MOwNiT

Zastosowania DFT

Laboratorium 9

Sprawozdanie

Maria Polak

Wstęp Teoretyczny

1. Discrete Fourier Transform

Procedurą matematyczna używana do wyznaczenia zawartości harmonicznej, lub częstotliwościowej, sygnału dyskretnego. Przetwarza skończoną sekwencję próbek w sekwencję liczb zaspolonych opisującą funkcję częstotliwości. Do przetwarzania obrazów używane jest wielowymiarowe DFT.

Zadanie 1. Analiza obrazów

Celem tego zadania było napisanie programu który szuka oraz zlicza wystąpienia danego wzorca na obrazie przy użyciu ***DFT*** oraz twierdzenia o splocie.

Napisany program zaczyna swoje działanie od wczytania obrazu oraz wzorca. W przypadku tekstu obrazki zostają przekształcone w czarno-białe, a następnie kolory zostają zamienione (tekst biały, tło czarne). W przypadku zdjęcia ławicy ryb, zdjęcie zostaje ograniczone do kanału czerwonego dla którego rozpoznawanie daje subiektywnie najlepsze wyniki.

Kolejnym krokiem jest policzenie korelacji między obrazami, którą można wyrazić w nastepujący sposób:

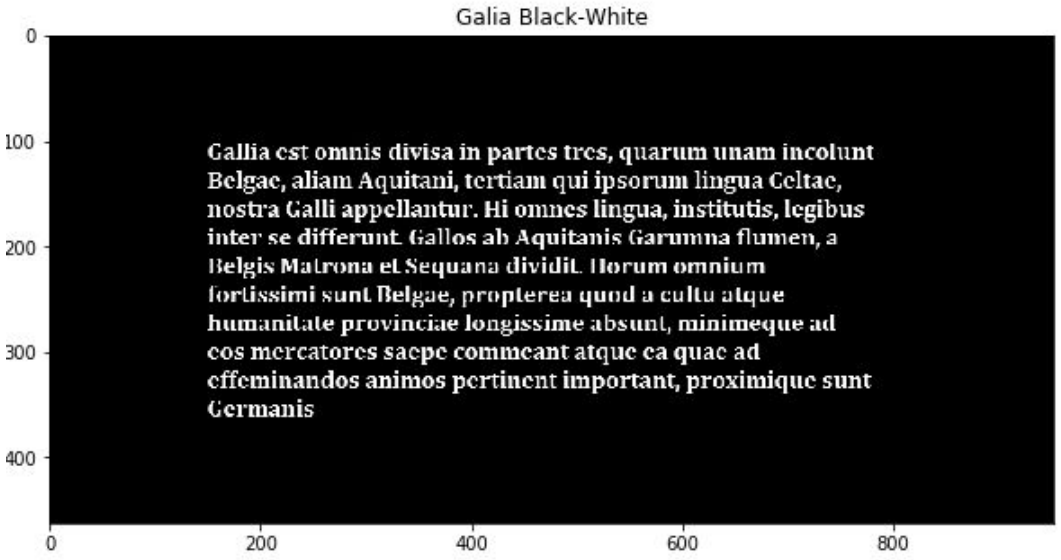
$$C = real(ifft2(fft2(image) .* fft2(rot90(pattern, 2), imageHeight, imageWidth)))$$

