Tomasz Kryjak, Piotr Pawlik 9 października 2016

2 — Rozdzielczość obrazu. Interpolacja.

2.1 Cel

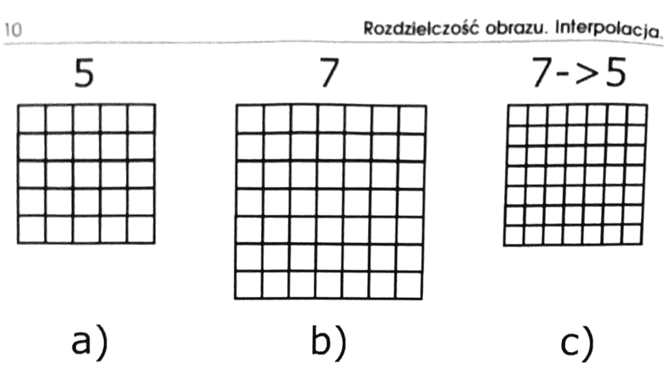
- zapoznanie z pojęciem rozdzielczości przestrzennej (rozmiaru obrazu),
- metody interpolacji najbliższego sąsiada oraz dwuliniowa,
- zapoznanie z pojęciem rozdzielczości dpi (ang. *dots per inch*),
- zapoznanie z pojęciem rozdzielczości poziomów jasności (dla obrazów w skali szarości),
- zadanie domowe: interpolacja dwusześcienna.

2.2 Rozdzielczość przestrzenna

Dyskretna reprezentacja obrazu to zwykle macierz dwu ($N \times M$ – obraz w skali szarości) lub trójwymiarowa ($N \times M\ times3$ – obraz kolorowy). Przez rozdzielczość przestrzenną rozumie się liczbę pikseli z których składa się obraz. Przykładowo rozdzielczość VGA to 640×480, a tzw. Full HD to 1920×1080. Rozdzielczość obrazu można modyfikować (zwiększać/zmniejszać), co nazywa się skalowaniem obrazu. Warto wiedzieć, że zwiększenie rozdzielczości obrazu nie zwiększa ilości informacji, a jedynie liczbę pikseli. Ponadto tego typu operacja zawsze wprowadza pewne zniekształcenia, nawet przy zmniejszaniu rozmiaru.

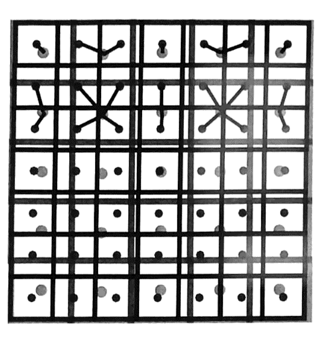
W ramach niniejszego ćwiczenia zapoznamy się z metodami interpolacji, które są podstawą takich operacji jak: przybliżanie (zoom), zmiana rozdzielczości, rotacja obrazu, czy też korekcje geometryczne. Jako przykład posłuży nam zmiana rozdzielczości, czyli inaczej mówiąc przepróbkowanie obrazu. Dla przypomnienia – interpolacja to wykorzystanie znanych danych (wartości dla tzw. punktów węzłowych) do określania wartości w nieznanych lokalizacjach.

Zacznijmy od prostego przykładu. Mamy obraz o rozdzielczości 500×500 pikseli, a chcemy go powiększyć do 750×750 pikseli – tj. o współczynnik 1,5. Wyobraźmy sobie zatem, że dysponujemy siatką 750×750 o takim samym rozmiarze pojedynczego piksela. Następnie siatkę tą „ścieśniamy” tak aby miała rozmiar 500×500. W rezultacie otrzymana siatka będzie miała mniejszy rozmiar pojedynczego piksela niż obraz oryginalny. Schematycznie przedstawiono to na rysunku 2.1.



Rysunek 2.1: Przykład interpolacji: a) – obraz 5×5, b) – obraz 7×7, c) – obraz 7×7 zmniejszony do 5×5.

Chcemy teraz poszczególnym elementom nowej siatki przyporządkować piksele z obrazu wejściowego. Jedną z możliwości jest poszukanie “najbliższego” piksela w oryginalnym obrazie i wzięcie jego wartości. Przykład takiego postępowania zaprezentowano na rysunku 2.2. Po zrealizowaniu powyższego kroku dla całego obrazu wykonujemy “rozciągniecie” do rozdzielczości 750×750. W ten sposób uzyskujemy finalny efekt zmiany rozdzielczości.



