

Przetwarzanie Obrazów: Sprawozdanie

Damian Ubowski Maciej Tarach

Warszawa, 2019

Spis treści

1 Wstęp	7
1.1 Format obrazu	7
1.1.1 Struktura formatu	7
1.1.2 Przykładowa struktura IFF	8
1.1.3 Instrukcja obsługi programu	8
2 Operacje ujednoliciania obrazów	9
2.1 Ujednolicenie obrazów szarych geometryczne	9
2.2 Ujednolicenie obrazów szarych rozdzielczościowe	12
2.3 Ujednolicenie obrazów RGB geometryczne	15
2.4 Ujednolicenie obrazów RGB rozdzielczościowe	18
3 Operacje sumowania arytmetycznego obrazów szarych	19
3.1 Sumowanie (określonej) stałej z obrazem	20
3.2 Sumowanie dwóch obrazów	20
3.3 Mnożenie obrazu przez zadaną liczbę	20
3.4 Mnożenie obrazu przez inny obraz	20
3.5 Mieszanie obrazów z określonym współczynnikiem	20
3.6 Potęgowanie obrazu (z zadaną potęgą)	20
3.7 Dzielenie obrazu przez (zadaną) liczbę	20
3.8 Dzielenie obrazu przez przez inny obraz	20
3.9 Pierwiastkowanie obrazu	20
3.10 Logarytmowanie obrazu	20
4 Operacje sumowania arytmetycznego obrazów barwowych	21
4.1 Sumowanie (określonej) stałej z obrazem	22
4.2 Sumowanie dwóch obrazów	22
4.3 Mnożenie obrazu przez zadaną liczbę	22
4.4 Mnożenie obrazu przez inny obraz	22
4.5 Mieszanie obrazów z określonym współczynnikiem	22
4.6 Potęgowanie obrazu (z zadaną potęgą)	22

4.7	Dzielenie obrazu przez (zadaną) liczbę	22
4.8	Dzielenie obrazu przez przez inny obraz	22
4.9	Pierwiastkowanie obrazu	22
4.10	Logarytmowanie obrazu	22
5	Operacje geometryczne na obrazie	23
5.1	Przemieszczenie obrazu o zadany wektor	23
5.2	Jednorodne skalowanie obrazu	23
5.3	Niejednorodne skalowanie obrazu	23
5.4	Obracanie obrazu o dowolny kąt	23
5.5	Symetrie względem osi układu	23
5.6	Symetrie względem zadanej prostej	23
5.7	Wycinanie fragmentów obrazu	23
5.8	Kopiowanie fragmentów obrazów	23
6	Operacje na histogramie obrazu szarego	25
6.1	Obliczanie histogramu	25
6.2	Przemieszczanie histogramu	25
6.3	Rozciąganie histogramu	25
6.4	Progowanie lokalne	25
6.5	Progowanie globalne	25
7	Operacje na histogramie obrazu barwowego	27
7.1	Obliczanie histogramu	27
7.2	Przemieszczanie histogramu	27
7.3	Rozciąganie histogramu	27
7.4	Progowanie 1-progowe lokalne	27
7.5	Progowanie wielo-progowe lokalne	27
7.6	Progowanie 1-progowe globalne	27
7.7	Progowanie wielo-progowe globalne	27
8	Operacje morfologiczne na obrazach binarnych	29
8.1	Okrawanie (erozja)	29
8.2	Nakładanie (dylatacja)	29
8.3	Otwarcie	29
8.4	Zamknięcie	29
9	Operacje morfologiczne na obrazach szarych	31
9.1	Okrawanie (erozja)	31
9.2	Nakładanie (dylatacja)	31
9.3	Otwarcie	31

SPIS TREŚCI 5

9.4 Zamknięcie	31
10 Filtrowanie liniowe i nieliniowe	33
10.1 Filtr dolnoprzepustowy uśredniający	33
10.2 Filtr dolnoprzepustowy Gaussowski	33
10.3 Operator Roberts'a	33
10.4 Operator Prewitt'a	33
10.5 Operator Sobel'a	33
10.6 Filtr kompasowy	33
10.7 Gradient wektora kierunkowego	33
10.8 Filtr medianowy	33
10.9 Filtr maksymalny	33
10.10 Filtr minimalny	33
10.11 Filtr płaskorzeźbowy	33

