Laboratorium 2

Biblioteki: numpy, matplotlib, scikit-image

Słowa kluczowe: Przekształcenia afiniczne, interpolacja.

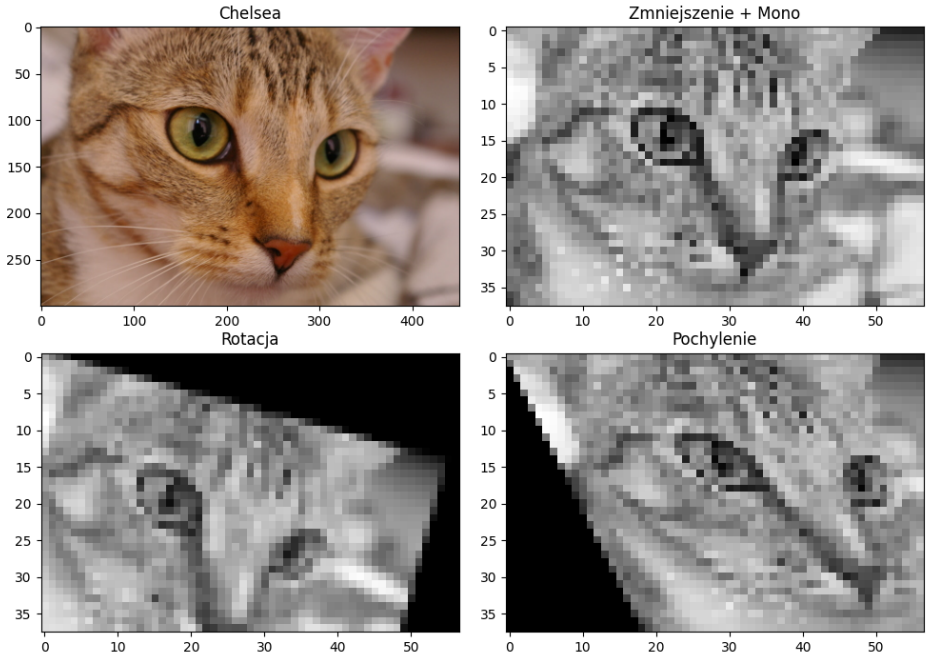
Zadanie 1:

- Zadeklaruj wykres składający się z 2 kolumn i 3 wierszy, o wymiarach 10×7 cali.
- Wczytaj obraz `chelsea` z modułu `data` biblioteki `scikit-image`. Pokaż ten obraz w pierwszej komórce wykresu.
- Oryginalny sygnał jest trójwymiarowy — przedstawia obraz barwny. Przekształć go w obraz monochromatyczny **uśredniając wszystkie kanały barwne** przy życiu metody `np.mean`.
- Zredukuj jego rozmiar, zmniejszając każdy wymiar przestrzenny ośmiokrotnie. Efekt przedstaw w kolejnej komórce wykresu używając colormapy *binary_r*.
    - Najprościej można zredukować wielkość obrazu wykorzystując odpowiednią adresację. Np. `img[::2]` da w wyniku co drugi element z pierwszej osi tablicy `img`.
- Wykonaj dwie różne transformacje afiniczne obrazu:
    - rotacja o kąt 15 stopni.
    - pochylenie w osi X o 0.5.

  Wykorzystaj w tym celu gotową funkcję `warp` z biblioteki `scikit-image`. Konieczna będzie definicja obiektu klasy `AffineTransform` . Zdefiniuj go, przekazując **macierz transformacji** jako parametr.

  Jak zdefiniować macierz transformacji?
- Wyświetlić efekty transformacji w drugim wierszu wykresu.

Efekt zadania 1:



Zadanie 2:

W tym zadaniu *manualnie* wykonamy transformację afiniczną, korzystając z operacji macierzowych. W tym zadaniu wykorzystamy zmniejszony obraz monochromatyczny z zadania 1.

