OCR of handwritten transcriptions of Ancient Egyptian hieroglyphic text*

Mark-Jan Nederhof
University of St Andrews

Abstract

Encoding hieroglyphic texts is time-consuming. If a text already exists as hand-written transcription, there is an alternative, namely OCR. Off-the-shelf OCR systems seem difficult to adapt to the peculiarities of Ancient Egyptian. Presented is a proof-of-concept tool that was designed to digitize texts of Urkunden IV in the hand-writing of Kurt Sethe. It automatically recognizes signs and produces a normalized encoding, suitable for storage in a database, or for printing on a screen or on paper, requiring little manual correction.

The encoding of hieroglyphic text is RES (Revised Encoding Scheme) rather than (common dialects of) MdC (Manuel de Codage). Earlier papers argued against MdC and in favour of RES for corpus development. Arguments in favour of RES include longevity of the encoding, as its semantics are font-independent. The present study provides evidence that RES is also much preferable to MdC in the context of OCR. With a well-understood parsing technique, relative positioning of scanned signs can be straightforwardly mapped to suitable primitives of the encoding.

1 Introduction

Our aim was to digitize the first few volumes of *Urkunden IV*, in the handwriting of Sethe (1927); cf. Figure 1. These contain transcriptions of texts from the 18th Dynasty, mostly historical records. Rather than encoding these by hand, we wanted a more cost-effective solution, by automating as much as possible.

There are several reasons why we have chosen for the Urkunden IV for our study. First, the copyright has expired. Although we would wish for more sharing of resources, we are bound by the realities of Egyptology. Secondly, texts from the Urkunden IV are frequently used in teaching. Availability of the encodings could be useful for setting translation exercises. Lastly, the texts are relatively straightforward, and most signs can be disambiguated and encoded easily.

Optical character recognition (OCR) is the task of automatically converting an image into a text encoding. Such an image could be the scan of a (typeset) book or of a handwritten document. By *text encoding* we mean a representation that explicitly identifies signs. Signs can be referred to by their codepoints in e.g. Unicode. In the case of Ancient Egyptian hieroglyphic, signs are commonly referred to by their so-called Gardiner names (Gardiner, 1957).

A related task is *handwritten text recognition* (HTR), which tends to place an emphasis on cursive handwriting, where letters are joined together, produced using flowing hand movements. Whereas in the present study we are interested in handwritten hieroglyphic, most signs are physically separated by whitespace, and thereby OCR appears to be more relevant to us than HTR. However, the variation that is found between different occurrences of hieroglyphic signs is considerable, which sets our study apart from the most common applications of OCR.

Another complication is the writing system. The most common hieroglyphs are available in Unicode, which as of version 5.2 contains 1071 hieroglyphic signs. The large number of signs makes training of

* Extended version of a presentation at **Altertumswissenschaften in a Digital Age: Egyptology, Papyrology and Beyond** (DHEgypt15), Leipzig, November 5, 2015.

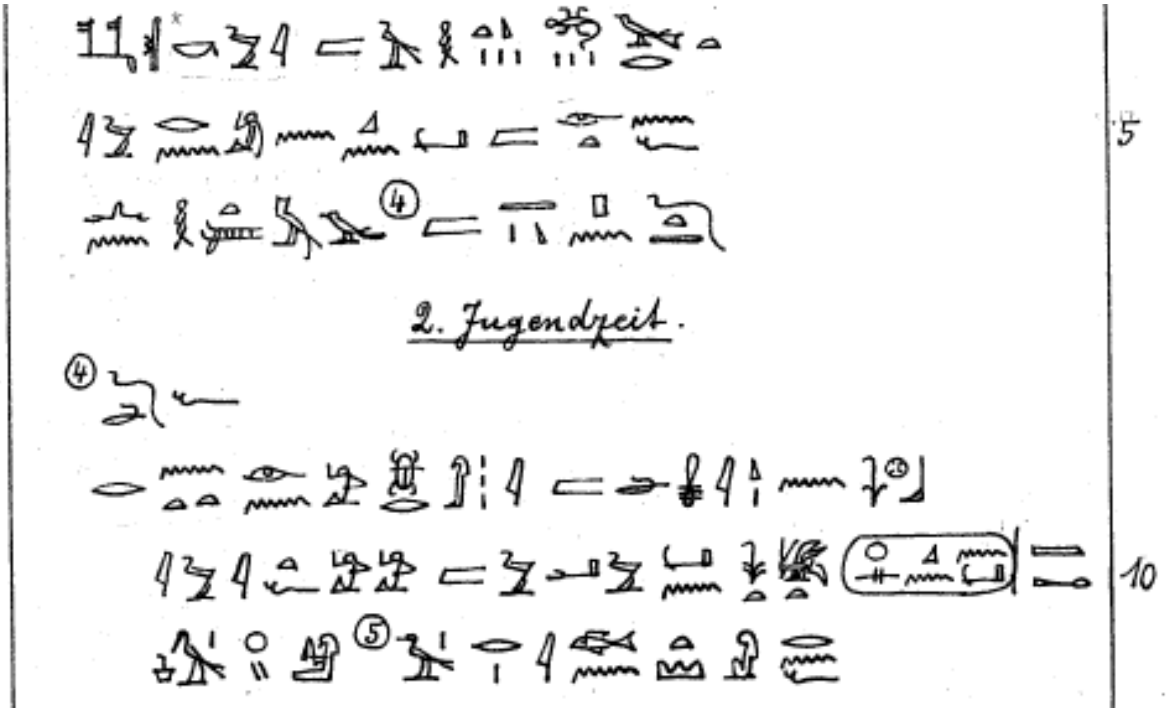

Figure 1: Example from the first volume of Urkunden IV.