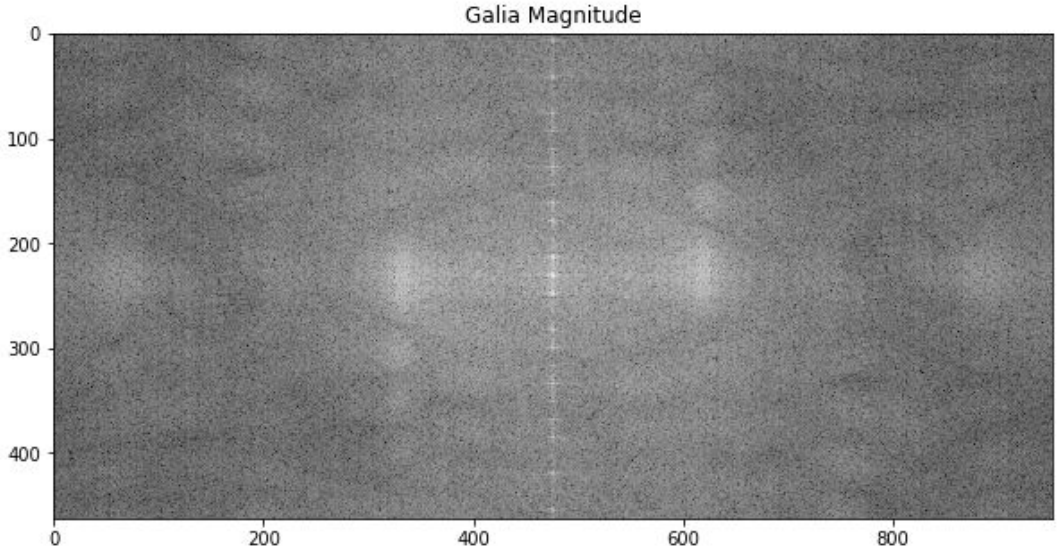
Obydwa obrazy zostają przekształcone poprzez transformację Fouriera (w wypadku wzorca zostaje on uzupełniony zerami do rozmiaru obrazu wejściowego oraz odwrócony o $180°$), następnie przemnożone (*element-wise*) przez siebie. Wynik mnożenia zostaje przekształcony spowrotem poprzez *Inverse DFT*.

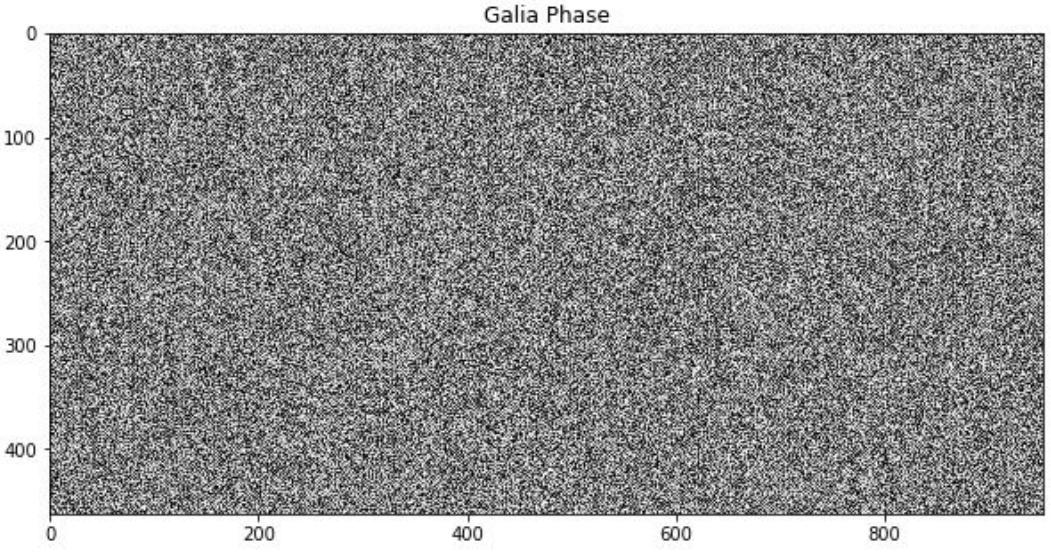
Jako kryterium znalezienia wzorca na obrazie, program przyjmuje korelację wyższą niż maksymalna wartość korelacji przemnożona przez pewne $\mu \in (0, 1]$. Współczynnik μ został wyzanczony empirycznie. Metoda ta nie powinna być stosowana, gdy wzorzec może nie występować na obrazie (nawet jeśli nie występuje to nasz program go gdzięś ‘znajdzie’).

Program był testowany dla dwóch obrazów: tekstu oraz ławicy ryb. Poniżej znajdują się odpowiednio dla tekstu/ryb: obraz ze zmienionymi kolorami, wartości modułu współczynników Fouriera, wartości fazy oraz obraz wejściowy z zaznaczonymi wystąpieniami wzorca.

