from the background.[16] The task of binarization is necessary since most commercial recognition algorithms work only on binary images, as it is simpler to do so.[17] In addition, the effectiveness of binarization influences to a significant extent the quality of character recognition, and careful decisions are made in the choice of the binarization employed for a given input image type; since the quality of the method used to obtain the binary result depends on the type of image (scanned document, scene text image, degraded historical document, etc.).[18][19]
- Line removal – Cleaning up non-glyph boxes and lines
- Layout analysis or zoning – Identification of columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-column layouts and tables.

- Line and word detection – Establishment of a baseline for word and character shapes, separating words as 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.[20]
- 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.
- Normalization of aspect ratio and scale[21]

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.[22]

Text recognition

There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters.[23]

- *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 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 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 most modern OCR software.[24] 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.[25]

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 better recognize 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).[22]

As of December 2016, modern OCR software includes Google Docs OCR, ABBYY FineReader, and Transym.[26] Others like OCRopus and Tesseract use neural networks which are trained to recognize whole lines of text instead of focusing on single characters.

A technique known as iterative OCR automatically crops a document into sections based on the page layout. OCR is then performed on each section individually using variable character confidence level thresholds to maximize page-level OCR accuracy. A patent from the United States Patent Office has been issued for this method.[27]

The OCR result can be stored in the standardized ALTO format, a dedicated XML schema maintained by the United States Library of Congress. Other common formats include hOCR and PAGE XML.

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.[15] 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.[22]

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.[28] 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.[29]

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's 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.[30]

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. 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 very 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.[31]

Comb fields are pre-printed boxes that encourage humans to write more legibly – one glyph per box.[28] These are often printed in a dropout color which can be easily removed by the OCR system.[28]

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 that obtained via 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.[32] 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.[33]

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.[35]

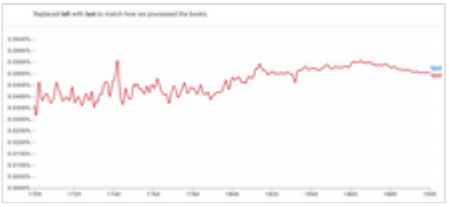
Occurrence of laft and last in Google's n-grams database, in English documents from 1700 to 1900, based on OCR scans for the "English 2009" corpus

Recognition of typewritten, Latin script 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%;[36] 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 recognize 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 (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% or worse if the measurement is based on whether each whole word was recognized with no incorrect letters.[37] Using a large enough dataset is important in a neural-network-based handwriting recognition solutions. On the other hand, producing natural datasets is very complicated and time-consuming.[38]

Occurrence of laft and last in Google's n-grams database, based on OCR scans for the "English 2012" corpus[34]

An example of the difficulties inherent in digitizing old text is the inability of OCR to differentiate between the "long s" and "f" characters.[39][34]



Searching for words with a long S in English 2012 or later are normalized to an S.

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 check (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.

An error introduced by OCR scanning is sometimes termed a *scanno* (by analogy with the term *typo*).[40][41]

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 16.0
2.^ Grey areas indicate non-assigned code points

See also

- AI effect
- Applications of artificial intelligence
- Comparison of optical character recognition software
- Computational linguistics
- Digital library
- Digital mailroom
- Digital pen
- eScriptorium
- 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
- Tesseract OCR engine
- Voice recording

References

1. "OCR Document" (https://web.archive.org/web/20160415060125/https://dev.havenondemand.com/apis/ocrdocument). *Haven OnDemand*. Archived from the original (https://dev.havenondemand.com/apis/ocrdocument#overview) on April 15, 2016.
2. "Supported Media Formats" (https://web.archive.org/web/20160419063444/https://dev.havenondemand.com/docs/ImageFormats.html). *Haven OnDemand*. Archived from the original (https://dev.havenondemand.com/docs/ImageFormats.html) on April 19, 2016.

