

Przetwarzanie obrazów  
Sprawozdanie z laboratorium

Małgorzata Wiśniewska

Warszawa, 2020

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>3</b>
1.1	Format obrazów . . . . .	3
1.2	Instrukcja obsługi programu . . . . .	3
<b>2</b>	<b>Operacje ujednolicania obrazów</b>	<b>4</b>
2.1	Ujednolicanie obrazów szarych geometryczne . . . . .	4
2.2	Ujednolicanie obrazów szarych rozdzielczościowe . . . . .	7
2.3	Ujednolicanie obrazów RGB geometryczne . . . . .	9
2.4	Ujednolicanie obrazów RGB rozdzielczościowe . . . . .	12
<b>3</b>	<b>Operacje sumowania arytmetycznego obrazów szarych</b>	<b>15</b>
3.1	Sumowanie obrazów szarych . . . . .	15
3.1.1	Sumowanie obrazu z określoną stałą . . . . .	15
3.1.2	Sumowanie dwóch obrazów . . . . .	15
3.2	Mnożenie obrazów szarych . . . . .	15
3.2.1	Mnożenie obrazu przez określoną stałą . . . . .	15
3.2.2	Mnożenie obrazu przez inny obraz . . . . .	15
3.3	Mieszanie obrazów z określonym współczynnikiem . . . . .	15
3.4	Potęgowanie obrazu z zadaną potęgą . . . . .	15
3.5	Dzielenie obrazów szarych . . . . .	15
3.5.1	Dzielenie obrazu przez zadaną stałą . . . . .	15
3.5.2	Dzielenie obrazu przez inny obraz . . . . .	15
3.6	Pierwiastkowanie obrazu . . . . .	15
3.7	Logarytmowanie obrazu . . . . .	15
<b>4</b>	<b>Operacje sumowania arytmetycznego obrazów barwowych</b>	<b>16</b>
4.1	Sumowanie obrazów barwowych . . . . .	16
4.1.1	Sumowanie obrazu z określoną stałą . . . . .	16
4.1.2	Sumowanie dwóch obrazów . . . . .	16
4.2	Mnożenie obrazów barwowych . . . . .	16
4.2.1	Mnożenie obrazu przez określoną stałą . . . . .	16
4.2.2	Mnożenie obrazu przez inny obraz . . . . .	16
4.3	Mieszanie obrazów z określonym współczynnikiem . . . . .	16
4.4	Potęgowanie obrazu z zadaną potęgą . . . . .	16
4.5	Dzielenie obrazów barwowych . . . . .	16
4.5.1	Dzielenie obrazu przez zadaną stałą . . . . .	16
4.5.2	Dzielenie obrazu przez inny obraz . . . . .	16
4.6	Pierwiastkowanie obrazu . . . . .	16
4.7	Logarytmowanie obrazu . . . . .	16

<b>5</b>	<b>Operacje geometryczne na obrazie</b>	<b>17</b>
5.1	Przemieszczanie obrazu o zadany wektor . . . . .	17
5.2	Skalowanie obrazu . . . . .	17
5.2.1	Skalowanie jednorodne . . . . .	17
5.2.2	Skalowanie niejednorodne . . . . .	17
5.3	Obracanie obrazu o dowolny kąt . . . . .	17
5.4	Symetrie obrazu . . . . .	17
5.4.1	Symetria względem osi OX . . . . .	17
5.4.2	Symetria względem osi OY . . . . .	17
5.4.3	Symetria względem zadanej prostej . . . . .	17
5.5	Wycinanie fragmentów obrazów . . . . .	17
5.6	Kopiowanie fragmentów obrazów . . . . .	17
<b>6</b>	<b>Operacje na histogramie obrazu szarego</b>	<b>18</b>
6.1	Obliczanie histogramu . . . . .	18
6.2	Przemieszczanie histogramu . . . . .	18
6.3	Rozciąganie histogramu . . . . .	18
6.4	Progowanie lokalne . . . . .	18
6.5	Progowanie globalne . . . . .	18
<b>7</b>	<b>Operacje na histogramie obrazu barwowego</b>	<b>19</b>
7.1	Obliczanie histogramu . . . . .	19
7.2	Przemieszczanie histogramu . . . . .	19
7.3	Rozciąganie histogramu . . . . .	19
7.4	Progowanie 1 progowo lokalne . . . . .	19
7.5	Progowanie 1 progowo globalne . . . . .	19
7.6	Progowanie wieloprogowe lokalne . . . . .	19
7.7	Progowanie wieloprogowe globalne . . . . .	19

