



Politechnika
Wrocławska

Inżynieria Obrazów

Laboratorium nr 4

Implementacja formatów graficznych

Szymon Datko & Mateusz Gniewkowski

szymon.datko@pwr.edu.pl , mateusz.gniewkowski@pwr.edu.pl

Wydział Elektroniki,
Politechnika Wrocławska

semestr letni 2020/2021



Cel ćwiczenia

1. Zrozumienie, w jaki sposób w plikach reprezentowane są dane.
2. Nauczenie się, jak można zapisywać dane graficzne w plikach.
3. Zapoznanie się z pomysłowymi elementami algorytmu JPEG.

Uwaga!

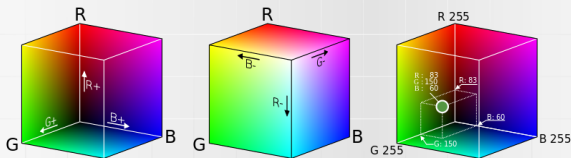
W typowych, życiowych przypadkach, większość omawianych w tym ćwiczeniu rzeczy realizuje się za pomocą gotowych funkcji i bibliotek. Tutaj zaś zależy nam na przestudiowaniu pewnych fundamentalnych mechanizmów.

Słowo na temat plików

- W dużym uogólnieniu: plik stanowi jakiś zbiór danych (Bajtów).
- Z plików można **czytać** dane, ale i można do nich **zapisywać** dane.
- Sposób organizacji i możliwe właściwości plików określa **system plików**.
- Zwyczajowo wyróżnia się dwa rodzaje plików:
 - ▶ pliki **binarne** – bardziej podstawowe,
 - ciągi Bajtów o ustalonej strukturze, która określa sens danych,
 - np. że kolejne 4 Bajty opisują 32-bitową liczbę całkowitą,
 - taką strukturę fachowo nazywa się **formatem pliku**,
 - ▶ pliki **tekstowe**,
 - ciągi Bajtów o arbitralnie ustalonym ich znaczeniu,
 - znaczenie to określane jest przez **kodowanie znaków**,
 - np. ASCII, CP1250, ISO-8859-2, UTF-8.

Słowo na temat reprezentacji kolorów

- **Kolor** jest pewnym subiektywnym wrażeniem, każdy postrzega go inaczej.
- Zwykle ludzie posługują się mało precyzyjną reprezentacją barw:
 - czerwony, khaki, ciepły piasek, błękit narodów zjednoczonych.
- Przy pracy z kolorem lepiej jest posłużyć się jednoznacznymi zapisami,
 - jednym z takich modeli jest na przykład model **RGB**,
 - barwę opisujemy przy pomocy trzech ustalonych kolorów → wektor.
- 3 wektory ortogonalne modelu rozpinają przestrzeń możliwych odcieni,
 - ▶ wykonalne jest płynne przejście między dwoma kolorami!

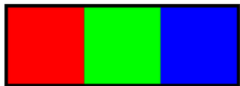


Format PPM (1/3)

^{BINARY} PORTABLE PIXMAP



ANGE ALBERTINI 
<http://www.corkami.com>



```

<signature> <whitespace>
P6 <width> <whitespace> <height> <whitespace>
3 1 <max. value> <whitespace>
255
ÿ ÿ ÿ
<raw RGB values>
FF 00 00 00 FF 00 00 00 FF
  
```

THE PPM FORMAT WAS DEVELOPED IN 1988, FOR NETPBM
 IT EASILY TURNS RAW DATA INTO A RGB PICTURE

Format PPM (2/3)

- Jest to bardzo prosty format zapisu danych graficznych w pliku.
- Wykorzystuje on model RGB, który jest stosunkowo intuicyjny dla nas.
- Istnieją dwa warianty kolorowe tego modelu:
 - ▶ **P3** – stosuje zapis w plikach tekstowych,
 - ▶ **P6** – wykorzystuje format binarny.
- Każdy plik w tym formacie składa się z:
 - ▶ nagłówka – zawsze zapisanego tekstowo, zawierającego określenie:
 - ▶ wariantu zapisu,
 - ▶ szerokości i wysokości obrazka,
 - ▶ palety odcieni (np. 255 dla zakresu 0-255);
 - ▶ danych obrazu – wartości kolejnych składowych każdego piksela:
 - ▶ w wariantie P3: zapisane tekstowo, rozdzielone białymi znakami, z dodatkowym znakiem nowej linii na końcu pliku,
 - ▶ w wariantie P6: kolejno po sobie, każda wartość to jeden Bajt.

