Projekt zaliczeniowy - Rozpoznawanie hieroglifów wraz z tłumaczeniem na język polski

A. Szymczuk, K. Dzięgiel, M. Szpor

The Faculty of Physics, Mathematics and Computer Science
T. Kościuszko Cracow University of Technology
SYMULACJE KOMPUTEROWE

Plan

Cel projektu

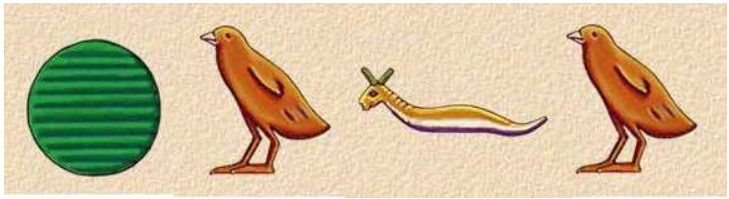
Celem projektu było napisanie programu rozpoznającego graficzne znaki – egipskie hieroglify oraz tłumaczącego znalezione słowa na język polski.

Hieroglify

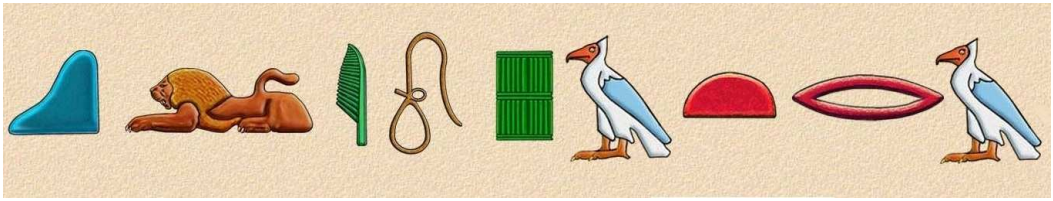
- Istnieją 3 rodzaje hieroglifów: znaki fonetyczne, znaki ideograficzne oraz determinatywy
- Założenie do projektu: znaki są pojedynczymi literami lub znakami dwuzgłoskowymi
- W projekcie przedstawiono 3 słowa, które oznaczają imiona egipskich władców:
  - Khufu - Cheops
  - Kliopatra - Kleopatra
  - Ptolimis - Ptolemeusz

Hieroglify

Obrazy wybrane do tłumaczenia:



Khufu



Kliopatra

Ptolmis

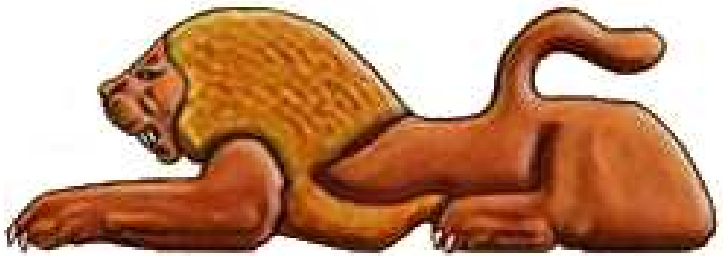
Biblioteki

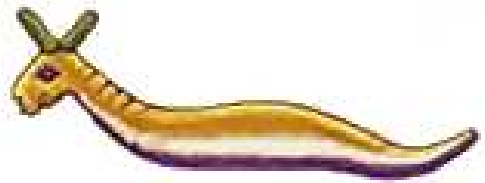
scikit-image
image processing in python

- Biblioteka Scikit-image, która pozwoliła na dokonywanie analizy obrazów:
  - Biblioteka ta jest zbiorem gotowych funkcji do przetwarzania obrazów
  - Jest ona bezpłatna i dostępna bez ograniczeń
- Biblioteka Numpy – do obliczeń numerycznych
- Biblioteka Matplotlib – do tworzenia wykresów

Wzorce

Pierwszym krokiem przed napisaniem kodu było wykonanie tzw. „wzorców” w celu wyszukania konkretnych liter na obrazie. Każdy wzorzec odpowiada za inną literę. Przykładowe wzorce:





Analiza kodu

1.Wczytywanie obrazów wraz z ich przekonwertowaniem na kolor czarnobiały i utworzeniem z nich negatywu:

```
image_khufu = rgb2gray(np.invert(imread('chufu.jpg')))
image_kliopatra = rgb2gray(np.invert(imread('kleopatra.jpg')))
image_ptolemeus = rgb2gray(np.invert(imread('ptolemeusz.jpg')))
```

