System odczytywania hieroglifów

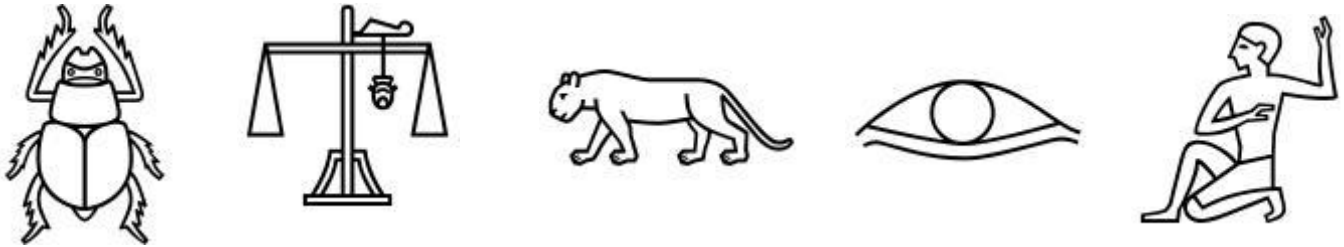
Albert Mouhoubi, Dawid Płaneta, Jakub Pietrzko

Abstrakt

Celem naszego projektu jest implementacja systemu do odczytywania **egipskich hieroglifów** z obrazków. W tym celu wygenerowano bazę wszystkich hieroglifów znajdujących się w tablicy **UTF8**. Z pomocą wikipedii pozwoliło to na późniejsze dopasowanie tych obrazków do opisów i znaczenia. W celu odszukaniu hieroglifów na obrazku użyto funkcji **match_template** z biblioteki **scikit-image**. Po znalezieniu kodów Unicode, program odczytuje znaczenie znaku z bazy i wyświetla ich znaczenia na wykresie wynikowym.

Przygotowanie bazy danych

Nasz projekt rozpoczęliśmy od przygotowania bazy danych, która zawiera **1071 hieroglifów** w postaci plików png gdzie nazwa pliku jest unicodem danego hieroglifu.



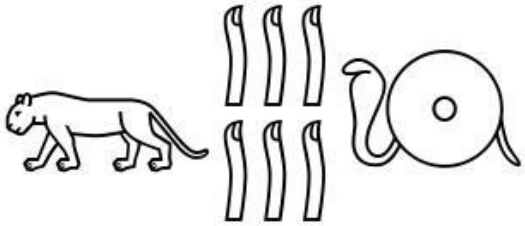
Żadna publiczna baza danych z hieroglifami, która nadawałby się do naszego projektu nie jest na tę chwilę dostępna, dlatego postanowiliśmy wygenerować własną w następujący sposób:

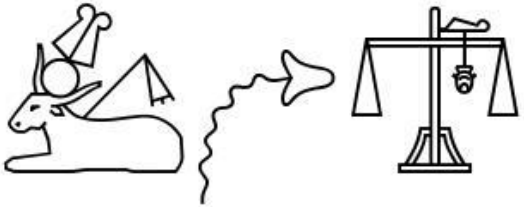
1. Jako obrazy hieroglifów wykorzystaliśmy dostępną czcionkę **Noto Sans Egyptian Hieroglyphs** [link]
2. Czcionkę załadowano do programu **FontForg**e [link], który z pomocą skryptu pythonowego pozwolił na eksport znaków z czcionki do postaci plików graficznych png z nazwą jako numer unicode.
3. Pliki **png** należało przekonwertować do formatu **jpg** dla lepszej współpracy z **scikit-image.** Do tego wykorzystaliśmy pakiet **Pillow** w pythonie do obrabiania plików graficznych.
4. Tak zebrana baza zawiera wszystkie dostępne hieroglify, każdy osobno w dobrze przyciętym pliku graficznym.

Generowanie pliku testowego

Do testowania naszego rozwiązania potrzebowaliśmy przygotować osobne grafiki, które będą zawierać więcej niż jeden hieroglif na raz.

Do utworzenia takich plików testowych wykorzystaliśmy rysowanie napisów na wykresach **matplotlib**





Wyciąganie info z wikipedii

Samo rozpoznanie danego znaku niewiele daje nam informacji. W celu poznania dodatkowego kontekstu oraz znaczenia znalezionych znaków, pobraliśmy dane na podstronie wikipedii na temat hieroglifów [link]. Do wyciągnięcia danych ze strony wykorzystaliśmy pakiet **BeautifulSoup.**

Do tych informacji należą:

1. Mapowanie Gardinera kategorii hieroglifów na poszczególne litery
2. Opis hieroglifu
3. Transliteracja
4. Fonetyka
5. Dodatkowe informacje

	A9 U+1300B	man steadying basket on head	**work**, **toil** (*kꜣt*) **load** (verb or noun), **burden** (*ꜣdp*) **carry** or **haul** (*fꜣj*)	1a. ꜣtp, ꜣṯp 1b. -fꜣ, fꜣj 2. -kꜣt	1a. to load, to be laden, master of the load; equals Coptic language, ϣτπ; (minor use for *ꜣtp*, *ꜣṯp*); 1b. to carry, to bear; additional constructs for carrier, bearer, supporter, (and types thereof); for fꜣ **dnj**, the "bearer-of-the-basket", see: Greek Kanephoros; 2. for *kꜣt*, *kꜣwtj*

Szukanie hieroglifów w obrazie

W projekcie bardzo pomocna okazała się funkcja paczki skimage - **match_template**. Poniżej jej sygnatura:

skimage.feature.match_template(image, template, pad_input=False, mode='constant', constant_values=0)