- Zadeklaruj wykres składający się z 2 kolumn i jednego wiersza o rozmiarze 9×6 cali. Ustaw wspólne współrzędne osi x i y (parametry `sharex`, `sharey`)
- Utwórz tablicę zawierającą współrzędne wszystkich pikseli obrazu w odpowiedniej kolejności:

  ```
  [[0, 0], [0, 1], [0, 2], [0, 3]...]
  ```

  Możesz zrobić to wykorzystując zagnieżdżone pętle for (wersja bardziej intuicyjna) lub polecenie `meshgrid` z biblioteki numpy (wersja bardziej elegancka).

  Tablica powinna mieć kształt (2166, 2).
- W pierwszej komórce wykresu narysuj punkty (funkcja `scatter`) w wyznaczonych koordynatach, następnie nadaj im kolor w zależności od intensywności obrazu. Wybierz colormapę *binary_r*.
  - Podpowiedź: parametr `c` funkcji scatter może przyjąć wektor wartości (czyli tablicę jednowymiarową) mówiący o barwie kolejno rysowanych punktów.
  - Podpowiedź 2: aby z obrazu utworzyć wektor, wystarczy go *spłaszczyć* wykorzystując funkcję `flatten` lub `reshape`.

- Do każdego elementu tablicy w drugiej osi doklej wartość 1, tak aby wartości wyglądały następująco:

  ```
  [[0, 0, 1], [0, 1, 1], [0, 2, 1], [0, 3, 1]...]
  ```

  Można to zrobić między innymi za pomocą `np.concatenate` lub `np.column_stack`.

  Po tej operacji tablica powinna mieć kształt (2166, 3).

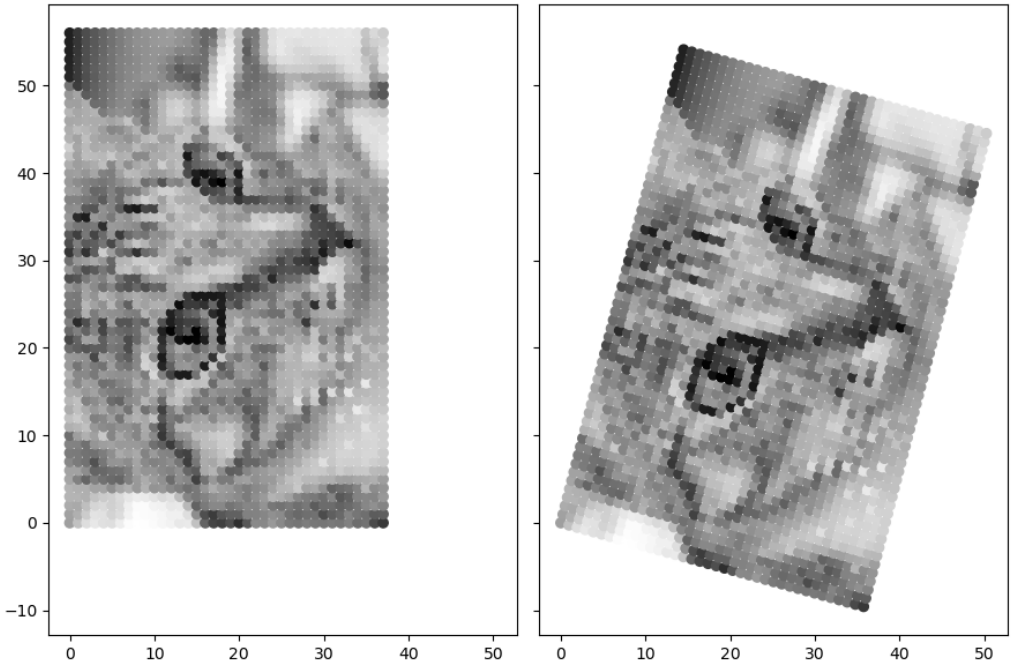
- Zdefiniuj macierz transformacji dla operacji rotacji o 15 stopni (będzie analogiczna do tej z zadania 1.)
- Wykonaj transformację afiniczną wykorzystując operację macierzową:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  gdzie A jest macierzą transformacji, a tablicę współrzędnych masz już z poprzedniego punktu tego zadania. Pamiętaj, że $\times$ opisuje mnożenie macierzowe, czyli w *pythonie* operator `@`. Otrzymana tablica będzie zawierać nowe współrzędne punktów: *x'* i *y'*.