Analiza kodu

2.Wczytywanie wzorow wraz z ich przekonwertowaniem na kolor czarnobiały i utworzeniem z nich negatywu:

```
wzor_u = rgb2gray(np.invert(imread('wzorzec_u.JPG')))
wzor_kh = rgb2gray(np.invert(imread('wzorzec_kh.JPG')))
wzor_f = rgb2gray(np.invert(imread('wzorzec_f.JPG')))
wzor_a = rgb2gray(np.invert(imread('wzorzec_a.JPG')))
wzor_i = rgb2gray(np.invert(imread('wzorzec_i.JPG')))
wzor_k = rgb2gray(np.invert(imread('wzorzec_k.JPG')))
wzor_l = rgb2gray(np.invert(imread('wzorzec_l.JPG')))
wzor_m = rgb2gray(np.invert(imread('wzorzec_m.JPG')))
wzor_o = rgb2gray(np.invert(imread('wzorzec_o.JPG')))
wzor_p = rgb2gray(np.invert(imread('wzorzec_p.JPG')))
wzor_r = rgb2gray(np.invert(imread('wzorzec_r.JPG')))
wzor_s = rgb2gray(np.invert(imread('wzorzec_s.JPG')))
wzor_t = rgb2gray(np.invert(imread('wzorzec_t.JPG')))
```

Analiza kodu - funkcja

3.Kluczowy element programu - funkcja szukająca wzoru na danym obrazie:

```
def wykrywacz(image_wzor,image_obraz,tolerancja,litera_egipska):
    #Tworzenie okienka do wyswietlania wzorca i obrazow na nim odnalezionych
    fig = plt.figure(figsize=(12, 5))
    #Podzial okienka  na dwa obszary
    ax1 = plt.subplot(1, 2, 1)
    ax2 = plt.subplot(1, 2, 2)


    result= match_template(image_obraz, image_wzor)
    #Pomocnicza zmienna w ktorej przechowywana bedzie kopia result, na ktorej bedziemy operowac
    result2=result
    #wysokosc i szerokosc wzoru
    hwzor, wwzor = image_wzor.shape
    #lista w ktorej  bedziemy zapisywac znalezione hieroglify
    literka=[]
    #listy, w ktorej zapisywac bedziemy wspolrzedne znalezionych hieroglifow
    x1=[]
    y1=[]
    #Indeks elementu listy result2, ktory posiada najwyzsza wartosc korelacji(czyli jest najbardziej prawdopodobne ze w tym miejscu znajduje sie hieroglif)
    ij=np.unravel_index(np.argmax(result2), result.shape)
    #pobranie skladowych indeksu i odwrocenie ich kolejnosci
    x, y= ij[::-1]
    #dodanie do list wspolrzednych indeksu najwyzszej wartosci result
    x1.append(x)
    y1.append(y)
    #do listy literka dodajemy wartosc wspolrzednej x (by pozniej posortowac odpowienio hieroglify), literke oraz informacyjnie wartosc w result
    literka.append([x , litera_egipska,np.max(result2)])
    #zerujemy najwieksza wartosc  w result by znalezc kolejna najwieksza wartosc w innymm miejscu
    result2[ij]=0
    #na wszelki wypadek gdyby najwyzsza wartosc okazala sie byc w poblizu starej wartosci, zerujemy rowniez elementy obok
    result2[(y,x-1)]=0
    result2[(y,x+1)]=0
```

Analiza kodu - funkcja

```
def wykrywacz(image_wzor,image_obraz,tolerancja,litera_egipska):
```

Argumenty funkcji wykrywacz:

- Wzorzec
- Obraz
- Tolerancja
- Litera egipska

Analiza kodu - poszukiwanie wzorca

1. Na obrazie szukane są obszary podobne do danego wzorca w stopniu ustalonym przez zadany próg prawdopodobieństwa.
2. Ustalenie progu (tolerancji) jest konieczne, aby wybrane kształty nie były znajdywane też na fragmentach innych liter.
3. Najbardziej znaczącą funkcją jest *match template*:
   - porównuje zadany wzorzec z zadanym obrazkiem
   - Jej wynikiem jest tablica, w której znajdują się współczynnik korelacji dla poszczególnych pikseli obrazu
   - Wartość współczynników może wahać się od -1 do 1
   - Tablica wynikowa może miec wielkość mniejszą niz ilość pikseli w obrazie podstawowym