3. Schantz, Herbert F. (1982). *The history of OCR, optical character recognition* (https://archive.org/details/historyofocropti0000scha). [Manchester Center, Vt.]: Recognition Technologies Users Association. ISBN 9780943072012.
4. Dhavale, Sunita Vikrant (2017). *Advanced Image-Based Spam Detection and Filtering Techniques* (https://books.google.com/books?id=InFxDgAAQBAJ&q=1914+Emanuel+Goldberg&pg=PA91). Hershey, PA: IGI Global. p. 91. ISBN 9781683180142.
5. 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 (https://ui.adsabs.harvard.edu/abs/1914RSPSA..90..373D). doi:10.1098/rspa.1914.0061 (https://doi.org/10.1098%2Frspa.1914.0061).
6. "The History of OCR". *Data Processing Magazine*. **12**: 46. 1970.
7. "Extracting text from images using OCR on Android" (https://web.archive.org/web/20160315001012/https://community.havenondemand.com/t5/Blog/Extracting-text-from-images-using-OCR-on-Android/ba-p/1883). June 27, 2015. Archived from the original (https://community.havenondemand.com/t5/Blog/Extracting-text-from-images-using-OCR-on-Android/ba-p/1883) on March 15, 2016.
8. "[Tutorial] OCR on Google Glass" (https://web.archive.org/web/20160305231423/https://community.havenondemand.com/t5/Blog/Tutorial-OCR-on-Google-Glass/ba-p/1164). October 23, 2014. Archived from the original (https://community.havenondemand.com/t5/Blog/Tutorial-OCR-on-Google-Glass/ba-p/1164) on March 5, 2016.
9. Zeng, Qing-An (2015). *Wireless Communications, Networking and Applications: Proceedings of WCNA 2014* (https://books.google.com/books?id=vCnUCgAAQBAJ). Springer. ISBN 978-81-322-2580-5.
10. "[javascript] Using OCR and Entity Extraction for LinkedIn Company Lookup" (https://web.archive.org/web/20160417145657/https://community.havenondemand.com/t5/Blog/javascript-Using-OCR-and-Entity-Extraction-for-LinkedIn-Company/ba-p/460). July 22, 2014. Archived from the original (https://community.havenondemand.com/t5/Blog/javascript-Using-OCR-and-Entity-Extraction-for-LinkedIn-Company/ba-p/460) on April 17, 2016.
11. "How To Crack Captchas" (http://www.andrewt.net/blog/how-to-crack-captchas/). andrewt.net. June 28, 2006. Retrieved June 16, 2013.
12. "Breaking a Visual CAPTCHA" (http://www.cs.sfu.ca/~mori/research/gimpy/). Cs.sfu.ca. December 10, 2002. Retrieved June 16, 2013.
13. Resig, John (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.
14. 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). S2CID 42920826 (https://api.semanticscholar.org/CorpusID:42920826).
15. "Optical Character Recognition (OCR) – How it works" (https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/). Nicomsoft.com. Retrieved June 16, 2013.
16. Sezgin, Mehmet; Sankur, Bulent (2004). "Survey over image thresholding techniques and quantitative performance evaluation" (https://web.archive.org/web/20151016080410/http://webdocs.cs.ualberta.ca/~nray1/CMPUT605/track3_papers/Threshold_survey.pdf) (PDF). *Journal of Electronic Imaging*. **13** (1): 146. Bibcode:2004JEI....13..146S (https://ui.adsabs.harvard.edu/abs/2004JEI....13..146S). doi:10.1117/1.1631315 (https://doi.org/10.1117%2F1.1631315). Archived from the original (http://webdocs.cs.ualberta.ca/~nray1/CMPUT605/track3_papers/Threshold_survey.pdf) (PDF) on October 16, 2015. Retrieved May 2, 2015.