Rysunek 2.2: Przykład poszukiwania najbliższego sąsiada. Obraz czarny – 7×7 zmniejszony do 5×5, obraz szary 5×5, kreski w pierwszych trzech rzędach ilustrują najbliższego sąsiada.

2.2.1 Interpolacja metodą najbliższego sąsiada

Takie postępowanie określa się mianem **interpolacji metodą najbliższego sąsiada** (ang. *nearest neighbour interpolation*). W ramach pierwszego etapu ćwiczenia zaimplementujemy to podejście.

1. Otwórz program Matlab. Ustal ścieżkę *Current Directory* na swój własny katalog na dysku D. Utwórz nowy m-plik (*New Script*) lub (*New->Script*). W skrypcie umieść polecenia `clearvars; close all; clc;`

2. Wczytaj obraz *parrot.bmp* (dostępny w archiwum do bieżącego ćwiczenia). Wyświetl go.
3. Zaczniemy teraz tworzyć szablon przekształcenia rozdzielczości obrazu. Na początek definiujemy dwa współczynniki skalowania. Przykładowo: `xReScale` i `yReScale`. Obu nadajemy na początek wartość *1,5*.
4. Na podstawie współczynników możemy określić rozdzielczość nowego obrazu. Pobieramy rozdzielczość obrazu wejściowego (`[YY,XX] = size(I);`) i wyznaczmy nową wg. wzoru: $nYY = YY * yReScale$. Uwaga. Wynik operacji należy zaokrąglić (polecenie `round` lub `floor`). Dla współrzędnej *XX* postępujemy analogicznie. Dysponując nową rozdzielczością możemy stworzyć nowy obraz: `nI = uint8(zeros(nYY,nXX));`.
5. Operację skalowania będziemy przeprowadzać w pętli `for` po obrazie (w dwóch pętlach). Ustalamy ich zakres na od 0 do *nYY-1* (*nXX-1*). Uwaga. Wariant z liczeniem od 1 do *nYY*(*nXX*) również jest dopuszczalny, ale wymaga implementacji nieco innego sposobu obsługi potencjalnego wykroczenia poza zakres.
6. W tym kroku implementujemy metodę najbliższego sąsiada. Zaczynamy od wyznaczenia aktualnie wyliczanej współrzędnej nowego obrazu poddanego dopasowaniu do rozdzielczości obrazu wejściowego (por. rysunek 2.1). Przykład: `i = ii*xStep`, gdzie: `ii` – współrzędna w nowym obrazie, `xStep` – krok określony zależnością XX/nXX, `i` – obliczona współrzędna w obrazie wejściowym. Analogicznie postępujemy dla drugiej współrzędnej. Uwaga. Obliczanie kroków realizujemy poza pętlą. Aby obliczone współrzędne były „najbliższym sąsiadem” należy zaokrąglić uzyskane wartości (polecenie `round`). Wykorzystując powyższe obliczenia możemy określić nową wartość piksela jako: `nI(jj+1,ii+1) = I(j+1,i+1);`. Uwaga. “+1” wynika przyjętej konwencji indeksowania od “0” (dla przypomnienia w pakiecie Matlab indeksowanie jest „od 1”).
7. Do poprawnego działania konieczne jest również dodanie obsługi przypadku przekroczenia zakresu. Jeśli któraś ze współrzędnych wykracza poza wartości $YY-1$, $XX-1$ to trzeba jej przypisać te wartości.
8. Przetestuj działanie metody na obrazach *parrot.bmp*, *chessboard.bmp* i *clock.bmp*. Wypróbuj różne skale (mniejsze od 1, większe od 1, odmiennie dla współrzędnej X i Y). Uwaga. Doświadczenie uczy, że błędy w implementacji ujawniają się przy „asymetriach” – tj. dla obrazu *clock.bmp* i różnych skal dla osi.

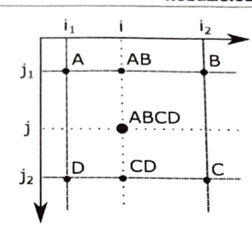
Wykonana metoda jest bardzo prosta i szybka, ale wprowadza pewne niepożądane artefakty, w szczególnie źle odwzorowane są linie proste. Z drugiej strony sprawdza się w pewnych nietypowych przypadkach. Zostanie to zademonstrowane w dalszej części ćwiczenia.