OCR systems challenging. For this reason we have avoided neural networks and related concepts that require large quantities of training material even for relatively few signs.

The hieroglyphic writing system allows signs to be arranged next to one another, horizontally, vertically, or in various other ways. Existing OCR systems do not normally have mechanisms for dealing with these kinds of formatting, and therefore they seem ill-equipped to deal with hieroglyphic text.

Sethe used *hatching* (also referred to as *shading*) to indicate damaged and illegible text, by drawing diagonal lines through groups of signs, effectively connecting together the individual signs within groups, which causes severe complications for identifying the signs. Other complications include circled line numbers and footnote markers.

2 Related work

Photographs from the Pyramid of Unas were the subject of a related project with the aim to automatically recognize hieroglyphs (Franken and Gemert, 2013). Various techniques of computer vision were used, and trained on a manually created collection of almost 4000 hieroglyphs. Accuracies of around 85% were reported.

In terms of the range of image processing techniques, our project has more modest aims. Identifying potential signs is relatively easy, as signs are invariably drawn in black on a white background, whereas the photographs from the Pyramid of Unas involve signs and backgrounds of different colours. In other respects, our project goes a step further, by also processing the relative positioning of signs, to produce a complete text encoding.

An HTR project that has drawn considerable attention recently is Transkribus, which places an emphasis on historical documents.[1] It was designed for alphabetical writing systems however, and as far as we have been able to determine, hieroglyphic is far outside its present capabilities.

3 Motivations

A wealth of hieroglyphic text is available in printed form, whereas very few resources exist in machine-readable format. Were we able to turn the printed documents into text encodings, this would offer

[1] https://transkribus.eu

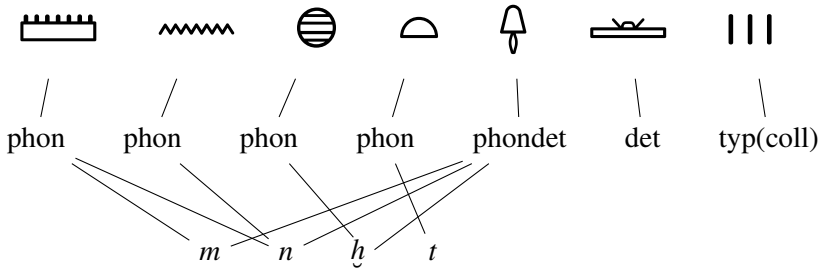

Figure 2: Annotation of a words with sign use.

considerable advantages. We will discuss three applications.

3.1 Corpus linguistics

Corpus linguistics is the study of language on the basis of actual language use, supported by annotated or unannotated texts. It may focus on different aspects of language, such as lexicography (cf. Thesaurus Linguae Aegptiae[2]) or grammar (cf. Ramsés[3]). A recent use of corpus linguistics in Egyptology is a systematic study of the use of signs to form words, by annotating words with sign functions (Nederhof and Rahman, 2015). This is illustrated in Figure 2, with a sequence of signs at the top, a sequence of letters from the transliteration at the bottom, and between those the sign functions, which may connect signs and letters together.

3.2 Non-linguistic studies

Texts annotated with translations can also be used by, e.g., historians who are not able to read hieroglyphic. In the past, printed volumes of translations such as those by Breasted (1906), Sethe (1914), Helck (1984) and Cumming (1982) have been widely used, but having the translations in digital form, linked to the original hieroglyphic would offer considerable added value. In particular, the texts would then be searchable and contentious translations could be easily verified.

3.3 Learning and teaching

The St Andrews corpus contains several dozens of classical texts.[4] As far as we are aware, it is the only corpus of Middle Egyptian of which all source data can be freely downloaded. An important objective of the corpus is to help those trying to learn Ancient Egyptian through self-study. It should be appreciated that self-study is the only option available to those interested in Egyptology who live far away from the very few centers of study today that offer relevant courses in the traditional classroom setting.

The corpus currently contains the first 32 texts from the Urkunden IV. The hieroglyphic transcriptions were obtained by manually correcting the output of our OCR tool. To this we added transliterations and translations. The only complete translation of the first volumes of the Urkunden IV that is available today is in German (Sethe, 1914). There are excellent translations of some of the same texts in English by Breasted (1906), produced at the beginning of the 20th century, but these could naturally not rely on all the lexicographical knowledge that we possess today. A modern English translation would thereby fill an important gap.