17. Gupta, Maya R.; Jacobson, Nathaniel P.; Garcia, Eric K. (2007). "OCR binarisation and image pre-processing for searching historical documents" (https://web.archive.org/web/20151016080410/http://www.rfai.li.univ-tours.fr/fr/ressources/_dh/DOC/DocOCR/OCRbinarisation.pdf) (PDF). *Pattern Recognition*. **40** (2): 389. Bibcode:2007PatRe..40..389G (https://ui.adsabs.harvard.edu/abs/2007PatRe..40..389G). doi:10.1016/j.patcog.2006.04.043 (https://doi.org/10.1016%2Fj.patcog.2006.04.043). Archived from the original (http://www.rfai.li.univ-tours.fr/fr/ressources/_dh/DOC/DocOCR/OCRbinarisation.pdf) (PDF) on October 16, 2015. Retrieved May 2, 2015.
18. 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). Archived (https://web.archive.org/web/20151016080411/http://heim.ifi.uio.no/inf386/trier2.pdf) (PDF) from the original on October 16, 2015. Retrieved May 2, 2015.
19. Milyaev, Sergey; Barinova, Olga; Novikova, Tatiana; Kohli, Pushmeet; Lempitsky, Victor (2013). "Image Binarization for End-to-End Text Understanding in Natural Images". *2013 12th International Conference on Document Analysis and Recognition* (https://www.microsoft.com/en-us/research/wp-content/uploads/2016/11/mbnlk_icdar2013.pdf) (PDF). pp. 128–132. doi:10.1109/ICDAR.2013.33 (https://doi.org/10.1109%2FICDAR.2013.33). ISBN 978-0-7695-4999-6. S2CID 8947361 (https://api.semanticscholar.org/CorpusID:8947361). Archived (https://web.archive.org/web/20171113184347/https://www.microsoft.com/en-us/research/wp-content/uploads/2016/11/mbnlk_icdar2013.pdf) (PDF) from the original on November 13, 2017. Retrieved May 2, 2015.
20. Pati, P.B.; Ramakrishnan, A.G. (May 29, 1987). "Word Level Multi-script Identification". *Pattern Recognition Letters*. **29** (9): 1218–1229. Bibcode:2008PaReL..29.1218P (https://ui.adsabs.harvard.edu/abs/2008PaReL..29.1218P). doi:10.1016/j.patrec.2008.01.027 (https://doi.org/10.1016%2Fj.patrec.2008.01.027).
21. "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.
22. Smith, Ray (2007). "An Overview of the Tesseract OCR Engine" (https://web.archive.org/web/20100928052954/http://tesseract-ocr.googlecode.com/svn/trunk/doc/tesseracticdar2007.pdf) (PDF). Archived from the original (http://tesseract-ocr.googlecode.com/svn/trunk/doc/tesseracticdar2007.pdf) (PDF) on September 28, 2010. Retrieved May 23, 2013.
23. "OCR Introduction" (http://www.dataid.com/aboutocr.htm). Dataid.com. Retrieved June 16, 2013.
24. "How OCR Software Works" (https://web.archive.org/web/20090816210246/http://ocrwizard.com/ocr-software/how-ocr-software-works.html). OCRWizard. Archived from the original (http://ocrwizard.com/ocr-software/how-ocr-software-works.html) on August 16, 2009. Retrieved June 16, 2013.
25. "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.
26. Assefi, Mehdi (December 2016). "OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym" (https://www.researchgate.net/publication/310645810). *ResearchGate*.
27. "How the Best OCR Technology Captures 99.91% of Data" (https://www.bisok.com/grooper-data-capture-method-features/multi-pass-ocr/). *www.bisok.com*. Retrieved May 27, 2021.
28. Woodford, Chris (January 30, 2012). "How does OCR document scanning work?" (http://www.explainthatstuff.com/how-ocr-works.html). Explain that Stuff. Retrieved June 16, 2013.

29. "How to optimize results from the OCR API when extracting text from an image? - Haven OnDemand Developer Community" (https://web.archive.org/web/20160322103356/https://community.havenondemand.com/t5/Wiki/How-to-optimize-results-from-the-OCR-API-when-extracting-text/ta-p/1656). Archived from the original (https://community.havenondemand.com/t5/Wiki/How-to-optimize-results-from-the-OCR-API-when-extracting-text/ta-p/1656) on March 22, 2016.
30. Fehr, Tiff (March 26, 2019). "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). *The New York Times*. ISSN 0362-4331 (https://search.worldcat.org/issn/0362-4331). Retrieved June 16, 2023.
31. "Train Your Tesseract" (http://trainyourtesseract.com/). *Train Your Tesseract*. September 20, 2018. Retrieved September 20, 2018.
32. "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.
33. 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). S2CID 11873638 (https://api.semanticscholar.org/CorpusID:11873638).
34. "Google Books Ngram Viewer" (https://books.google.com/ngrams/info). *books.google.com*. Retrieved July 20, 2023. "When we generated the original Ngram Viewer corpora in 2009, our OCR wasn't as good […]. This was especially obvious in pre-19th century English, where the elongated medial-s (ſ) was often interpreted as an f, […]. Here's evidence of the improvements we've made since then, using the corpus operator to compare the 2009, 2012 and 2019 versions […]"
35. "Code and Data to evaluate OCR accuracy, originally from UNLV/ISRI" (https://code.google.com/p/isri-ocr-evaluation-tools/). Google Code Archive.
36. 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.
37. 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.
38. Mohseni, Maedeh Haji Agha; Azmi, Reza; Layeghi, Kamran; Maleki, Sajad (2019). *Comparison of Synthesized and Natural Datasets in Neural Network Based Handwriting Solutions* (https://civilica.com/doc/924198/certificate/pdf/). ITCT – via Civilica.
39. Kapidakis, Sarantos; Mazurek, Cezary and Werla, Marcin (2015). *Research and Advanced Technology for Digital Libraries* (https://books.google.com/books?id=kEyGCgAAQBAJ&q=OCR+and+long+s). Springer. p. 257. ISBN 9783319245928.
40. Atkinson, Kristine H. (2015). "Reinventing nonpatent literature for pharmaceutical patenting". *Pharmaceutical Patent Analyst*. **4** (5): 371–375. doi:10.4155/ppa.15.21 (https://doi.org/10.4155%2Fppa.15.21). PMID 26389649 (https://pubmed.ncbi.nlm.nih.gov/26389649).
41. "scanno" (https://www.hoopoes.com/jargon/entry/scanno.shtml). *Hoopoes*. May 2001.

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)