Rozdział 1

Wstęp

1.1 Format obrazu

Wybranym przez nas formatem obrazów cyfrowych jest DjVu, który jest oparty na zaawansowanej metodzie segmentacji obrazu. Tworzenie pliku DjVu polega na rozdzieleniu dowolnie skomplikowanego obrazu na odrębne warstwy, a następnie poddaniu warst odrębnym optymalizacjom i kompresjom. Format ten stosuje ładowanie progresywne, kodowanie arytmetyczne, oraz kompresję stratną dzięki czemu przy minimalnej ilości przestrzeni dyskowej można delektować się obrazami i dokumentami w wysokiej jakości.

1.1.1 Struktura formatu

Pliki DjVu rozpoczynają się od swojej “Magic number” potwierdzającej rodzaj pliku i mającej wartość `0x41 0x54 0x26 0x54`. Następnie czerpiąc inspirację ze struktury IFF (**Interchange File Format**) plik dzieli się na kawałki (*ang. chunks*) zawierające interesujące nas cenne dane. Takie jak szerokość lub wysokość obrazu, dpi, informacje o kolorach, rozmieszczeniu pikseli, etc. Każdy kawałek składający się z ID typu, długości zawartości i samej zawartości tworzy zwarty format. Identyfikator typu określa rolę w jakiej przyjdzie służyć kawałkowi. Do dyspozycji ma ich całkiem sporo, ale uwzględniając najbardziej przydatne w naszym kontekście to ograniczymy liczbę do:

- * BGjp - warstwa tylna przechowywana przy użyciu kodowania JPEG.
- * BFjp - warstwa przednia w formacie JPEG.
- * INFO - opisuje wysokość, szerokość, rozdzielczość, wersję kodera, oraz flagi wskazujące na obrót obrazu.

1.1.2 Przykładowa struktura IFF

FORM:DJVU [14260]

INFO [10]

Sjbz [13133]

FG44 [181]

BG44 [935]

Powyższa struktura przedstawia dokument składający się z jednej strony, na co wskazuje *FORM:DJVU*, wraz z grafiką. Ten znacznik informuje, że mamy do czynienia z kontenerem o długości 14260 bajtów, który może zawierać inne kawałki dokumentu. Zgodnie z konwencją, po identyfikatorze typu i informacji o długości znajduje się zawartość kawałka. W tym wypadku jak i w każdym innym po *FORM:DJVU* powinno znaleźć się *INFO* z podstawowymi informacjami. Jeśli konwencji i wymagań specyfikacyjnych stało się zadość wtedy czas nastąpił na jakieś wizualne atrakcje takie jak *Sjbz*, czyli masce wyboru pomiędzy kolorami z warstwy przedniej (*FG44*) i tylnej (*BG44*).

1.1.3 Instrukcja obsługi programu

W celu uruchomienia kodu źródłowego będzie niezbędny:

- * [DjVuLibre](#) ($\geq 3.5.21$)
- * [Python](#) (≥ 2.6 lub $3.X$)
- * [Cython](#) (≥ 0.19 , lub ≥ 0.20 dla Python 3)
- * [pkg-config](#) (POSIX)

Rozdział 2

Operacje ujednolicania obrazów

Ujednolicanie obrazów oznacza sprowadzenie ich do wspólnego gruntu pod względem określonego parametru. W tym wypadku będziemy ujednolicać obrazy pod względem geometrycznym (ilości kolumn i wierszy pikseli) i następnie rozdzielczościowym (wypełnienia pikselami). Sekwencyjność tych operacji jak i one same nie są w stanie spowodować spadku jakości obrazu.

2.1 Ujednolicenie obrazów szarych geometryczne

Algorytm

Opis

Algorytm geometrycznego ujednolicenia obrazów ma za zadanie sprowadzić oba obrazy do tej samej liczby pikseli w każdym wierszu i każdej kolumnie.