Format PPM (3/3)

Przykład:

```
image = numpy.array([[255, 0, 0, 0, 255, 0, 0, 0, 255]], dtype=numpy.uint8)
```


Rezultat:


```
[sdatko@polluks programy]$ hexdump -C example-binary.ppm
0000  50 36 20 33 20 31 20 32  35 35 0a ff 00 00 00 ff  |P6 3 1 255.....|
0010  00 00 00 ff                                     |....|
0014
```

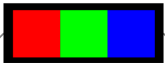
```
[sdatko@polluks programy]$ hexdump -C example-ascii.ppm
0000  50 33 20 33 20 31 20 32  35 35 0a 32 35 35 20 30  |P3 3 1 255.255 0|
0010  20 30 20 30 20 32 35 35  20 30 20 30 20 30 20 32  | 0 0 255 0 0 0 2|
0020  35 35 0a                                       |55.|
0023
```

Pliki PPM można wyświetlić za pomocą programu **Gimp**, **feh** oraz **IrfanView**.

Format PNG (1/4)

PORTABLE **N**ETWORK **G**RAPHICS **ANGE ALBERTINI**
<http://www.corkami.com> 





0 1 2 3 4 5 6 7 8 9 A B C D E F

```

00: 89 .P .N .G 0D 0A 1A 0A 00 00 00 0D .I .H .D .R
10: 00 00 00 03 00 00 01 08 02 00 00 00 94 82 83
20: E3 00 00 00 15 .I .D .A .T 08 1D 01 0A 00 F5 FF
30: 00 FF 00 00 00 FF 00 00 00 FF 0E FB 02 FE E9 32
40: 61 E5 00 00 00 00 .I .E .N .D AE 42 60 82
  
```

FIELDS	VALUES			
SIGNATURE	signature <code>\x89 PNG \r\n \x1a \n</code>			
HEADER	size id width height bpp color compression filter interlace CRC32	<code>0x0000000D IHDR 0x00000003 0x00000001 0x08 0x02 RGB 0x00 DEFLATE 0x00 0x00 0x948283E3</code>		
	DATA	size id window size method level / dict. checksum last block block type data length length line filter adler32 CRC32	<code>0x00000015 IDAT 0b00001000 0b00001000 DEFLATE 0b0001101 0x081D % 31 = 0 0b00000001 FINAL 0b00000001 RAW 0x000A 0xFFFF5 0x00 NONE FF 00 00 00 FF 00 00 00 FF 0xEFB02FE 0xE93261E5</code>	
		END	size id CRC32	<code>0x00000000 IEND 0xAE426082</code>

Labels for data fields: ZLIB, DEFLATE, PIXELS

download hi-res pictures at <http://pics.corkami.com>
 order prints, stickers, shirts at <http://prints.corkami.com>

Format PNG (2/4)

- Popularny obecnie standard zapisu grafik w internecie.
- Jego przewagą nad formatem PPM jest znacznie mniejszy rozmiar plików.
- Bazuje na obrazie w formacie RGB, podobnie jak pliki PPM,
 - ▶ istnieje także wariant wspierający przezroczystość – model RGBA.
- Plik w formacie PNG składa się z kilku części:
 - ▶ sygnatury – identyfikującej plik w formacie PNG,¹
 - ▶ nagłówka – zawierającego podstawowe informacje na temat obrazu,
 - ▶ danych obrazu – skompresowanej tablicy Bajtów RGB,
 - ▶ zakończenia pliku.

¹ podpowiedź w nawiązaniu do kursu Systemy Operacyjne 2 → takich sygnatur poszukuje program `file`

Format PNG (3/4)

- Nagłówek, dane obrazu i zakończenia mają podobną strukturę:
 - ▶ rozmiar – wielkość **zawartości** sekcji, 4 Bajty (typ uint32),
 - ▶ identyfikator sekcji – 4 Bajty ASCII, napis IHDR, IDAT lub IEND,
 - ▶ (zawartość sekcji – pusta w przypadku zakończenia),
 - ▶ suma kontrolna – 4 Bajty, liczona dla identyfikatora oraz zawartości.
- Informacje zawarte w nagłówku to między innymi:
 - ▶ rozmiar obrazka w pikselach (wysokość i szerokość),
 - ▶ informację na temat głębi koloru (liczba bitów na każdą składową),
 - ▶ informację o modelu barw (2 = RGB, 6 = RGBA),
 - ▶ sposób kompresji danych, trybie filtrowania i przeplotu.
- Zawartość stanowi tablica Bajtów, jak w formacie P6, ale dodatkowo:
 - ▶ każdy wiersz obrazu poprzedzony jest Bajtem zerowym (`\x00`),
 - ▶ całość jest skompresowana za pomocą biblioteki `zlib`.