4 The tool

The input to the OCR tool is a directory containing a number of image files, each of which is a scan of a page from a hieroglyphic text. In order to keep the design simple, we assume all images are binary, i.e., they consist exclusively of black and white pixels. In the same directory, the tool creates an XML file, in

[2] http://aaew.bbaw.de/tla/
[3] http://ramses.ulg.ac.be
[4] https://mjn.host.cs.st-andrews.ac.uk/egyptian/texts/

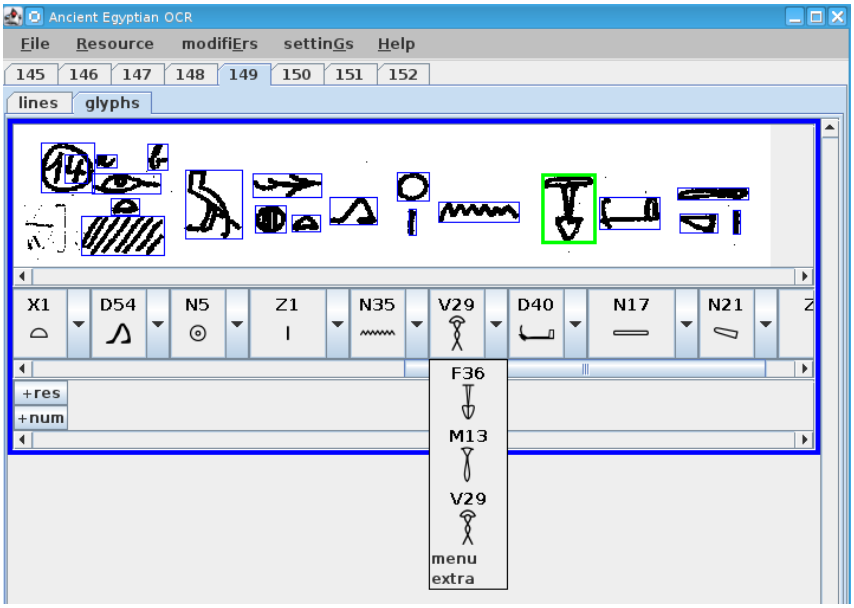

Figure 3: Manual correction of automatic classification of signs, via drop-down menus.

which partial results of the OCR process are stored. This allows the processing of a text to be spread out over a number of sessions.

From the user's perspective, this processing consists of four stages:

1. identifying lines,

2. detecting and classifying signs within each line,

3. formatting the signs,

4. exporting the formatted hieroglyphic.

Identifying lines can be done automatically, possibly followed by manual correction, or be done completely manually, by drawing a polygon around each line of hieroglyphic text, by a sequence of mouse clicks. For identifying lines automatically, various heuristics are applied to distinguish hieroglyphic text from other text, such as lines of German.

After lines have been identified, the surface within the relevant polygon is analyzed for neighbouring pixels that are all black. A connected set of black pixels is here called a *blob*. In many cases a blob is a sign, which can be classified (i.e. its Gardiner name can be guessed) by the procedures described in Section 4.1. Sometimes however, several blobs make one sign, or several signs are connected into one blob, as discussed in Section 4.2.

Formatting classified signs into a hieroglyphic encoding will be discussed in Section 4.3. After all lines in a text have been formatted, the encoding of the entire text can be exported as a single file. For our application, this file may subsequently be combined with files containing transliteration and translation to produce one interlinear representation (Nederhof, 2009).

At each stage, graphical user interfaces allow the user to make corrections. For example, if the black pixels within a line are incorrectly partitioned into blobs, the user may split up one blob, join several blobs, remove black pixels from consideration by an 'eraser' tool, etc. The classification procedure provides up to five suggestions of candidate signs. If the top choice is not correct, one of the others may be selected, or an entirely different one may be chosen from a menu including all signs; see Figure 3 for an example. Also the automatically formatted hieroglyphic may be manually corrected by a graphical user interface; see Figure 4 for an example.

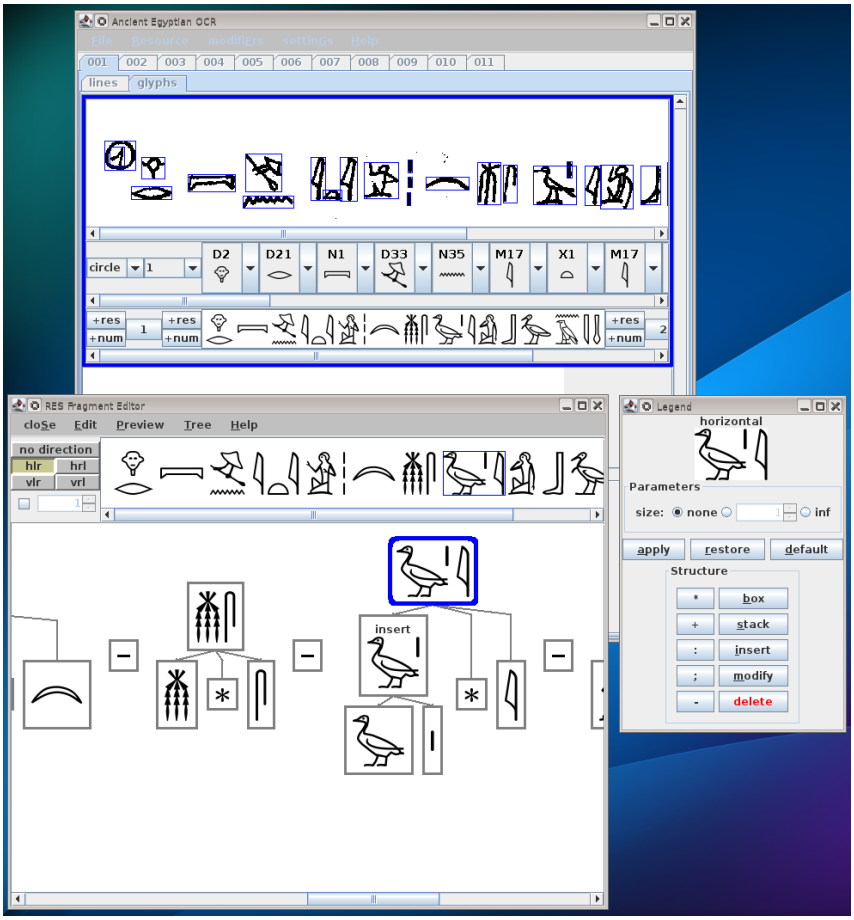

Figure 4: The graphical editor of formatted hieroglyphic.