# Rozdział 1

## Wstęp

1.1 Format obrazów

1.2 Instrukcja obsługi programu

# Rozdział 2

## Operacje ujednolicania obrazów

Operacje ujednolicania obrazów dzieli się na dwa etapy. Pierwszym etapem jest ujednolicanie geometryczne, drugim jest ujednolicenie rozdzielczościowe. W prezentowanym programie ujednolicane są dwa obrazy, w taki sposób, że mniejszy z nich jest doprowadzany do takiego samego rozmiaru jak większy. Skutkuje to wygenerowaniem nowego obrazu o zwiększonej ilości pikseli niż początkowa wartość. Dzięki zastosowaniu tego typu ujednolicania w efekcie nie następuje widoczny spadek jakości.

### 2.1 Ujednolicanie obrazów szarych geometryczne

#### Opis algorytmu

Operacje geometrycznego ujednolicania polega na wyrównaniu liczby pikseli w kolumnach i wierszach w obu obrazach, poprzez zwiększenie liczby pikseli w kolumnach i wierszach mniejszego z obrazów.

1. Wybierz największą wysokość i największą szerokość spośród obu obrazów.
2. Jeśli dany obraz ma mniejszą wysokość lub szerokość, wypełnij różnicę pikselami o wartości 1, tak, żeby wysokość i szerokość obu obrazów była równa.

#### Efekty wykorzystania algorytmu



(a) Obraz 1: 256x256



(b) Obraz 2: 512x512

Rysunek 2.1: Obrazy wejściowe



(a) Obraz 1: 512x512



(b) Obraz 2: 512x512

Rysunek 2.2: Obrazy wyjściowe



(a) Obraz 1: 369x480



(b) Obraz 2: 623x640

Rysunek 2.3: Obrazy wejściowe



(a) Obraz 1: 623x640



(b) Obraz 2: 623x640

Rysunek 2.4: Obrazy wyjściowe

## Kod źródłowy algorytmu

```
def geoUnificationGrey(self):
    # porównaj wielkość obrazów, jeżeli są tego samego rozmiaru
    ↪ nie rob nic
    if self.biggerPicture == 0 and self.smallerPicture == 0:
        print('Both pictures have the same size')
        return 0
    # stworz tablice zer do zapisu efektu algorytmu
    result = np.zeros((self.maxLength, self.maxWidth), np.uint8)
    startWidthIndex = int(round((self.maxWidth - self.minWidth) /
    ↪ 2))
```

```

startLengthIndex = int(round((self.maxLength - self.minLength
    ↪ ) / 2))
for w in range(0, self.minWidth):
    for l in range(0, self.minLength):
        result[l + startLengthIndex, w +
            ↪ startWidthIndex] = self.matrix[l, w]
#zapisz zunifikowany obraz
path = self.ex + self.smallerPictureName + '_' + self.
    ↪ biggerPictureName + '.png'
self.saver.savePictureFromArray(result, 'L', path)

```

## 2.2 Ujednolicanie obrazów szarych rozdzielczościowe

### Opis algorytmu

Operacja rozdzielczościowego ujednolicania obrazów następuje po ujednoliceniu geometrycznym obrazów wejściowych. Polega na wypełnieniu obrazu pikslami. Brakujące piksele powinny zostać zinterpolowane.

1. Wypełnij cały obraz pikslami o znanej wartości zachowując pewien odstęp między nimi, gdzie odstępem będą piksele o wartości 0.
2. Każdemu pikslowi o nieznannej wartości przypisz średnią wartość znanych ( $\neq 0$ ) pikseli z jego bezpośredniego otoczenia.