Format PNG (4/4)

Przykład:

```
image = numpy.array([[[[255, 0, 0], [ 0, 255, 0]],  
                    [[ 0, 0, 255], [ 55, 55, 0]]],  
                    dtype=numpy.uint8)
```

Rezultat:

```
[sdatko@polluks programy]$ hexdump -C example.png  
0000  89 50 4e 47 0d 0a 1a 0a  00 00 00 0d 49 48 44 52  |.PNG.....IHDR|  
0010  00 00 00 02 00 00 00 02  08 02 00 00 00 fd d4 9a  |.....|  
0020  73 00 00 00 13 49 44 41  54 78 da 63 f8 cf c0 c0  |s....IDATx.c....|  
0030  00 c2 0c ff cd cd 19 00  1b 07 03 6c 23 5e ae 7c  |.....l#^.|  
0040  00 00 00 00 49 45 4e 44  ae 42 60 82                |....IEND.B`.|  
004c
```

Pliki PNG można wyświetlić za pomocą każdego współczesnego programu ;-)

Format JFIF (1/2)

JPEG FILE INTERCHANGE FORMAT



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
000:	FF	D8	FF	E0	00	10	.	J	.	F	.	F	00	01	01	01	00	48
010:	00	48	00	00	FF	DB	00	43	00	01	01	01	01	01	01	01	01	01
020:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
030:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
040:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
050:	01	01	01	01	01	01	01	01	01	FF	DB	00	43	01	01	01	01	01
060:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
070:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
080:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
090:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	FF	C0		
0A0:	00	11	08	00	02	00	06	03	01	22	00	02	11	01	03	11		
0B0:	01	FF	C4	00	15	00	01	01	00	00	00	00	00	00	00	00	00	00
0C0:	00	00	00	00	00	00	00	09	FF	C4	00	19	10	01	00	02		
0D0:	03	00	00	00	00	00	00	00	00	00	00	00	00	00	06	08		
0E0:	38	88	B6	FF	C4	00	15	01	01	01	00	00	00	00	00	00		
0F0:	00	00	00	00	00	00	00	00	07	0A	FF	C4	00	1C	11	00		
100:	01	03	05	00	00	00	00	00	00	00	00	00	00	00	00	00	08	
110:	00	07	B8	09	38	39	76	78	FF	DA	00	0C	03	01	00	02		
120:	11	03	11	00	3F	00	86	F7	E7	1D	A9	16	CA	77	30	D0		
130:	14	F7	41	DC	5A	8E	FB	31	19	26	5D	C4	2A	F4	5C	81		
140:	7B	DB	06	84	A0	75	17	FF	D9									

SEGMENTS	FIELDS	VALUES
START OF IMAGE	marker	FFD8
APPLICATION (DEFAULT HEADER)	marker/length	FFD9/16
	identifier	JFIF 00
	version	1.1
	units	1 (dpi)
	density	72x72
	thumbnail	000
QUANTIZATION TABLE	marker/length	FFD9/57
	destination	0 (Luminance)
table (8x8)		12 (100% quality)
QUANTIZATION TABLE	marker/length	FFD9/57
	destination	1 (Chrominance)
table (8x8)		1 (100% quality)
START OF FRAME	marker/length	FFC0/17
	precision	8
	line no	2
	samples/line	0
	components	3
	id factor table	1 2x1 0 (LumY)
	id factor table	2 2x1 1 (ChromCb)
	id factor table	3 2x1 1 (ChromCr)
HUFFMAN TABLE	marker/length	FFC2/21
	class	0 (DC)
	destination	0
	1 code of 1 bit	00
	1 code of 2 bits	00
HUFFMAN TABLE	marker/length	FFC4/25
	class	0 (AC)
	destination	0
	1 code of 1 bit	00
	2 code of 3 bits	00 00
	3 code of 4 bits	10 00 00
HUFFMAN TABLE	marker/length	FFC4/23
	class	0 (AC)
	destination	0
	1 code of 1 bit	00
	1 code of 2 bits	00
HUFFMAN TABLE	marker/length	FFC4/20
	class	1 (AC)
	destination	1
	1 code of 2 bits	00
	2 code of 3 bits	00 00 00
	3 code of 4 bits	00 10 00 78
START OF SCAN	marker/length	FFDA/12
	components	3
	selector / DC, AC table	1 / 0, 1
		2 / 1, 1
		3 / 1, 1
IMAGE DATA (ENTROPY-CODED SEGMENT)	spectral select.	0...03
	00 00 00 00 00 00 00 00	
	80 7E 7D 00 00 00 00 00 00	
	FF 4DC 5A 8F B3 11 20 00 0C 4	
	2A F 45 C 1 7 0 0 0 0 6 1 0 0 7 5 1 7	
END OF IMAGE	marker	FFD9



