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.

### 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.

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 dwuszcścienna** (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, dwuszcścienną. 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 $512 \times 512$. Wykorzystując funkcję `imresize` stwórz obrazy o rozmiarach $256 \times 256$, $128 \times 128$, $64 \times 64$ – metoda interpolacji jest w tym wypadku mniej istotna.
2. Wyświetlając obrazy "wymusimy" zachowanie rozmiaru na ekranie $512 \times 512$. 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 $128 \times 128$ (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 $a_{ij}$:

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