2.2.2 Interpolacja dwuliniowa

W praktyce, lepszym rozwiązaniem zwykle okazuje tzw. **interpolacja dwuliniowa** (ang. *bilinear interpolation*). Wykorzystuje ona informację o czterech najbliższych sąsiadach do określenia nowej wartości piksela. Jeśli przez *(i,j)* oznaczymy współrzędne poszukiwanego piksela, a przez $I(i,j)$ jego jasność (składową w odcieniach szarości) to jego wartość można obliczyć wykorzystując równanie:

$$I(i,j) = a \cdot i + b \cdot j + c \cdot i \cdot j + d \tag{2.1}$$

gdzie: współczynniki a,b,c,d można wyliczyć na podstawie czterech najbliższych sąsiadów.



Rysunek 2.3: Ilustracja interpolacji dwuliniowej. Źródło: opracowanie własne na podstawie Wikipedia.

Prześledźmy to na przykładzie z rysunku 2.3. Niech współrzędne poszczególnych punktów to $A = (j_1, i_1)$, $B = (j_1, i_2)$, $C = (j_2, i_2)$ oraz $D = (j_2, i_1)$. W pierwszej kolejności dokonujemy interpolacji wartości w punktach AB i CD – czyli poziomo. Wychodząc od równania prostej otrzymujemy:

$$f(AB) \approx \frac{i_2 - i}{i_2 - i_1} f(A) + \frac{i - i_1}{i_2 - i_1} f(B) \tag{2.2}$$

$$f(CD) \approx \frac{i_2 - i}{i_2 - i_1} f(D) + \frac{i - i_1}{i_2 - i_1} f(C) \tag{2.3}$$

Następnie wykonujemy analogiczną interpolację w pionie:

$$f(ABCD) \approx \frac{j_2 - j}{j_2 - j_1} f(AB) + \frac{j - j_1}{j_2 - j_1} f(CD) \tag{2.4}$$

Łącząc powyższe równania otrzymujemy:

$$f(ABCD) \approx \frac{1}{(i_2 - i_1)(j_2 - j_1)} (f(A)(i_2 - i)(j_2 - y) + f(B)(i - i_1)(j_2 - j) \\ + f(C)(i - i_1)(j - j_1) + f(D)(i_2 - i)(j - j_1)) \tag{2.5}$$

gdzie zapis $f(X)$ oznacza wartość piksela w punkcie X.

Rozważania można uprościć przyjmując, że narożniki rozpatrywanego kwadratu mają następujące współrzędne: $A = (0,0)$, $B = (0,1)$, $C = (1,1)$ oraz $D = (1,0)$. Wtedy równanie (2.5) można zapisać:

$$f(ABCD) \approx f(A)(1-i)(1-j) + f(B)i(1-j) + f(C)ij + f(D)(1-i)j \tag{2.6}$$

lub macierzowo:

$$f(ABCD) \approx \begin{bmatrix} 1-i & i \end{bmatrix} \begin{bmatrix} f(A) & f(D) \\ f(B) & f(C) \end{bmatrix} \begin{bmatrix} 1-j \\ j \end{bmatrix} \tag{2.7}$$

Uwaga. Wbrew nazwie interpolacja dwuliniowa nie jest operacją liniową. W złożeniu dwóch operacji liniowych pojawia się człon *xy*.

1. Wykorzystując równanie (2.7) zaimplementuj interpolację dwuliniową. Uwaga. Proszę nie „nadpisać” kodu do interpolacji metodą najbliższego sąsiada, a skopiować go do nowego m-pliku jako „szablon”.
2. Na początku trzeba obliczyć współrzędne punktu A tj. $(0,0)$ – (j_1, i_1). Należy do tego wykorzystać instrukcję `floor`. Przykład: $i_1 = ii \times xStep$. Na tej podstawie możemy wyznaczyć współrzędne punktów B,C,D (odpowiednio dodając 1). Ponadto musimy bieżącą współrzędną (i,j) przekształcić do przedziału $[0;1]$. Podpowiedź. Wykorzystaj współrzędne punktu A – (j_1, i_1). Dodatkowo należy wprowadzić zabezpieczanie przed przekroczeniem górnego zakresu – podobnie jak w poprzedniej metodzie.
3. Wykorzystując wyznaczone współrzędne, należy pobrać wartości jasności w punktach A,B,C,D, tj. $f(A), f(B), f(C), f(D)$, podstawić do równania (2.7) i wykonać interpolację. Uwaga. Aby mnożenie macierzowe zrealizowane zostało poprawnie należy obraz wejściowy przekształcić z typu `uint8` na typ `double`. Natomiast dla celów wizualizacji należy dokonać operacji odwrotnej: `imshow(uint8(I));`. Proszę sprawdzić działanie metody dla kilku różnych skal.