Tekst:

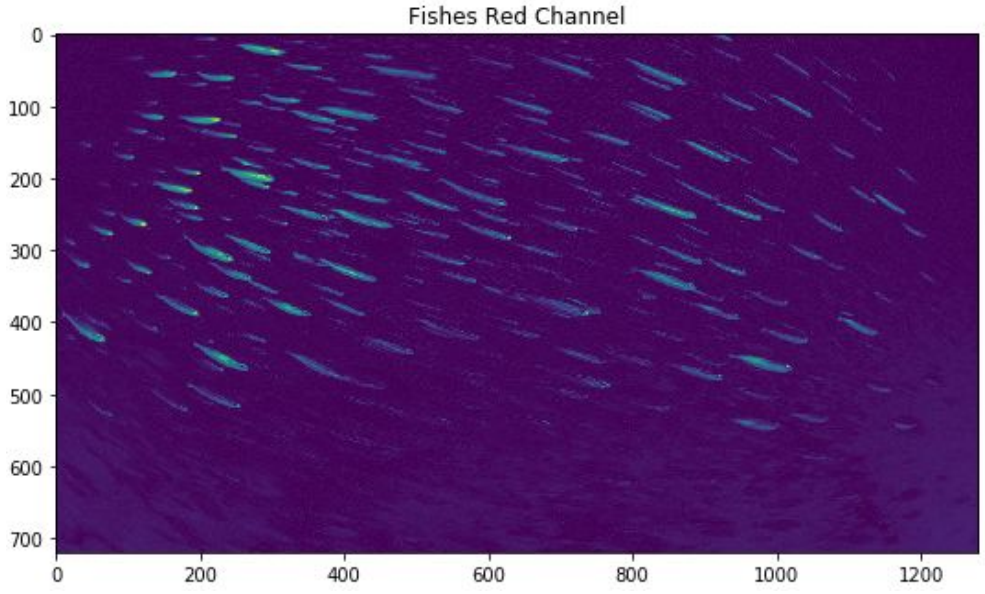

Galia Black-White
Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. Hi omnes lingua, institutis, legibus inter se differunt. Gallos ab Aquitanis Garumna flumen, a Belgis Matrona et Sequana dividit. Horum omnium fortissimi sunt Belgae, propterea quod a cultu atque humanitate provinciae longissime absunt, minimeque ad eos mercatores saepe commeant atque ea quae ad effeminandos animos pertinent important, proximique sunt Germanis
0
100
200
300
400
0
200
400
600
800