4.1 Classification

Our approach to classifying signs centers around the concept of *prototype*, which is a classified example of a sign. In our tool, prototypes are stored simply as images in a directory. Figure 5 provides an example. Initially, the database is empty and all signs must be classified manually. A selection of the classified signs are then transferred to the database to act as prototypes. New signs are subsequently compared to each of the prototypes, and the name of the prototype that is 'closest' to a new sign is then proposed as the name of the new sign. The user may correct this manually. After annotating more text, more signs may be added to the database of prototypes, etc. In this way, the database grows steadily, and at the same time, accuracy improves and fewer and fewer manual corrections are needed.

Because a sign may be drawn in different ways, it is beneficial to accuracy to keep more than one prototype per sign. At the same time, the overall number of prototypes should be kept small, as each prototypes incurs additional computational costs. The decision which signs from processed texts to keep as prototypes is semi-automatic. That is, automatic procedures are applied to prune the database and discard prototypes that are too similar to one another, while the user can manually intervene and remove atypical prototypes, through a graphical user interface.

Our working assumption is that separate databases of prototypes are maintained for different handwritings. For the Urkunden IV, one may have one such database for Sethe's handwriting, and one for the handwriting of Helck (1955), who compiled later volumes of the Urkunden IV. Future versions of our software may employ several databases of prototypes when processing a new or unidentified handwriting.

In order to decide which prototype is 'closest' to a new sign, we apply a very simple algorithm called the Image Distortion Model (IDM), following Keysers et al. (2007). This compares two images of the same size, and computes a score for their dissimilarity. Two identical images result in score 0, and the

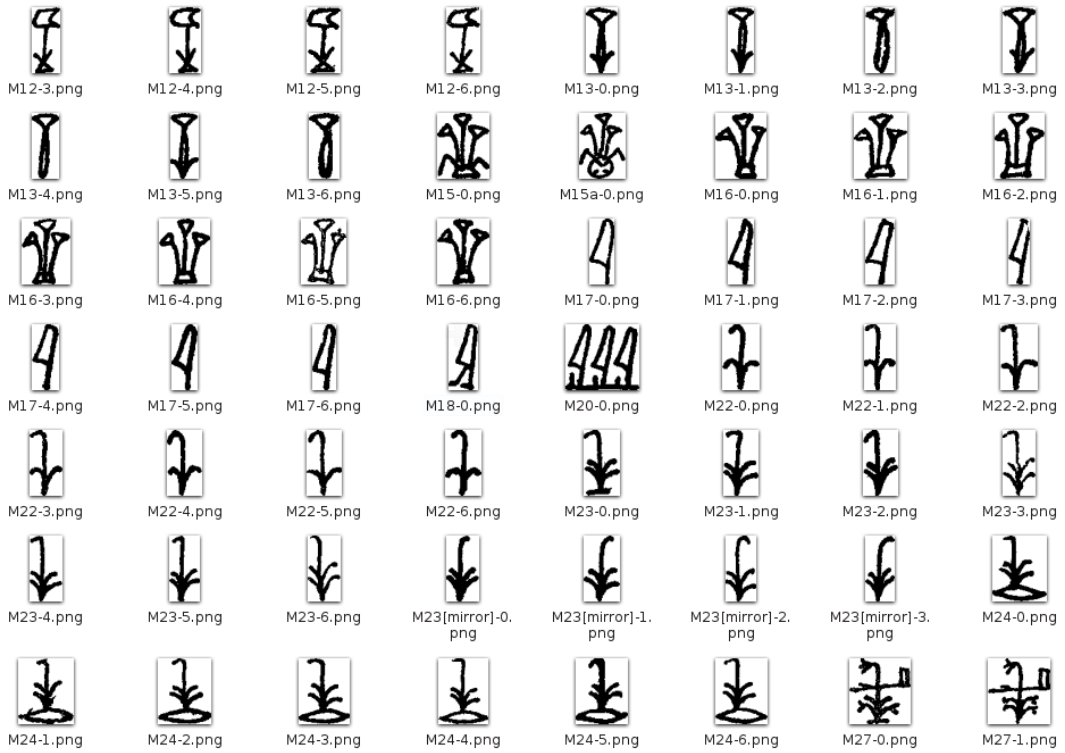

Figure 5: Examples of prototypes in our database, as shown by a standard file browser.

more different two images are the higher the score will be. In our tool, the images are rescaled to squares of uniform size before the IDM is applied.