ANGE ALBERTINI
http://pics.corkami.com



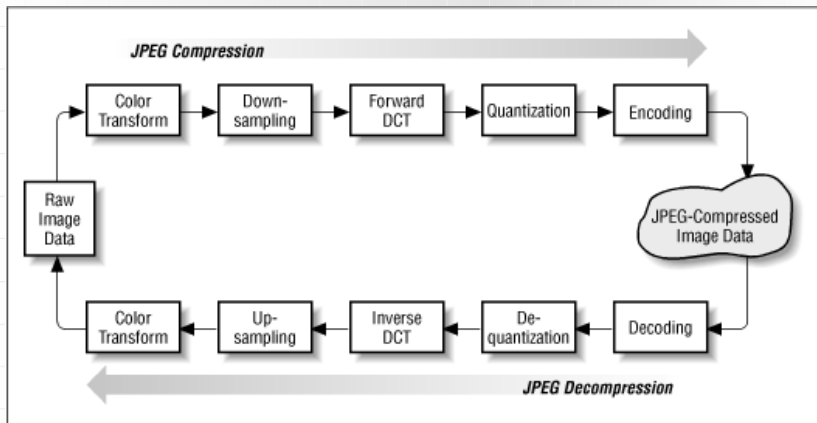
JPEG IS THE ENCODING STANDARD, JFIF IS THE FILE FORMAT

Format JFIF (2/2)

- Pliki te są popularnie nazywane po prostu JPEGami.
- Fachowo jednak należy rozróżnić:
 - ▶ kontener JFIF – przechowuje informacje o osadzonym obrazie,
 - ▶ algorytm JPEG – standard kompresji stratnej danych graficznych.
- Format JFIF ma dość złożoną strukturę, nie będziemy go implementować.
- Pochylimy się natomiast nad kunsztem w ramach idei algorytmu JPEG!
 - ▶ Celem algorytmu JPEG było jak największe zmniejszenie rozmiaru pliku.
 - ▶ Wykorzystuje się w nim słabości ludzkiej percepcji wzrokowej.
 - ▶ Proces kompresji algorytmem JPEG jest procesem stratnym,
 - ▶ utracona zostaje część informacji graficznej (zwykle niezauważalna),
 - ▶ stopień strat jest regulowalnym parametrem algorytmu.

Algorytm JPEG (1/2)

Można znaleźć wiele opisów algorytmu JPEG. Różnią się one drobnymi szczegółami, na przykład na temat tego, jak duże powinno być próbkowanie, co kwantyzować i w jaki sposób należy zakodować dane. Ogólny ich zamysł jest jednak taki sam.



Algorytm JPEG (2/2)

1. Konwersja modelu barw: RGB \rightarrow YCbCr.
2. Przeskalowanie w dół (stratne) macierzy składowych Cb i Cr.
3. Podział obrazu na bloki o rozmiarze 8x8.
4. Wykonanie dyskretnej transformacji cosinusowej na każdym bloku obrazu.
5. Podzielenie każdego bloku obrazu przez macierz kwantyzacji.
6. Zaokrąglenie wartości w każdym bloku do liczb całkowitych.
7. Zwinięcie każdego bloku 8x8 do wiersza 1x64 – algorytm ZigZag.
8. Zakodowanie danych obrazu – m.in. algorytmem Huffmana.¹

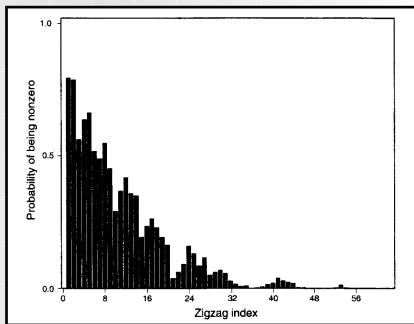
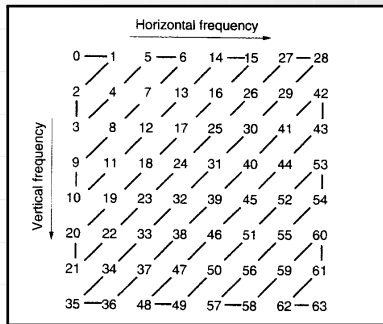
Algorytm jest symetryczny – dekompresja to po prostu odwrócenie procesu, czyli przejście powyższych kroków w odwrotnej kolejności i działaniach.