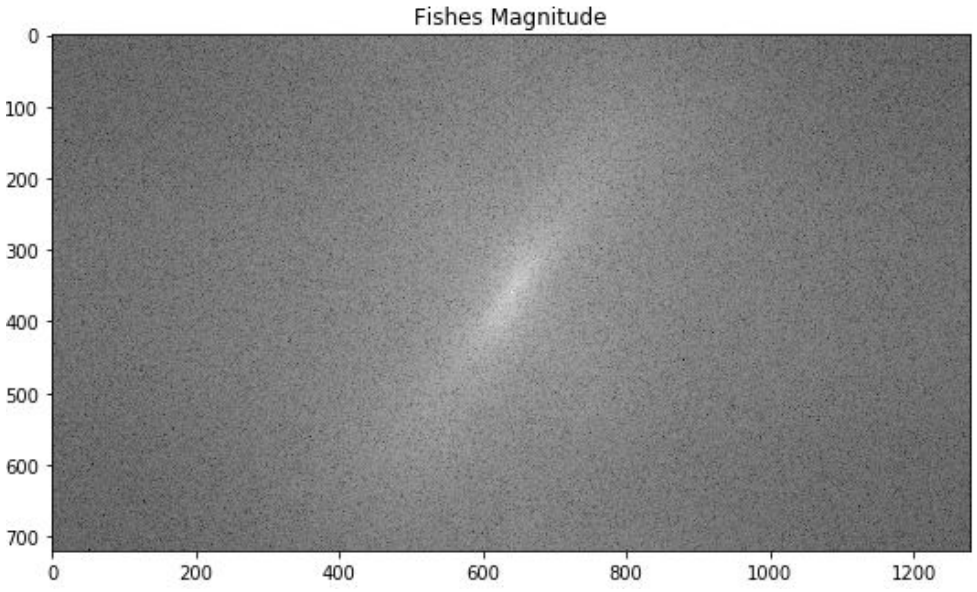

Galia Magnitude
0
100
200
300
400
0
200
400
600
800

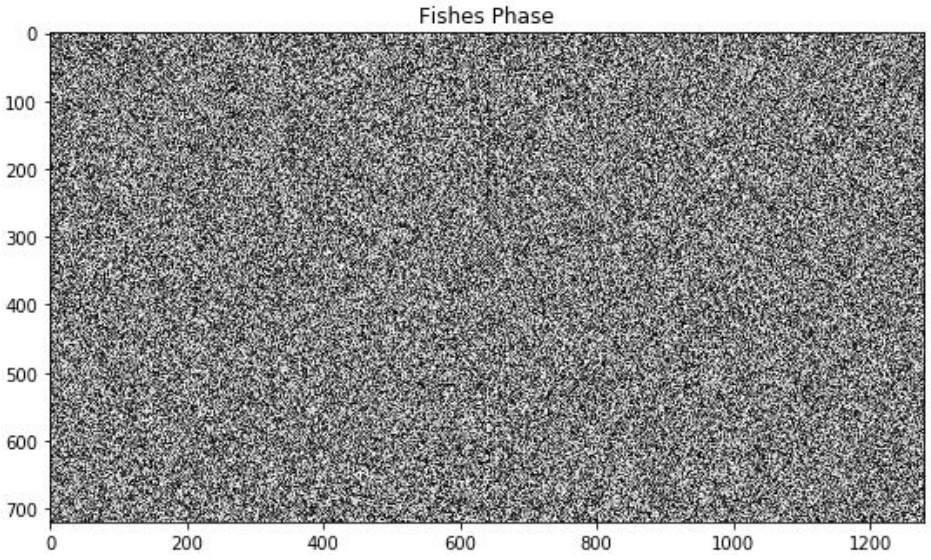

Galia Phase
0
100
200
300
400
0
200
400
600
800

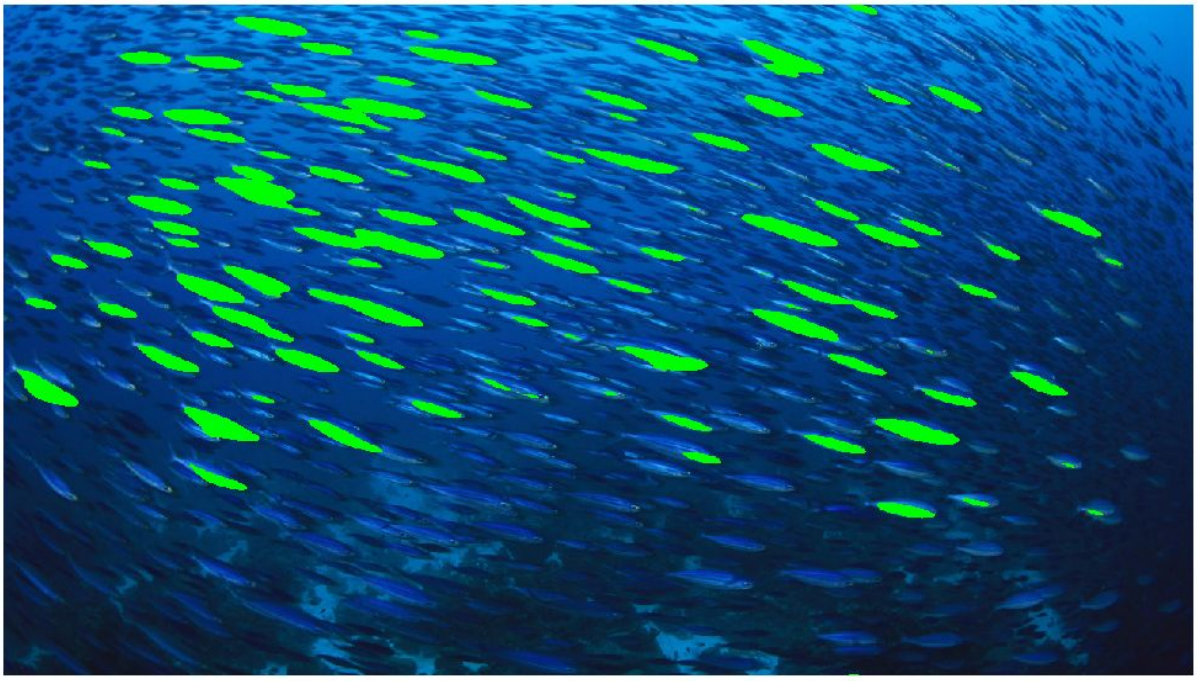
Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. Hi omnes lingua, institutis, legibus inter se differunt. Gallos ab Aquitanis Garumna flumen, a Belgis Matrona et Sequana dividit. Horum omnium fortissimi sunt Belgae, propterea quod a cultu atque humanitate provinciae longissime absunt, minimeque ad eos mercatores saepe commeant atque ea quae ad effeminandos animos pertinent important, proximique sunt Germanis