Kroki

1. Porównaj szerokości i wysokości obu obrazów i wybierz największe.
2. Jeśli pierwszy lub drugi obraz mają szerokość lub wysokość mniejszą od największej dostępnej to:
 - (a) Utwórz czarne tło
 - (b) Przenieś z wyśrodkowaniem piksle na czarne tło
3. Jeśli żaden z warunków jest niespełniony to nie rób nic

Rysunek 2.1: Przed uruchomieniem algorytmu (od lewej): obraz 1 (1067x1067, 300dpi), obraz 2 (2133x2133, 300dpi)



Rysunek 2.2: Po uruchomieniu algorytmu (od lewej): obraz 1 (2133x2133, 300dpi), obraz 2 (2133x2133, 300dpi)



Kod źródłowy algorytmu

```

def geometricGray(self):
    print('geometric gray unification start')
    width, height = self.firstDecoder.width, self.firstDecoder.
                    height
    if width < self.maxWidth or height < self.maxHeight:
        # Create black background
        firstResult = numpy.zeros((self.maxHeight, self.maxWidth),
                                  numpy.uint8)
        # Copy smaller image to bigger
        startWidthIndex = int(round((self.maxWidth - width) / 2))
        startHeightIndex = int(round((self.maxHeight - height) /
                                      2))
        pixelsBuffer = self.firstDecoder.getPixels()
        for h in range(0, height):
            for w in range(0, width):
                firstResult[h + startHeightIndex, w + startWidthIndex] =
                    pixelsBuffer[h, w]
        img = Image.fromarray(firstResult, mode='L')
        img.save('Resources/ggUnification_1.png')
        print('first image done')

    width, height = self.secondDecoder.width, self.
                    secondDecoder.height
    if width < self.maxWidth or height < self.maxHeight:
        # Create black background
        secondResult = numpy.zeros((self.maxHeight, self.maxWidth),
                                   numpy.uint8)
        # Copy smaller image to bigger
        startWidthIndex = int(round((self.maxWidth - width) / 2))
        startHeightIndex = int(round((self.maxHeight - height) /
                                      2))
        pixelsBuffer = self.secondDecoder.getPixels()
        for h in range(0, height):
            for w in range(0, width):
                secondResult[h + startHeightIndex, w + startWidthIndex] =
                    pixelsBuffer[h, w]
        img = Image.fromarray(secondResult, mode='L')
        img.save('Resources/ggUnification_2.png')
        print('second image done')
    print('geometric gray unification done')

```

2.2 Ujednolicenie obrazów szarych rozdzielczościowe

Algorytm

Opis

Po użyciu ujednolicenia geometrycznego można użyć ujednolicenia rozdzielczościowego, które przeskakuje obraz z mniejszej postaci do większej dzięki czemu nie zostanie nam czarna ramka wokół obrazu. Wynikiem będzie większy obraz niż początkowo bez czarnego obwodu wokół. Mniejszy obraz można przeskalać do większych wymiarów przenosząc wszystkie piksele z uwzględnieniem luk pomiędzy nimi i następnie użycia interpolacji do zamazania tych luk. Interpolacja działa na zasadzie pobierania wartości z okolicznych pikseli i wyciągania z nich średniej, która posłuży jako baza koloru dla nowego piksela.

Kroki

1. Ustalenie nowych wymiarów obrazu
2. Obliczenie odległości pomiędzy pikselami ($scaleFactoryH$, $scaleFactoryW$)
3. Naniesienie pikseli z mniejszego obrazu na większy z uwzględnieniem luk
4. Interpolacja

Kod źródłowy algorytmu

```
def rasterGray(self):
    print('raster gray unification start')
    self._scaleUp(self.firstDecoder, 'Resources/rgUnification_1
                                         .png')
    print('first image done')
    self._scaleUp(self.secondDecoder, 'Resources/
                                         rgUnification_2.png')
    print('second image done')
    print('raster gray unification done')

def _scaleUp(self, decoder, outputPath):
    width, height = decoder.width, decoder.height
```