Analiza kodu - poszukiwanie wzorca

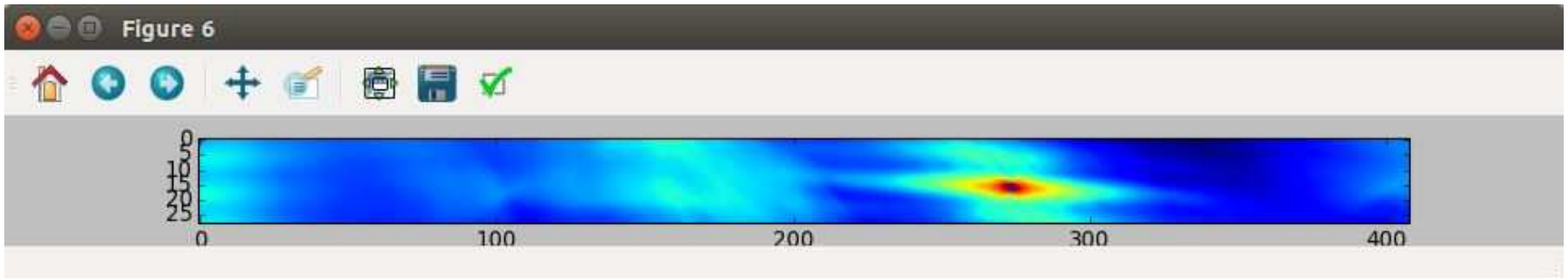
1. Na obrazie szukane są obszary podobne do danego wzorca w stopniu ustalonym przez zadany próg prawdopodobieństwa.
2. Ustalenie progu (tolerancji) jest konieczne, aby wybrane kształty nie były znajdywane też na fragmentach innych liter.
3. Najbardziej znaczącą funkcją jest *match template*:
   - porównuje zadany wzorzec z zadanym obrazkiem
   - Jej wynikiem jest tablica, w której znajdują się współczynnik korelacji dla poszczególnych pikseli obrazu
   - Wartość współczynników może wahać się od -1 do 1
   - Tablica wynikowa może miec wielkość mniejszą niz ilość pikseli w obrazie podstawowym

Analiza kodu - poszukiwanie wzorca

1. Na obrazie szukane są obszary podobne do danego wzorca w stopniu ustalonym przez zadany próg prawdopodobieństwa.
2. Ustalenie progu (tolerancji) jest konieczne, aby wybrane kształty nie były znajdywane też na fragmentach innych liter.
3. Najbardziej znaczącą funkcją jest *match template*:
   - porównuje zadany wzorzec z zadanym obrazkiem
   - Jej wynikiem jest tablica, w której znajdują się współczynnik korelacji dla poszczególnych pikseli obrazu
   - Wartość współczynników może wahać się od -1 do 1
   - Tablica wynikowa może miec wielkość mniejszą niz ilość pikseli w obrazie podstawowym