Ławica ryb:









Wnioski:

- Dobranie odpowiedniej wartości μ jest bardzo ważne. W przypadku gdy jego wartość była zbyt niska, wzorzec był ropoznawany w miejscach, w których nie występował. Gdy wartość była zbyt wysoka program nie znajdował wszystkich wystąpień.
- Na zdjęciu ławicy nie wszystkie ryby zostały znalezione oraz nie wszystkie zaznaczone są poprawne. Wynika to z faktu że na zdjęciu ryby są różnych wielkości, obrócone pod różnym kątem oraz kolorystycznie zlewające się z tłem.

Zadanie 2. OCR

Celem zadania było napisanie programu przkształcającego obraz w tekst.

Program został napisany analogicznie do zadania pierwszego. Obrazy wejściowe oraz wzorce znaków są przygotowywane za pomoca biblioteki PIL (*ImageFont*).

Program był testowany dla dwóch różnych tekstów:

Pierwszy tekst - abc

Drugi tekst - dolphin

Obrazki testowe zostały przygotowane w:

- dwóch czcionkach
    - czcionka bezszeryfowa - Arial
    - czcionka szeryfowa - Times New Roman
- dwóch wariantach plików
    - JPG
    - PNG
- obrócone w czterech kierunkach
    - 0°
    - 90°
    - 180°
    - 270°

Między OCR, a pierwszym zadaniem było kilka różnic.

Pierwszą różnicą jest redukcja szumu na obrazie wejściowym. Została ona zrealizowana za pomocą SVD. Przed obliczeniem korelacji obraz wejściowy jest modyfikowany poprzez funkcję *reduceNoice(A,k)*:

```
def reduceNoice(A, k):
    U, S, V = np.linalg.svd(A)
    U = np.matrix(U[:, :k])
    S = np.diag(S[:k])
    V = np.matrix(V[:k, :])
    return U*S*V
```

Drugą różnicą była możliwa rotacja obrazu wejściowego. Do sprawdzenia czy i w jakim kierunku jest obrócony obraz została napisana funkcja *isTextFlipped,* która oblicza, a następnie porównuje najwyższe korelacje litery ‘e’ z obrazem wejściowym i wybiera obrót który zwrócił najwyższy wynik. Litera ‘e’ została wybrana w związku z jej najczęstszym występowaniem w słowach języka angielskiego.

```
def isTextFlipped(text,font,changeImage):
    rots = {"0":0, "1":0, "2":0, "3":0}
    pat = getLetterPattern('e',font) #e bo jest popularne :)))
    pat = changeImage(pat)
    charFFT = np.fft.fft2(np.rot90(pat,2), s=text.shape)
    charFFT2 = np.fft.fft2(np.rot90(pat,2), s=(text.shape[1],text.shape[0]))

    textFFT = np.fft.fft2(text)
    rots[0] = np.max(getCorrelationMatrix(textFFT,charFFT))

    textFFT = np.fft.fft2(np.rot90(text,1))
    rots[1] = np.max(getCorrelationMatrix(textFFT,charFFT2))

    textFFT = np.fft.fft2(np.rot90(text,2))
    rots[2] = np.max(getCorrelationMatrix(textFFT,charFFT))

    textFFT = np.fft.fft2(np.rot90(text,3))
    rots[3] = np.max(getCorrelationMatrix(textFFT,charFFT2))

    return max(rots, key=rots.get)
```