### Efekty wykorzystania algorytmu



(a) Obraz 1: 512x512



(b) Obraz 2: 512x512

Rysunek 2.5: Obrazy wejściowe po ujednoliceniu geometrycznym





(a) Obraz 1: 512x512



(b) Obraz 2: 512x512

Rysunek 2.6: Obrazy wyjściowe bez interpolacji



(a) Obraz 1: 512x512



(b) Obraz 2: 512x512

Rysunek 2.7: Obrazy wyjściowe po interpolacji

## Kod źródłowy algorytmu

```
def resolutionUnificationGrey(self):
    print('Beginning of resolution unification for two grey pictures.
    ↪ ')
    if self.biggerPicture == 0 and self.smallerPicture == 0:
        print('Both pictures have the same size')
        return 0
    scaleFactorLength = float(self.maxLength / self.minLength)
    scaleFactorWidth = float(self.maxWidth / self.minWidth)
    result = np.zeros((self.maxLength, self.maxWidth), np.uint8)
    for l in range(self.minLength):
        for w in range(self.minWidth):
            if w % 2 == 0:
                pomL = int(scaleFactorLength * l)
                pomW = int(round(scaleFactorWidth * w))
```

```

        result[pomL, pomW] = self.matrix[l, w]
    elif w % 2 == 1:
        pomL = int(round(scaleFactorLength * l))
        pomW = int(scaleFactorWidth * w)
        result[pomL, pomW] = self.matrix[l, w]
    # zapisz obraz bez interpolacji
    path = self.ex + self.smallerPictureName + '_' + self.
        ↪ biggerPictureName + '_withoutInterpolation.png'
    self.saver.savePictureFromArray(result, 'L', path)
    # interpolacja
    for l in range(self.maxLength):
        for w in range(self.maxWidth):
            value = 0
            count = 0
            if result[l, w] == 0:
                for lOff in range(-1, 2):
                    for wOff in range(-1, 2):
                        lSave = l if ((l + lOff) > (self.maxLength - 2)
                            ↪ ) | ((l + lOff) < 0) else (l + lOff)
                        wSave = w if ((w + wOff) > (self.maxWidth - 2))
                            ↪ | ((w + wOff) < 0) else (w + wOff)
                        if result[lSave, wSave] != 0:
                            value += result[lSave, wSave]
                            count += 1
                        result[l, w] = value / count
    # zapisz obraz po interpolacji
    path = self.ex + self.smallerPictureName + '_' + self.
        ↪ biggerPictureName + '_withInterpolation.png'
    self.saver.savePictureFromArray(result, 'L', path)
    print('Finished_resolution_unification.')

```

## 2.3 Ujednolicanie obrazów RGB geometryczne

### Opis algorytmu

Operacje geometrycznego ujednolicania polega na wyrównaniu liczby pikseli w kolumnach i wierszach w obu obrazach, poprzez zwiększenie liczby pikseli w kolumnach i wierszach mniejszego z obrazów.

1. Wybierz największą wysokość i największą szerokość spośród obu obrazów.
2. Jeśli dany obraz ma mniejszą wysokość lub szerokość, wypełnij różnicę pikselami o wartości 1 dla każdego z kanałów (R,G,B), tak, żeby wysokość i szerokość obu obrazów była równa.