2.2. UJEDNOLICENIE OBRAZÓW SZARYCH ROZDZIELCZOŚCIOWE13

Rysunek 2.3: Skutki braku interpolacji



Rysunek 2.4: Przed uruchomieniem algorytmu (od lewej): obraz 1 (2133x2133, 300dpi), obraz 2 (2133x2133, 300dpi)



Rysunek 2.5: Po uruchomieniu algorytmu (od lewej): obraz 1 (2133x2133, 300dpi), obraz 2 (2133x2133, 300dpi)



```

scaleFactoryW = float(self.maxWidth) / width
scaleFactoryH = float(self.maxHeight) / height
if width < self.maxWidth or height < self.maxHeight:
    pixelsBuffer = decoder.getPixels()
    result = numpy.zeros((self.maxHeight, self.maxWidth),
                         numpy.uint8)

    # Fill values
    for h in range(height):
        for w in range(width):
            if w%2 == 0:
                result[int(scaleFactoryH * h), int(round(
                    scaleFactoryW * w)) + 1] =
                    pixelsBuffer[h, w]
            if w%2 == 1:
                result[int(round(scaleFactoryH * h)) + 1, int(
                    scaleFactoryW * w)] =
                    pixelsBuffer[h, w]

    # Interpolate
    self._interpolate(result)
    img = Image.fromarray(result, mode='L')
    img.save(outputPath)

def _interpolate(self, result):
    for h in range(self.maxHeight):
        for w in range(self.maxWidth):

```

```
value = 0
count = 0
if result[h, w] == 0:
    for iOff in range(-1, 2):
        for jOff in range(-1, 2):
            iSafe = h if ((h + iOff) > (self.maxLength - 2)) |
                ((h + iOff) < 0) else (h +
                iOff)
            jSafe = w if ((w + jOff) > (self.maxLength - 2)) |
                ((w + jOff) < 0) else (w +
                jOff)
            if result[iSafe, jSafe] != 0:
                value += result[iSafe, jSafe]
                count += 1
            result[h, w] = value / count
```

2.3 Ujednolicenie obrazów RGB geometryczne

Algorytm

Opis

Algorytm geometrycznego ujednolicenia obrazów ma za zadanie sprowadzić oba obrazy do tej samej liczby pikseli w każdym wierszu i każdej kolumnie.

Kroki

1. Porównaj szerokości i wysokości obu obrazów i wybierz największe.
2. Jeśli pierwszy lub drugi obraz mają szerokość lub wysokość mniejszą od największej dostępnej to:
 - (a) Utwórz czarne tło
 - (b) Przenieś z wyśrodkowaniem piksle na czarne tło z uwzględnieniem każdego z kanałów RGB
3. Jeśli żaden z warunków jest niespełniony to nie rób nic

Rysunek 2.6: Przed uruchomieniem algorytmu (od lewej): obraz 1 (512x512, 300dpi), obraz 2 (1024x1024, 300dpi)



Rysunek 2.7: Po uruchomieniu algorytmu (od lewej): obraz 1 (1024x1024, 300dpi), obraz 2 (1024x1024, 300dpi)



Kod źródłowy algorytmu

```
def geometricColor(self):
    print('geometric color unification start')
    self.firstDecoder.setColor()
    width, height = self.firstDecoder.width, self.firstDecoder.
                                height
    if width < self.maxWidth or height < self.maxHeight:
        result = self._paintInMiddleColor(self.firstDecoder)
        img = Image.fromarray(result, 'RGB')
        img.save('Resources/gcUnification_1.png')
        print('first image done')

    self.secondDecoder.setColor()
    width, height = self.secondDecoder.width, self.
                                secondDecoder.height
    if width < self.maxWidth or height < self.maxHeight:
        result = self._paintInMiddleColor(self.secondDecoder)
        img = Image.fromarray(result, 'RGB')
        img.save('Resources/gcUnification_2.png')
        print('second image done')
    print('geometric color unification done')

def _paintInMiddleColor(self, decoder):
    # Create black background
    result = numpy.full((self.maxHeight, self.maxWidth, 3), 0,
                        numpy.uint8)
    # Copy smaller image to bigger
    width, height = decoder.width, decoder.height
    startWidthIndex = int(round((self.maxWidth - width) / 2))
    startHeightIndex = int(round((self.maxHeight - height) / 2))
    pixelsBuffer = decoder.getPixels24Bits()
    for h in range(0, height):
        for w in range(0, width):
            result[h + startHeightIndex, w + startWidthIndex] =
                pixelsBuffer[h, w]
    return result
```