Następny „poziom wtajemniczenia” to **interpolacja dwusześcienna** (ang. *bicubic interpolation*). W trakcie jej obliczania wykorzystywane jest 16 pikseli z otoczenia. Dana jest ona wzorem:

$$I(i,j) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \tag{2.8}$$

Jej implementacja stanowi zadanie domowe do bieżącego ćwiczenia – rozdział 2.5.

2.2.3 Porównanie interpolacji

Omówione w ramach ćwiczenia metody interpolacji dostępne są w pakiecie Matlab – funkcja `imresize`.

1. Wykorzystując funkcję `imresize` porównaj trzy metody interpolacji: najbliższego sąsiada, dwuliniową, dwusześcienną. Wykorzystaj trzy obrazy: *parrot.bmp*, *chessboard.bmp* i *clock.bmp*. Spróbuj skalowania „do wielokrotności” (np. 100 na 200) i nie (np. 100 na 130).
2. Zastanów się nad przypadkiem szachownicy – dlaczego tutaj metoda najbliższego sąsiada ma pewną przewagę. Przeanalizuj też wyniki dla innych obrazów testowych.

2.3 Rozdzielczość (dpi)

Omówioną wcześniej rozdzielczość przestrzenną (rozmiar) należy utożsamiać z rozmiarem macierzy w której zapisany jest obraz. W tym ujęciu rozmiar pojedynczego piksela nie ma specjalnego znaczenia. Problem pojawia się, kiedy obraz trzeba wyświetlić lub wydrukować. Wtedy pojedynczy piksel staje się „obiektem fizycznym” i musi mieć swój rozmiar (wysokość/szerokość/powierzchnię).

Parametr dpi (ang. *dots per inch*) określa liczbę kropek (pikseli), która mieści się na jednym calu (25,4 mm) długości/szerokości. Dopiero kombinacja rozmiaru i rozdzielczości określa nam rzeczywisty rozmiar obrazu jaki uzyskamy na wydruku.

Dpi staje się istotne w przypadku drukowania, gdyż wyświetlanie na monitorze odbywa się zazwyczaj 1 piksel obrazka = 1 piksel na monitorze (w przypadku maksymalnej rozdzielczości wspieranej przez monitor), ew. następuje automatyczne skalowanie. Wpływ rozdzielczości można zademonstrować w opisany poniżej sposób.

1. Wczytaj obraz *lena.bmp*. Ma on rozmiar 512×512. Wykorzystując funkcję `imresize` stwórz obrazy o rozmiarach 256×256, 128×128, 64×64 – metoda interpolacji jest w tym wypadku mniej istotna.
2. Wyświetlając obrazy “wymusimy” zachowanie rozmiaru na ekranie 512×512. W tym celu wykorzystamy parametr funkcji `imshow – 'InitialMagnification'`. Ustawiamy ją odpowiednio na 200, 400, 800. Zapoznaj się z uzyskanymi wynikami (wyświetl obrazy oryginalny i po przeskalowaniach).

2.4 Liczba poziomów jasności

Dla obrazów w skali szarości pojedynczy piksel zapisuje się zazwyczaj na 8 bitach, co daje 256 rozróżnialnych poziomów szarości. Dla większości zastosowań wartość ta jest wystarczająca. Oko ludzkie nie potrafi rozróżnić wszystkich 256 poziomów jasności (jest za mało czułe). Zazwyczaj człowiek rozróżnia 20-30 poziomów szarości (to jakie dokładnie rozróżnia, zależy od konkretnego oświetlenia sceny i cech osobniczych).

1. Wczytaj obraz *lena.bmp*. Zmniejsz jego rozmiar do 128×128 (łatwiejsze wyświetlanie). Wykorzystując funkcję `imadjust` zmień liczbę poziomów szarości z 0 – 255 na:
   - 0-31,
   - 0-15,
   - 0-7,
   - 0-3,
   - 0-1 (binaryzacja).