### Efekty wykorzystania algorytmu

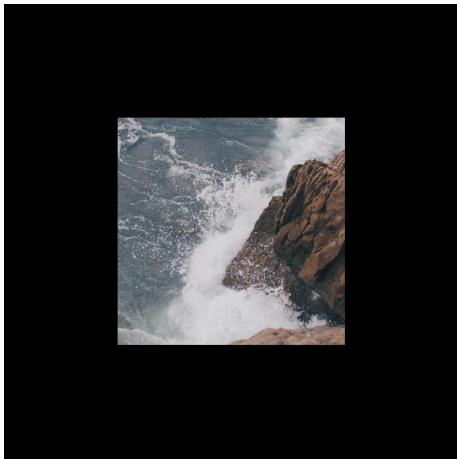


(a) Obraz 1: 512x512



(b) Obraz 2: 1025x1025

Rysunek 2.8: Obrazy wejściowe



(a) Obraz 1: 1025x1025

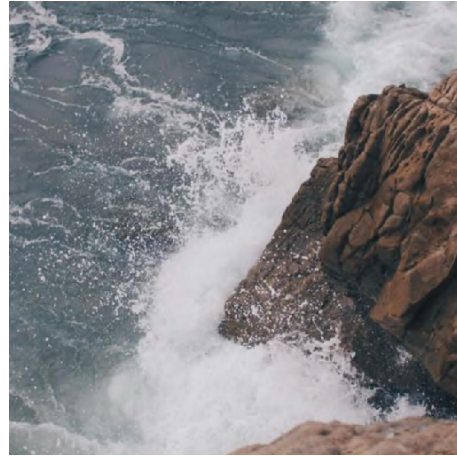


(b) Obraz 2: 1025x1025

Rysunek 2.9: Obrazy wyjściowe

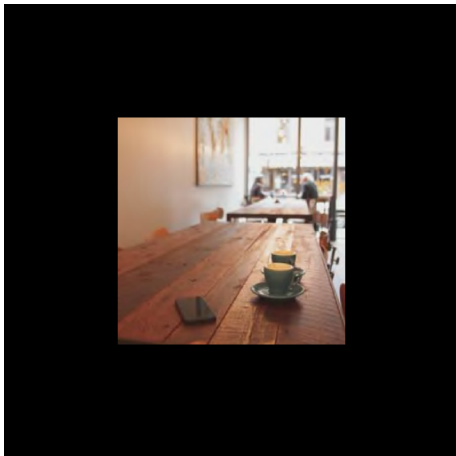


(a) Obraz 1: 256x256

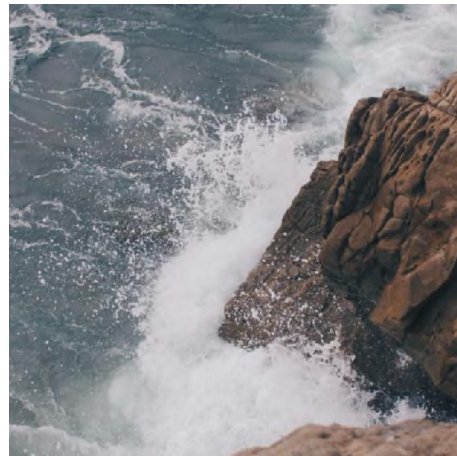


(b) Obraz 2: 512x512

Rysunek 2.10: Obrazy wejściowe



(a) Obraz 1: 512x512



(b) Obraz 2: 512x512

Rysunek 2.11: Obrazy wyjściowe

## Kod źródłowy algorytmu

```
def geoUnificationRGB(self):
    if self.biggerPicture == 0 and self.smallerPicture == 0:
        print('Both pictures have the same size')
        return 0
    # stwórz tablice z zerami jako odstawa dla unifikacji
    result = np.full((self.maxLength, self.maxWidth, 3), 0, np.uint8)
    startWidthIndex = int(round((self.maxWidth - self.minWidth) / 2))
    startLengthIndex = int(round((self.maxLength - self.minLength) /
    ↪ 2))
    for w in range(0, self.minWidth):
        for l in range(0, self.minLength):
            result[l + startLengthIndex, w + startWidthIndex] = self.
            ↪ matrix[w, l]
    # zapisz zunifikowany obraz
```

```
path = self.ex + self.smallerPictureName + '_' + self.  
    ↪ biggerPictureName + '.png'  
self.saver.savePictureFromArray(result, 'RGB', path)
```