Because of the high computational costs of the IDM, it is not feasible to compare a new sign to all prototypes. Therefore we initially apply the Fast Fourier Transform (FFT) to a new image, to decompose it into a sequence of amplitudes for different frequencies. This sequence is combined with sequences computed before for the prototypes. This is done by computing a weighted average of the differences of the amplitudes, between a new sign and a prototype. This average can be called a 'cost', just as in the case of the IDM. The 30 prototypes are determined that result in the lowest FFT costs with respect to the new sign. In the next phase, only these prototypes are compared to the new sign using the IDM.

A number of signs are very difficult to distinguish based on their shapes alone, especially after rescaling to squares of uniform size. This means that following application of the FFT and the IDM, there may be several prototypes that have very similar low IDM scores, and the small differences between these scores may not have much bearing on the question which one is correct. For this reason, we rerank these prototypes, based on three features for each sign, obtained by investigating sign occurrences in the corpus of texts processed before, namely:

1. relative frequency,

2. distribution of sizes, relative to the unit size (to be explained below),

3. distribution of ratios of width and height.

The first feature gives us an estimated probability for each sign. The second and third are captured by log-normal distributions, which give us conditional probability densities for a size and for a width/height ratio, given a sign. By multiplying these three numbers, assuming size and width/height ratio to be independent, we obtain a single number. The higher this number, the more likely a sign is. This determines a new ranking for the most promising prototypes. After this reranking, the top few signs are kept, and offered to the user for checking.

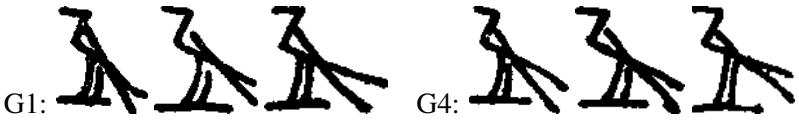

Figure 6: Three occurrences of the vulture or ‘aleph bird’ (left) and three occurrences of the buzzard or ‘tjw bird’ (right), which are indistinguishable in Sethe’s handwriting.

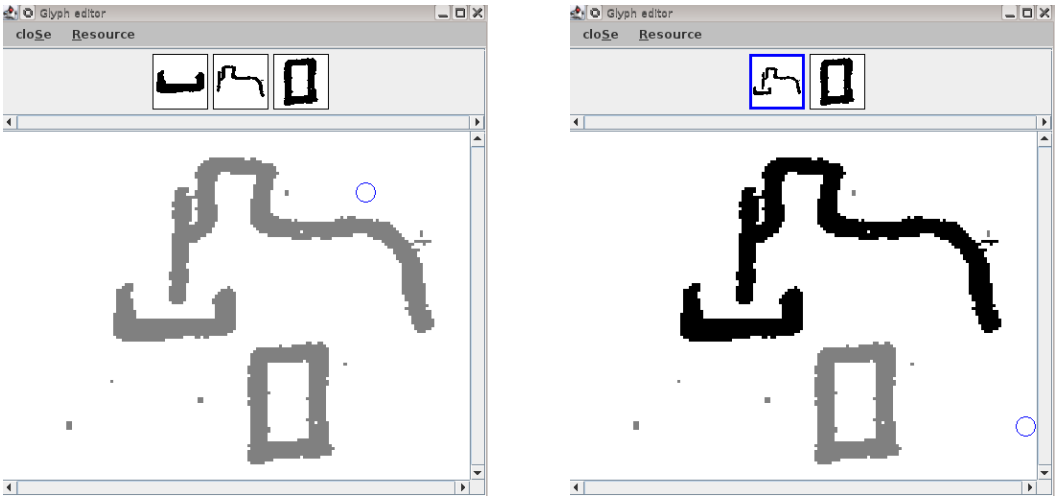

Figure 7: The first two blobs from the screenshot on the left can be selected and joined, to give the result on the right, which contains the two signs of the word *stp*.

The *unit size* is the height of an unscaled occurrence of sign A1 (’sitting man’), which normally corresponds to the height of a line. This sign can be scaled down if it needs to fit above or below another sign. There are other signs that are normally much smaller than unit size. For example, the typical height of the ‘sun’ sign is about half the unit size. The unit size can be estimated by looking at median sizes of blobs.

For the purposes of the FFT and the IDM, our tool excludes a few signs that are very similar to other signs. The reason is that a new sign may appear to be more similar to one prototype rather than to another due to purely accidental variations, and this may lead to misclassification. This holds in particular for the signs G1 and G4 (cf. Figure 6), for Z1 (semogram marker) and Z15 (numeral), which both appear as short vertical strokes, although the first tends to be somewhat shorter than the second, and for various kinds of circles, such as the signs for ‘sun’, ‘threshing floor’, ‘grain (of sand)’, ‘ring’, the letter ‘o’, etc. For example, a new sign would be compared to prototypes for G1, but not those for G4, when applying the FFT and the IDM. If prototypes for G1 are found to be close to the new sign, then G4 is automatically added to the list of candidates. The subsequent reranking, by relative frequency, by size and by width/height ratio, may then place G1 ahead of G4 or vice versa.

4.2 Blobs versus signs

Some signs consist of several blobs. An example is given in Figure 7 for the word *stp*, written with two signs, the first of which consists of two isolated blobs. Assuming the database does not yet have any examples of this sign, then these two blobs would in the first instance be proposed as separate potential signs. The user may intervene and first select a rectangle from a line. A graphical tool is then opened that allows the user to manipulate potential signs that all fall within that rectangle. One such manipulation is to join two blobs into one.