Trzecią różnicą było wypisywanie tekstu po jego rozpoznaniu. By było to możliwe, znaki po rozpoznaniu zostały zapisywane w strukturze, która przechowywała ich współrzędne, wartość korelacji oraz samą znak. Żeby lepiej określić kolejność znaków w tekście współrzędne były dzielone przez odpowiednio szerokość i wysokość litery ‘l’ co umożliwiło ich segregację do “szufladek” (rząd szufladek odpowiada jednej linii tekstu). Jeżeli w danej szufladce program znalazł dwie litery to wybierana była ta o wyższej korelacji.

Spacja wstawiana jest w miejsca, gdzie przerwa między dwoma rozpoznanymi znakami jest większa niż $\frac{9}{10}$ szerokości spacji.

By wyniki były jak najlepsze dla każdego formatu oraz czcionki, wartość μ była dobierana osobno dla każdej ich kombinacji.

Wyniki działania OCR:

```
abcdefglhijk mnopqrstu wxyz
12.34567890
,?
```

Abc, Times New Roman, plik png

```
.
do4ph?n s aco mmozn n m o
aqguai m mm s ywi in
t0e nfr or3 cxe cxea
```

Dolphin, Times New Roman, plik png

```
abcdefglhijk mnopqrstu wxyz
12.34567890
,?
```

Abc, Times New Roman, plik jpg

```
.
do4ph?n aco7mmon n 7me of
aqguaic 7ma mmas ywithb n
t0he nfr rder cxe cxea
```

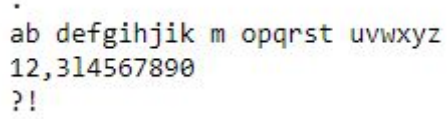
Dolphin, Times New Roman, plik jpg

```
.
abcdefghjk m opqrst uvwxyz
123l45,67890
?!
```

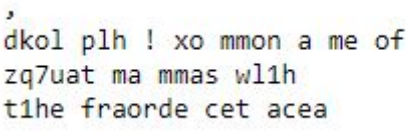
Abc, Arial, plik png

```
,
dkoplh ! acxo mmo a me of
aq7uat ma mma4s wtl1h
t1he fraorder cet acea
```

Dolphin, Arial, plik png



Abc, Arial, plik jpg



Dolphin, Arial, plik jpg

Program ma funkcję zliczania wystąpień każdego znaku:

```
.
do4ph?n s aco mmozn n m o
aqguai m mm s ywi in
t0e nfr or3 cxe cxea

| .:1 || 0:1 || 3:1 || 4:1 || ?:1 || a:4 || c:3 || d:1 || e:3 || f:1
|| g:1 || h:1 || i:3 || m:6 || n:5 || o:5 || p:1 || q:1 || r:2 || s:2
|| t:1 || u:1 || w:1 || x:2 || y:1 || z:1 |
```

Wnioski:

- Dla obróconych obrazków wyniki działania są identyczne. Dla kilku testów z tekstami niezawierającymi litery ‘e’ wyniki były poprawne, lecz bezpieczniej jest przyjąć jako założenie, że tekst ma zawierać tę literę.
- OCR działa lepiej dla czcionek szeryfowych. Litery przez swoje zdobienia są bardziej charakterystyczne i mniej z nich ma zawierające się w innych bitmapy.
- Problematyczne są znaki **.,** praktycznie na żadnym obrazku tekstowym nie zostały one poprawnie rozpoznane.
- OCR działałby o wiele lepiej gdyby dla każdego znaku z osobna dobrać wartość μ.