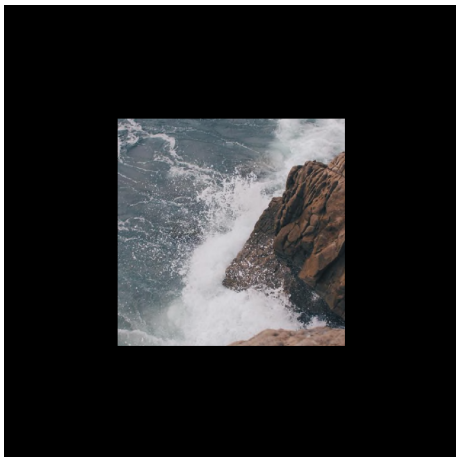
## 2.4 Ujednolicanie obrazów RGB rozdzielczościowe

### Opis algorytmu

Operacja rozdzielczościowego ujednolicania obrazów następuje po ujednoliceniu geometrycznym obrazów wejściowych. Polega na wypełnieniu obrazu pikslami. Brakujące piksele powinny zostać zinterpolowane.

1. Wypełnij cały obraz pikslami o znanej wartości zachowując pewien odstęp między nimi, gdzie odstępem będą piksele o wartości 0.
2. Każdemu pikslowi (ze wszystkich kanałów - R, G, B) o nieznaney wartości przypisz średnią wartość znanych ( $\neq 0$ ) piksli z jego bezpośredniego otoczenia.