For prototypes in the database that consist of several blobs, copies of these blobs are stored separately in the database as well, and new, unclassified blobs are matched to these using the FFT and the IDM. If two neighbouring blobs are then both classified as, for example, ‘part of U21’, then the IDM is applied

Figure 8: A scarab drawn with one of its legs detached, which is analyzed as a separate blob.

Figure 9: Damaged text in the original inscription is represented by Sethe using hatching.

on the combination of the two blobs and prototypes of U21 in the database. If this results in a much better IDM score, then the two blobs detected as 'part of U21' are automatically replaced by a single combined sign classified as U21.

Frequently the scarab is drawn with one or more of its legs detached, as illustrated in Figure 8. This may seem innocuous, but poses a severe challenge to our framework. In many cases, the above procedure is not able to 're-attach' the legs after recognizing the parts.

The problems caused by hatching are illustrated in Figure 9. There are five hieroglyphic signs, two square brackets and a footnote marker 'a'. The diagonal lines link the left-most four hieroglyphic signs together, to make them appear as a single blob. Unconnected diagonal lines are blobs by themselves.

We have experimented with algorithms to identify diagonal lines that potentially represent parts of hatching, and to automatically remove them from blobs. This is illustrated for the right-most sign in Figure 9, where only the black pixels remain after application. As can be seen, the result is less than perfect. Nonetheless it can be faster than applying the manual eraser tool.

4.3 Formatting

In hieroglyphic text, signs are laid out to fill the available surface in an aesthetically pleasing manner, which we will refer to here simply as *formatting*. For example, two flat signs may be on top of one

another, and two high but narrow signs may be next to one another. However, one may find many more arrangements of signs, e.g. where one sign occupies an open corner within the bounding box of another.

The most widely used encoding of hieroglyphic text is called the Manuel de Codage (MdC). A case was made by Nederhof (2013) that the MdC ought to be avoided for corpus development. The originally published version of the MdC could express only horizontal and vertical arrangements of signs, which amounts to a small fraction of the formatting one finds in original inscriptions. For this reason, dialects of the MdC have added a wide variety of new constructions, so as to be able to represent the 'special groups'. These new additions have invariably been impractical, ill-defined, implementation-specific and font-specific. The latter issue, that of the encoding losing its correctness when the font is replaced, is particularly damaging to the prospect of building a corpus that should still be used in ten, twenty or fifty years.

The Revised Encoding Scheme (RES) was proposed by Nederhof (2002) to remedy this situation. It offers a small number of simple primitives, which together are powerful enough to format almost all hieroglyphic text in a satisfactory manner. If one font is replaced by another, so that signs take slightly different shapes or sizes, then the formatting adapts automatically.[5]

In the process of developing our OCR tool, we found that RES is particularly convenient for translating scanned images to normalized encodings. This is because the primitives of RES correspond to relative positionings of groups that one can automatically recognize from an image. For example, one may recognize that one group is above another, which would be translated to the ':' operator of RES, and one may recognize that one group is roughly within the bounding box of another, which would be translated to the 'insert' operation of RES.

On a more technical level, the formatting of the encoding is obtained by the concept of *2D parsing*; see Álvaro et al. (2014) for a related application. The intuition is that parts of an image are combined into larger parts until the entire image is obtained. One may also see it from the opposite side, that of starting from the complete image and dividing it into parts until one reaches the atomic level, which in our case are the individual signs.

We apply a weighted version of 2D parsing, whereby a combination of two groups of signs into one larger group using a certain operation has a certain cost. The more natural this combination is, the lower the cost. For example, two groups can be combined in a vertical arrangement if the first group is strictly above the second, and the center of the first group has roughly the same x coordinate as the center of the second group. The less this is true, e.g. the greater the difference between the two mentioned x coordinates, the higher the cost will be. The tool exhaustively considers all parses of an image, and the one with the lowest total cost is taken.

A first example is presented in Figure 10. Here the classification of signs is done with 100% accuracy. The formatting is not perfect, but it should be appreciated that this is the result of fully automatic processing, before any manual correction. Please note that the positions of signs in the normalized encoding are not simply the physical coordinates of corresponding signs in the image, but rather, groupings of sign are recognized, such as Q3 next to X1, which as a group occur above N1, etc.

Two other examples are presented in Figures 11 and 12. These illustrate occurrences of the 'insert' operation. In the first example it is automatically obtained by the determination that sign N5 is roughly inside the bounding box of sign G39, in the upper-right ('te' = 'top-end') corner.

5 Accuracy

The accuracy of the classification procedure explained in Section 4.1 was measured in isolation, assuming correct partition of the surface into signs. That is, we assume that 'multi-blob' signs have already been joined and that 'multi-sign' blobs (e.g. due to hatching) have already been separated.

The 2252 prototypes come from the first 25 texts of Urkunden IV, by a combination of automatic selection and manual removal of atypical examples. The prototypes together represent 490 different signs, including some that are not hieroglyphs, such as footnote markers and (parts of) line numbers.

[5] See `https://mjn.host.cs.st-andrews.ac.uk/egyptian/res/js/edit.html` for an online hieroglyphic editor for RES encoding.