¹ w ramach zajęć nie będziemy realizować tego kroku zgodnie ze sztuką, a jedynie poglądowo

Algorytm ZigZag

Celem jest takie zwinienie macierzy do wektora, aby na jednym końcu zgrupować większość wartości niezerowych, a na drugim większość wartości zerowych.

Skutkiem tego jest możliwość efektywniejszego zakodowania uzyskanych danych.



Koniec wprowadzenia.

Zadania do wykonania...

Zadania do wykonania (1)

Na ocenę **3.0** należy zaimplementować obsługę formatu PPM.

Wskazówki:

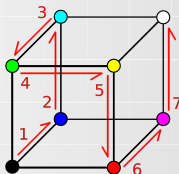
- wystarczy zapisać poprawny obrazek, wypełniony pojedynczym kolorem,
- szablon programu znajduje się w pliku `rozwiązania.ipynb`,
- do zaimplementowania został nagłówek i dane obrazu (macierz RGB),
 - ▶ zmodyfikować fragmenty oznaczone przez `# TODO: implement`,
- proszę zaimplementować wariant P3 (tekstowy) i P6 (binarny),
- spojrzeć i porównać rozmiary plików tworzonych w obu wariantach.

Zadania do wykonania (2)

Na ocenę **3.5** należy wyrysować przestrzeń barw w pliku PPM.

Wskazówki:

- rozbudować poprzedni program, o ile tworzone pliki PPM są poprawne,
- można już ograniczyć się do jednego wybranego wariantu formatu PPM,
- przykładowy wynik został zaprezentowany na dole slajdu,
- na rysunku sześcianu zaznaczono schemat przejść w przestrzeni RGB.



Zadania do wykonania (3)

Na ocenę **4.0** należy zaimplementować tworzenie pliku PNG.

Wskazówki:

- zapisywanym obrazkiem niech jest tęcza z poprzedniego ćwiczenia,
- szablon programu znajduje się w pliku `rozwiązania.ipynb`,
- wystarczy zaimplementować jeden, prosty wariant – bez przezroczystości.
- do skompresowania danych obrazu użyć funkcji `zlib.compress(...)`.
 - ▶ każdy wiersz obrazka poprzedzić najpierw Bajtem zerowym!
- do wygenerowania nagłówka można użyć funkcji `struct.pack(...)`.
- ustawienia w nagłówku to: (uwaga na kolejność!)
 - ▶ wysokość i szerokość w pikselach ($2 \times \text{uint32}$),
 - ▶ głębia, model kolorów itp. – liczby 8, 2, 0, 0 i 0 ($5 \times \text{uint8}$).

Zadania do wykonania (4)

Na ocenę **4.5** należy zaimplementować część algorytmu JPEG.

Wskazówki:

- szablon programu znajduje się w pliku `rozwiązania.ipynb`,
- obrazkiem roboczym niech jest tęcza z poprzedniego ćwiczenia,
- należy zaimplementować kroki 0, 1, 2, 3, 7, 8 i odpowiednie odwrotne.
- w etapie 8: zmierzyć rozmiar (liczbę Bajtów) powstałego obrazu,
- ocenić wpływ etapu 2 na rozmiar i wygląd obrazka:
 - ▶ do sprawdzenia: bez próbkowania, co drugi element i co czwarty,
 - ▶ obserwacje zanotować jako komentarz w osobnej komórce na końcu,
- warunek konieczny: algorytm musi zostać zaimplementowany poprawnie,
 - ▶ obrazek wynikowy musi odpowiadać wejściowemu,
- w etapach tego zadania **można** korzystać z bibliotek pomocniczych!

Zadania do wykonania (5)

Na ocenę **5.0** należy dokończyć implementację algorytmu JPEG.

Wskazówki:

- to zadanie jest rozwinięciem poprzedniego,
- należy zaimplementować tutaj dodatkowo etapy 4, 5 oraz 6 i odwrotne,
- funkcje pomocnicze, które można znaleźć w pliku `rozwiązania.ipynb`:
 - wyznaczanie macierzy kwantyzacji, zależnie od parametru QF ,
 - obliczanie 2-wymiarowej dyskretnej transformaty cosinusowej,
- ocenić jak wybór czynnika QF wpływa na rozmiar i wygląd obrazka,
- w etapach tego zadania **można** korzystać z bibliotek pomocniczych!