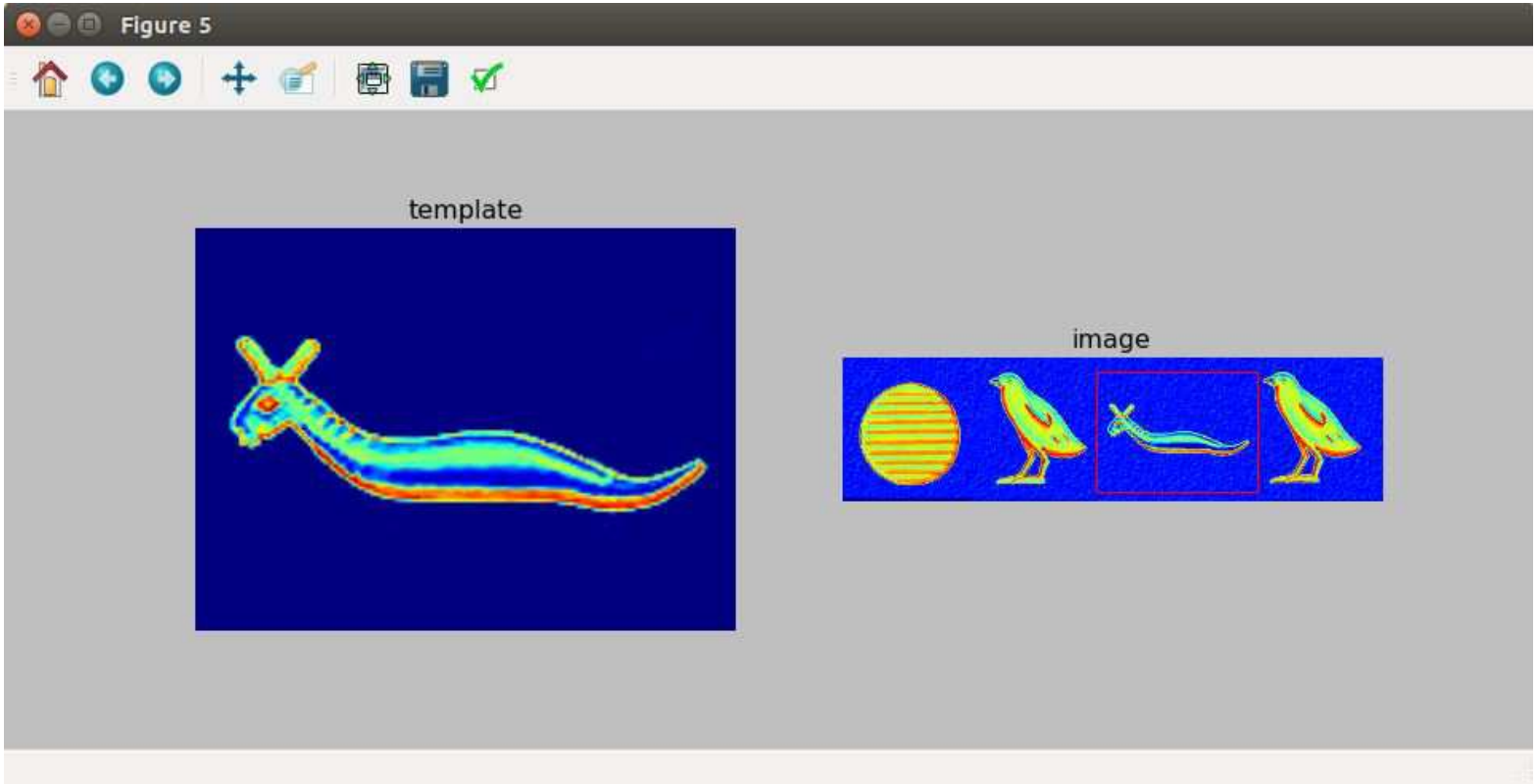
Analiza kodu - wynik

Wizualizacja wyniku operacji *match template*:



- Czerwone miejsca – największe prawdopodobieństwo
- Granatowe miejsca - najmniejsze prawdopodobieństwo

Analiza kodu - wynik



- Wyszukany hieroglif to ten z największym współczynnikiem korelacji
- Wokół znalezionego hieroglifu rysowana jest czerwona ramka
- Wyświetlenie wzoru i obrazu

Analiza kodu - wynik

```
literka.append([x , litera_egipska,np.max(result2)])
```

- Odnalezione hieroglify są gromadzone w liście “literka”
- Do listy trafia współrzędna x znalezionego hieroglifu, jej odpowiednik w formie litery oraz informacyjnie jaka jest wartość współczynnika korelacji w tym miejscu

Analiza kodu

```
result2[ij]=0

result2[(y,x-1)]=0
result2[(y,x+1)]=0
```

- Zerowanie argumentu tablicy, w którym znajdował się znaleziony hieroglif
- Zerowanie elementów obok

Analiza kodu

```
while (np.max(result2) > tolerancja):
    ij=np.unravel_index(np.argmax(result2), result.shape)
    x, y= ij[::-1]
    x1.append(x)
    y1.append(y)

    literka.append([x , litera_egipska,np.max(result2)])

    result2[ij]=0

    result2[(y,x-1)]=0
    result2[(y,x+1)]=0


    i=i+1
    rect = plt.Rectangle((x1[i], y1[i]), wwzor, hwzor, edgecolor='r', facecolor='none')
    ax2.add_patch(rect)
```

Czynności wykonywane wcześniej są następnie wykonywane w pętli do momentu, kiedy najwyższy znaleziony argument będzie mniejszy niz ustalony próg (tolerancja)

Analiza kodu - Słownik

Lista “literka”:

```
Znalezione:
[[136, 'u', 0.94119245], [431, 'u', 0.87158859], [5, 'kh', 0.99339813],
[273, 'f', 0.96394712]]
```

- Wszystkie zebrane w liście “literka” dane są sortowane po współrzędnej x
- Po sortowaniu wyciągamy z listy argumenty odpowiadające egipskim literom
- Łączymy wyciągnięte litery w wyraz
- Tłumaczenie za pomocą słownika

```
slowniczek={'khufu':"Cheops" , 'ptolmiis':"Ptolemeusz", 'kliopatra':"Kleopatra"}
```

Wynik programu

```
Znalezione literki (PO SORTOWANIU) :
['kh', 'u', 'f', 'u']

Po egipsku:
khufu

Po polsku:
Cheops
```

Podsumowanie

- Cel projektu został osiągnięty : udało się dokonać tłumaczenia 3 wybranych obrazów
- Program można wykorzystywać do zamieniania różnych obrazów rastrowych na tekst pisany.
- Istnieje możliwość udoskonalania i rozbudowywania programu poprzez dodawanie nowych symboli oraz nowych słów do słownika.