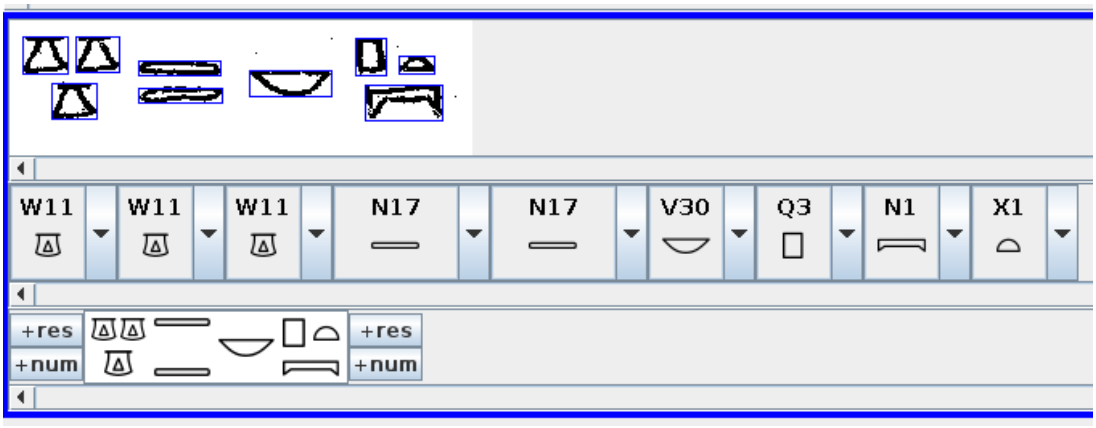

Figure 10: Fully automatic classification and formatting.

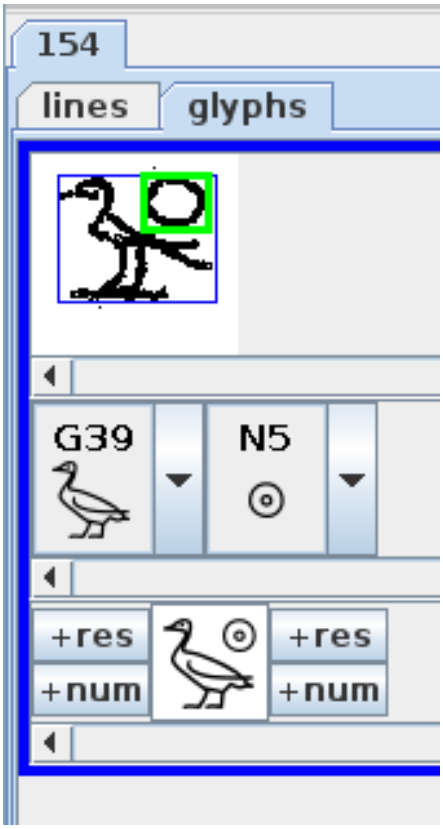

Figure 11: `insert[te](G39,N5)`.

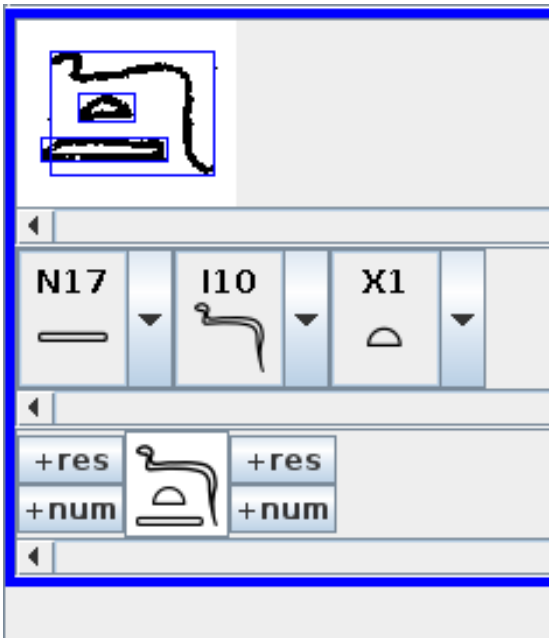

Figure 12: `insert[bs](I10,X1:N17)`.

The accuracy was measured on texts 26 – 32. Of the 2533 sign occurrences in these seven texts, 91.3% are classified correctly. In 95.5% of the cases, the correct name is among the top two candidates; the importance of this lies in the observation that minimal effort is needed in these cases to correct the result, by choosing the next option in the drop-down menu.

Figure 13 lists the most frequent errors. It is not surprising that the most frequently confused signs are semogram marker Z1 and numeral Z15, which we discussed before. Whereas N5 (‘sun’), N33 (‘grain’), O50 (‘threshing floor’) and the digit ‘0’ look quite distinct in a computer font, they all look like circles of slightly different sizes in Sethe’s handwriting. The distinction between the ‘vertical throw stick’ T14 and ‘slanted throw stick’ T15 is somewhat artificial, as they are graphical variants of one another. As remarked before, there is no visible difference between G1 and G4.

correct	top candidate	proportion
Z15:	Z1:	11.3%
N5: 	N33:	4.5%
N33:	0 (digit)	4.1%
T14: 	T15: 	4.1%
Z1:	Z15:	3.6%
G4: 	G1: 	3.2%
G1: 	G5: 	2.7%

correct	top candidate	proportion
G7: 	G26: 	2.3%
N37: 	O39: 	2.3%
O50: 	N5: 	1.8%
G17: 	G5: 	1.4%
G7: 	G17: 	1.4%
N16: 	N17:	1.4%
O50: 	N33:	1.4%

Figure 13: The most frequent classification errors in OCR of texts 26 – 32 of Urkunden IV, and their proportion of the total number of errors.