2.4 Ujednolicenie obrazów RGB rozdzielczościowe

Algorytm

Opis

Kod źródłowy algorytmu

Rozdział 3

Operacje sumowania arytmetycznego obrazów szarych

- 3.1 Sumowanie (określonej) stałej z obrazem**
- 3.2 Sumowanie dwóch obrazów**
- 3.3 Mnożenie obrazu przez zadaną liczbę**
- 3.4 Mnożenie obrazu przez inny obraz**
- 3.5 Mieszanie obrazów z określonym współczynnikiem**
- 3.6 Potęgowanie obrazu (z zadaną potęgą)**
- 3.7 Dzielenie obrazu przez (zadaną) liczbę**
- 3.8 Dzielenie obrazu przez inny obraz**
- 3.9 Pierwiastkowanie obrazu**
- 3.10 Logarytmowanie obrazu**

Rozdział 4

Operacje sumowania arytmetycznego obrazów barwowych

- 4.1 Sumowanie (określonej) stałej z obrazem
- 4.2 Sumowanie dwóch obrazów
- 4.3 Mnożenie obrazu przez zadaną liczbę
- 4.4 Mnożenie obrazu przez inny obraz
- 4.5 Mieszanie obrazów z określonym współczynnikiem
- 4.6 Potęgowanie obrazu (zadaną potęgą)
- 4.7 Dzielenie obrazu przez (zadaną) liczbę
- 4.8 Dzielenie obrazu przez inny obraz
- 4.9 Pierwiastkowanie obrazu
- 4.10 Logarytmowanie obrazu

Rozdział 5

Operacje geometryczne na obrazie

- 5.1 Przesunięcie obrazu o zadany wektor**
- 5.2 Jednorodne skalowanie obrazu**
- 5.3 Niejednorodne skalowanie obrazu**
- 5.4 Obracanie obrazu o dowolny kąt**
- 5.5 Symetria względem osi układu**
- 5.6 Symetria względem zadanej prostej**
- 5.7 Wycinanie fragmentów obrazu**
- 5.8 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

26 ROZDZIAŁ 6. OPERACJE NA HISTOGRAMIE OBRAZU SZAREGO

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 wielo-progowe lokalne
- 7.6 Progowanie 1-progowe globalne
- 7.7 Progowanie wielo-progowe globalne

28 ROZDZIAŁ 7. OPERACJE NA HISTOGRAMIE OBRAZU BARWOWEGO

Rozdział 8

Operacje morfologiczne na obrazach binarnych

8.1 Okrawanie (erozja)

8.2 Nakładanie (dylatacja)

8.3 Otwarcie

8.4 Zamknięcie

30 ROZDZIAŁ 8. OPERACJE MORFOLOGICZNE NA OBRAZACH BINARNYCH

Rozdział 9

Operacje morfologiczne na obrazach szarych

9.1 Okrawanie (erozja)

9.2 Nakładanie (dylatacja)

9.3 Otwarcie

9.4 Zamknięcie

32 ROZDZIAŁ 9. OPERACJE MORFOLOGICZNE NA OBRAZACH SZARYCH

Rozdział 10

Filtrowanie liniowe i nieliniowe

- 10.1 Filtr dolnoprzepustowy uśredniający**
- 10.2 Filtr dolnoprzepustowy Gaussowski**
- 10.3 Operator Roberts'a**
- 10.4 Operator Prewitt'a**
- 10.5 Operator Sobel'a**
- 10.6 Filtr kompasowy**
- 10.7 Gradient wektora kierunkowego**
- 10.8 Filtr medianowy**
- 10.9 Filtr maksymalny**
- 10.10 Filtr minimalny**
- 10.11 Filtr płaskorzeźbowy**