Najważniejsze dla nas parametry to:

image: tablica 2-D lub 3-D przedstawiająca obrazek

template: tablica 2-D lub 3-D przestawiająca obrazek do znalezienia w image

output: tablica współczynników korelacji image i template dla każdej pozycji w obrazku

Szukanie hieroglifów w obrazie

Funkcja ta wykorzystuje szybką **znormalizowaną korelacje krzyżową**. Pozwala to na obliczenie korelacji między dwoma obrazami w każdym punkcie. Daję to nie tylko prawdopodobieństwo występowania jednego w drugim, ale też pozycję w której występuje - również probabilistycznie.

```
result = match_template(test_word, glyph.imageFiltered)

if (np.max(result) < 0.65):
    continue

ij = np.unravel_index(np.argmax(result), result.shape)
x, y = ij[::-1]

resultData = Result()
resultData.x = x
resultData.glyph = glyph
results.append(resultData)
```

Szukanie hieroglifów w obrazie

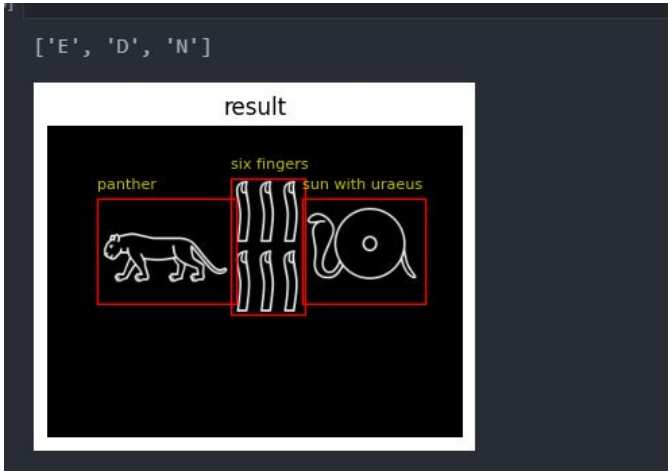
Przed uruchomieniem szukania, przeprowadzamy preprocessing obrazka, poprzez odwrócenie kolorów i przekonwertowanie go na skalę szarości by uprościć obliczenia i zwiększyć skuteczność wykrywania.

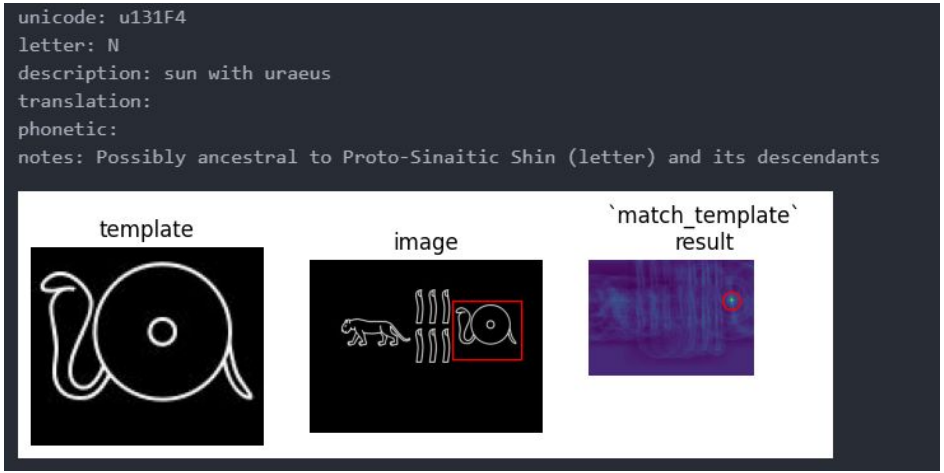
Przechodząc przez każdy plik w bazie i sprawdzając maksymalną korelacje z plikiem testowym, możemy znaleźć, które znaki występują w nim z największym prawdopodobieństwem. Dzięki informacji o korelacji w każdym punkcie obrazu testowego, znamy również najbardziej prawdopodobne pozycje tych znaków. Pozwala nam to na wyświetlenie prostokątów wokół nich na obrazie wynikowym.

```
glyph.image = imread('baza/'+glyph.unicode+'.jpg')
glyph.imageFiltered = rgb2gray(np.invert(glyph.image))
```

Wynik końcowy

Odnalezione hieroglify są zaznaczane na obrazie i wypisywane znane informacje





```
...  - - - - - - - - - - - - - - - - - - - - - -
     panther
     - unicode: u130EE
     - letter: E
     - transliteration:
     - phonetic: ʒby
     - notes:
     - - - - - - - - - - - - - - - - - - - - - -
     six fingers
     - unicode: u130B2
     - letter: D
     - transliteration:
     - phonetic:
     - notes:
     - - - - - - - - - - - - - - - - - - - - - -
     sun with uraeus
     - unicode: u131F4
     - letter: N
     - transliteration:
     - phonetic:
     - notes: Possibly ancestral to Proto-Sinaitic Shin (letter) and its descendants
     - - - - - - - - - - - - - - - - - - - - - -
```

Rozwój

Ulepszeniem projektu byłoby zamiast wykorzystywania algorytmów do szukania podobieństw zaimplementowanie **sieci konwolucyjnej** i wytrenowanie jej różnymi wariacjami (rotacja, rozmycie, uszkodzenia itp.) hieroglifów w celu osiągnięcia większej skuteczności, elastyczności i wydajności.

Baza w tym wypadku musiałaby się składać z grafik równych rozmiarów, co jest łatwe do osiągnięcia.

Bibliografia

- match_template - Dokumentacja
- List of Egyptian hieroglyphs
- J. P. Lewis, “Fast Normalized Cross-Correlation”, Industrial Light and Magic