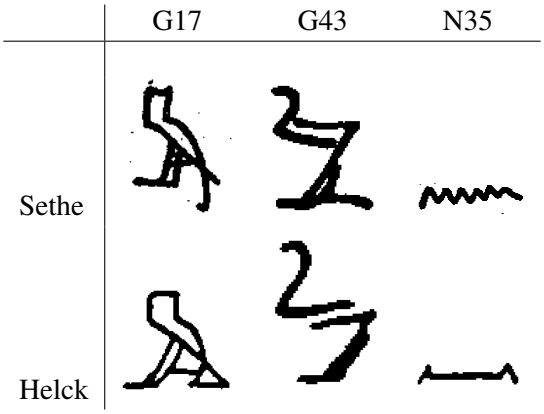

Figure 14: The two handwritings of Urkunden IV.

In Sethe's handwriting, the main noticeable difference between G1 and G5 is in the shape of the head. Misclassification may result if the body of some occurrence of G1 happens to be particularly close to that of a prototype of G5, despite the differences in the shapes of the head. This is one of the reasons why accuracy generally improves if we have more prototypes per sign. The errors involving G7 are caused by a complete lack of useable prototypes of this sign coming from texts 1 – 25.

Our tool ignores small blobs consisting of only a few black pixels, because these are typically caused by dust or other artefacts on the scanned pages. An adverse consequence of this is that the three grains of sand in N16 ('land with grains') are ignored. We have not manually corrected this by re-attaching the three dots to the rest of the sign. Consequently, occurrences of N16 are normally classified as N17. Because N16 and N17 can be seen as graphical variants, this is yet another artificial distinction.

We have also investigated whether the prototypes from Sethe could be used to recognize the handwriting of Helck, in later volumes of the Urkunden IV. The superficial comparison in Figure 14 is enough to conclude that the answer is negative. Even very common signs are drawn very differently. In the case of the common sign G43, there are two blobs in the case of Helck and only one in the case of Sethe, which poses an insurmountable obstacle for our framework.

6 Outlook

The source code of the OCR tool is freely available, including the database of prototypes, as part of PhilologEg.[6] It can be straightforwardly run from the command line on Linux and Mac OS X, and with some skills on Windows. We intend to make deployment more straightforward in the near future.

A weakness of the current design is that the classification of signs does not take context into account. We conjecture that a language model, such as that of Nederhof and Rahman (2015), could be used to improve accuracy.

Whereas the tool was designed for handwritten hieroglyphic transcriptions, it also works very well on typeset hieroglyphic, as we found in preliminary experiments with the IFAO font. This requires further investigation.

We also hope to extend investigations in the direction of hieratic. A major challenge is that processing of blobs then becomes much harder than in the case of hieroglyphic, due to signs commonly touching each other.

References

[Álvaro et al.2014] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí. 2014. Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden Markov models. *Pattern Recognition Letters*, 35:58–67.

[Breasted1906] J.H. Breasted. 1906. *Ancient Records of Egypt – Volume II*. The University of Chicago Press.

[Cumming1982] B. Cumming. 1982. *Egyptian Historical Records of the Later Eighteenth Dynasty*. Aris and Phillips.

[Franken and Gemert2013] M. Franken and J.C. Gemert. 2013. Automatic Egyptian hieroglyph recognition by retrieving images as texts. In *ACM International Conference on Multimedia*.

[Gardiner1957] A. Gardiner. 1957. *Egyptian Grammar*. Griffith Institute, Ashmolean Museum, Oxford.

[Helck1955] W. Helck. 1955. *Urkunden der 18. Dynastie, Fascicle 17*. Akademie-Verlag.

[Helck1984] W. Helck. 1984. *Urkunden der 18. Dynastie — Übersetzung zu den Heften 17–22*. Akademie-Verlag, Berlin.

[Keysers et al.2007] D. Keysers, T. Deselaers, C. Gollan, and H. Ney. 2007. Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1422–1435, August.

[Nederhof and Rahman2015] M.-J. Nederhof and F. Rahman. 2015. A probabilistic model of Ancient Egyptian writing. In *Proceedings of the 12th International Conference on Finite-State Methods and Natural Language Processing*, Düsseldorf, June.

[Nederhof2002] M.-J. Nederhof. 2002. A revised encoding scheme for hieroglyphic. In *Proceedings of the 14th Table Ronde Informatique et Égyptologie*, July. On CD-ROM.

[Nederhof2009] M.-J. Nederhof. 2009. Automatic creation of interlinear text for philological purposes. *Traitement Automatique des Langues*, 50(2):237–255.

[Nederhof2013] M.-J. Nederhof. 2013. The Manuel de Codage encoding of hieroglyphs impedes development of corpora. In S. Polis and J. Winand, editors, *Texts, Languages & Information Technology in Egyptology*, pages 103–110. Presses Universitaires de Liège.

[Sethe1914] K. Sethe. 1914. *Urkunden der 18. Dynastie – übersetzt. Volume I*. Hinrichs, Leipzig.

[Sethe1927] K. Sethe. 1927. *Urkunden der 18. Dynastie, Volume I*. Hinrichs, Leipzig.

[6] `https://mjn.host.cs.st-andrews.ac.uk/egyptian/align/`