### Efekty wykorzystania algorytmu



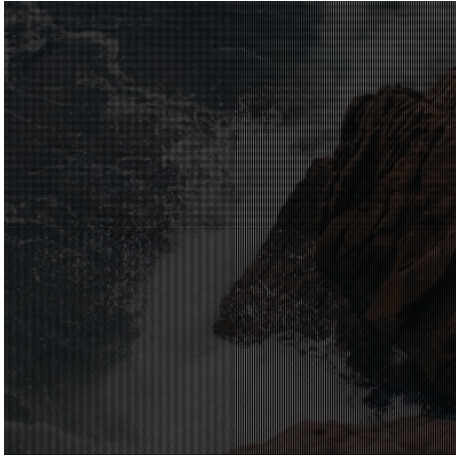
(a) Obraz 1: 1025x1025



(b) Obraz 2: 1025x1025

Rysunek 2.12: Obrazy wejściowe po ujednoliceniu geometrycznym



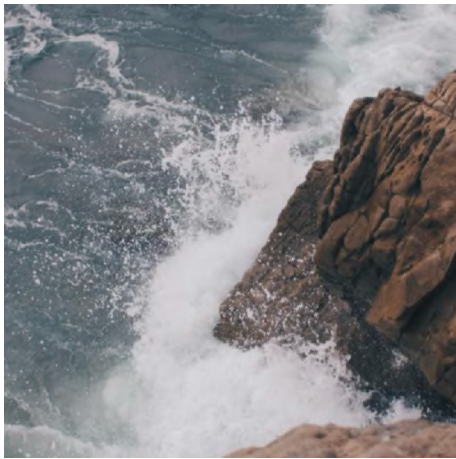


(a) Obraz 1: 1025x1025



(b) Obraz 2: 1025x1025

Rysunek 2.13: Obrazy wyjściowe bez interpolacji



(a) Obraz 1: 1025x1025



(b) Obraz 2: 1025x1025

Rysunek 2.14: Obrazy wyjściowe po interpolacji

## Kod źródłowy algorytmu

```
def resolutionUnificationGrey(self):
    print('Beginning of resolution unification for two grey pictures.
    ↪ ')
    if self.biggerPicture == 0 and self.smallerPicture == 0:
        print('Both pictures have the same size')
        return 0
    scaleFactorLength = float(self.maxLength / self.minLength)
    scaleFactorWidth = float(self.maxWidth / self.minWidth)
    result = np.zeros((self.maxLength, self.maxWidth), np.uint8)
    for l in range(self.minLength):
        for w in range(self.minWidth):
            if w % 2 == 0:
                pomL = int(scaleFactorLength * l)
                pomW = int(round(scaleFactorWidth * w))
```

```

        result[pomL, pomW] = self.matrix[l, w]
    elif w % 2 == 1:
        pomL = int(round(scaleFactorLength * l))
        pomW = int(scaleFactorWidth * w)
        result[pomL, pomW] = self.matrix[l, w]
# zapisz obraz bez interpolacji
path = self.ex + self.smallerPictureName + '_' + self.
    ↪ biggerPictureName + '_withoutInterpolation.png'
self.saver.savePictureFromArray(result, 'L', path)
# interpolacja
for l in range(self.maxLength):
    for w in range(self.maxWidth):
        value = 0
        count = 0
        if result[l, w] == 0:
            for lOff in range(-1, 2):
                for wOff in range(-1, 2):
                    lSave = l if ((l + lOff) > (self.maxLength - 2)
                                ↪ ) | ((l + lOff) < 0) else (l + lOff)
                    wSave = w if ((w + wOff) > (self.maxWidth - 2))
                                ↪ | ((w + wOff) < 0) else (w + wOff)
                    if result[lSave, wSave] != 0:
                        value += result[lSave, wSave]
                        count += 1
            result[l, w] = value / count
# zapisz obraz po interpolacji
path = self.ex + self.smallerPictureName + '_' + self.
    ↪ biggerPictureName + '_withInterpolation.png'
self.saver.savePictureFromArray(result, 'L', path)
print('Finished_resolution_unification.')

```

## Rozdział 3

# Operacje sumowania arytmetycznego obrazów szarych

### 3.1 Sumowanie obrazów szarych

#### 3.1.1 Sumowanie obrazu z określoną stałą

#### 3.1.2 Sumowanie dwóch obrazów

### 3.2 Mnożenie obrazów szarych

#### 3.2.1 Mnożenie obrazu przez określoną stałą

#### 3.2.2 Mnożenie obrazu przez inny obraz

### 3.3 Mieszanie obrazów z określonym współczynnikiem

### 3.4 Potęgowanie obrazu z zadaną potęgą

### 3.5 Dzielenie obrazów szarych

#### 3.5.1 Dzielenie obrazu przez zadaną stałą

#### 3.5.2 Dzielenie obrazu przez inny obraz

### 3.6 Pierwiastkowanie obrazu

### 3.7 Logarytmowanie obrazu



## Rozdział 4

# Operacje sumowania arytmetycznego obrazów barwowych

### 4.1 Sumowanie obrazów barwowych

#### 4.1.1 Sumowanie obrazu z określoną stałą

#### 4.1.2 Sumowanie dwóch obrazów

### 4.2 Mnożenie obrazów barwowych

#### 4.2.1 Mnożenie obrazu przez określoną stałą

#### 4.2.2 Mnożenie obrazu przez inny obraz

### 4.3 Mieszanie obrazów z określonym współczynnikiem

### 4.4 Potęgowanie obrazu z zadaną potęgą

### 4.5 Dzielenie obrazów barwowych

#### 4.5.1 Dzielenie obrazu przez zadaną stałą

#### 4.5.2 Dzielenie obrazu przez inny obraz

### 4.6 Pierwiastkowanie obrazu

### 4.7 Logarytmowanie obrazu

## Rozdział 5

### Operacje geometryczne na obrazie

5.1 Przemieszczanie obrazu o zadany wektor

5.2 Skalowanie obrazu

5.2.1 Skalowanie jednorodne

5.2.2 Skalowanie niejednorodne

5.3 Obracanie obrazu o dowolny kąt

5.4 Symetrie obrazu

5.4.1 Symetria względem osi OX

5.4.2 Symetria względem osi OY

5.4.3 Symetria względem zadanej prostej

5.5 Wycinanie fragmentów obrazów

5.6 Kopiowanie fragmentów obrazów

## Rozdział 6

# Operacje na histogramie obrazu szarego

6.1 Obliczanie histogramu

6.2 Przemieszczanie histogramu

6.3 Rozciąganie histogramu

6.4 Progowanie lokalne

6.5 Progowanie globalne

## Rozdział 7

# Operacje na histogramie obrazu barwowego

7.1 Obliczanie histogramu

7.2 Przemieszczanie histogramu

7.3 Rozciąganie histogramu

7.4 Progowanie 1 progowe lokalne

7.5 Progowanie 1 progowe globalne

7.6 Progowanie wieloprogowe lokalne

7.7 Progowanie wieloprogowe globalne