   Uwaga: funkcja `imadjust` jako parametr przyjmuje wartości z zakresu 0–1. Należy dokonać odpowiedniego przeskalowania.
2. Rezultaty wyświetl na wspólnym rysunku (funkcja `subplot`). Uwaga: Aby poprawnie wyświetlić obrazek należy wykorzystać następującą postać funkcji `imshow(lena,[]);`. Czy rezultaty eksperymentu pasują do teorii o rozpoznawaniu przez człowieka ograniczonego zakresu poziomów jasności? Wizualne porównanie których obrazów o tym świadczy ?
3. **Wyniki zaprezentuj prowadzącemu.**

2.5 Zadanie domowe

Interpolacja dwusześcienna, to podobnie jak w przypadku interpolacji dwuliniowej, rozszerzenie idei interpolacji jednowymiarowej na dwuwymiarową siatkę. W trakcie jej obliczania wykorzystywane jest 16 pikseli z otoczenia (dla dwuliniowej 4). Skutkuje to zwykle lepszymi wynikami – obraz wyjściowy jest bardziej gładki i z mniejszą liczbą artefaktów. Ceną jest znaczny wzrost złożoności obliczeniowej.

Interpolacja dana jest wzorem:

$$I(i,j) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \tag{2.9}$$

Zadanie sprowadza się zatem do wyznaczenia 16 współczynników a_{ij}. W tym celu wykorzystuje się, oprócz wartość w puntach A (0,0), B (1 0), C (1,1), D (0,1) – por. rysunek 2.3, także pochodne cząstkowe A_x, A_y, A_{xy}. Pozwala to rozwiązać układ 16-tu równań.

Jeśli zgrupujemy parametry a_{ij}:

$$a = [a_{00}\ a_{10}\ a_{20}\ a_{30}\ a_{01}\ a_{11}\ a_{21}\ a_{31}\ a_{02}\ a_{12}\ a_{22}\ a_{32}\ a_{03}\ a_{13}\ a_{23}\ a_{33}] \tag{2.10}$$

i przyjmiemy:

$$x = [A\ B\ D\ C\ A_x\ B_x\ D_x\ C_x\ A_y\ B_y\ D_y\ C_y\ A_{xy}\ B_{xy}\ D_{xy}\ C_{xy}]^T \tag{2.11}$$

To zagadnienie można opisać w postaci równania liniowego:

$$Aa = x \tag{2.12}$$

Gdzie macierz A^{-1} dana jest wzorem:

$$A^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 \\
-3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 \\
9 & -9 & -9 & 9 & 6 & 3 & -6 & -3 & 6 & -6 & 3 & -3 & 4 & 2 & 2 & 1 \\
-6 & 6 & 6 & -6 & -3 & -3 & 3 & 3 & -4 & 4 & -2 & 2 & -2 & -2 & -1 & -1 \\
2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
-6 & 6 & 6 & -6 & -4 & -2 & 4 & 2 & -3 & 3 & -3 & 3 & -2 & -1 & -2 & -1 \\
4 & -4 & -4 & 4 & 2 & 2 & -2 & -2 & 2 & -2 & 2 & -2 & 1 & 1 & 1 & 1
\end{bmatrix} \tag{2.13}$$

Potrzebne w rozważaniach pochodne cząstkowe obliczane są wg. następującego przybliżenia (przykład dla punktu A):

$$A_x = \frac{I(i+1,j) - I(i-1,j)}{2} \tag{2.14}$$

$$A_y = \frac{I(i,j+1) - I(i,j-1)}{2} \tag{2.15}$$

$$A_xy = \frac{I(i+1,j+1) - I(i-1,j) - I(i,j-1) + I(i,j)}{4} \tag{2.16}$$

Zadanie: Wykorzystując podane informacje zaimplementuj interpolację dwusześcienną.

Uwagi:

- macierz A^{-1} dostępna jest w pliku *a1.mat*
- trzeba się zastanowić nad potencjalnym wykraczaniem poza zakres obrazka.

Ponadto dokonaj porównania liczby operacji arytmetycznych i dostępów do pamięci koniecznych przy realizacji obu metod interpolacji: dwuliniowej i dwusześciennej.