- Zaprezentuj nowe współrzędne w drugiej kolumnie wykresu, przypisując im wartości intensywności z oryginalnego obrazu.

Efekt zadania 2:



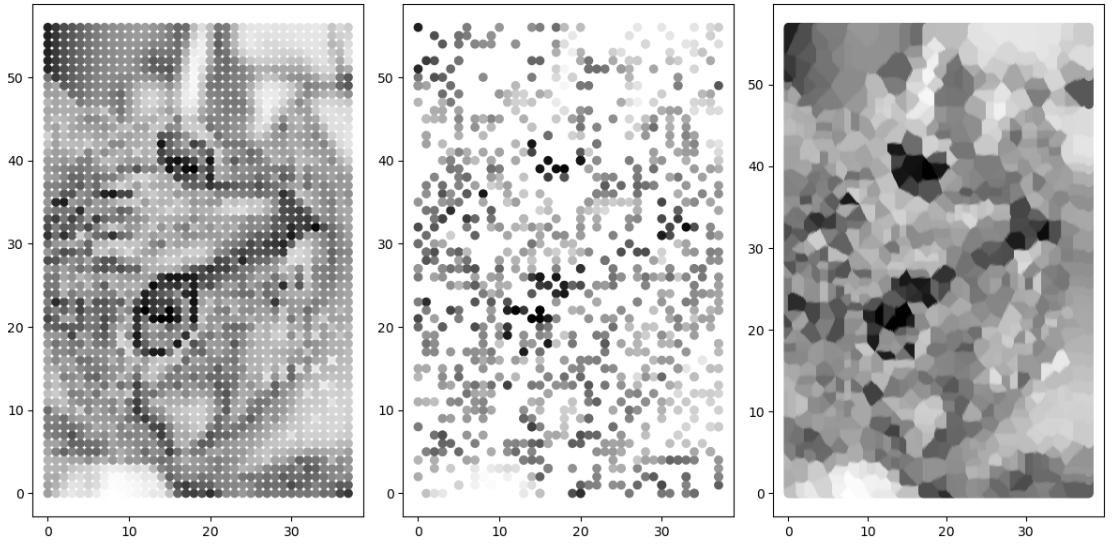
Zadanie 3:

W tym zadaniu zmierzymy się z interpolacją (ale tą najprostsza — bo *najbliższego sąsiada*). Ponownie wykorzystamy zmniejszony i monochromatyczny obraz *chelsea*.

- Zadeklaruj kolejny wykres, zawierający jeden wiersz i trzy kolumny.
- Przydadzą się wyznaczone współrzędne punktów (ale tym razem *jedynki* w trzeciej kolumnie będą już zbędne). Przedstaw współrzędne i intensywności pikseli w pierwszej komórce wykresu, tak jak w zadaniu 2.
- Losowo wybierz 1000 punktów z pierwotnego obrazu. W tym celu wykorzystaj losowanie ich **indeksów** za pomocą funkcji `np.random.choice`. Przedstaw *wybrakowany* sygnał w drugiej komórce wykresu.
- Przeprowadź procedurę **interpolacji najbliższego sąsiada** w dwóch wymiarach, zwiększajac rozdzielczość obrazu do 300 × 400 pikseli. Warto skorzystać z funkcji `scipy.spatial.distance.cdist` do obliczenia odległości.

  Przypomnienie: w interpolacji najbliższego sąsiada wartość punktu w danej współrzędnej jest **kopią wartości** z najbliżej leżącego punktu do szukanego.
- Przedstaw sygnał po interpolacji w ostatniej kolumnie wykresu, również używając funkcji `scatterplot`.

Efekt zadania 3:



Co warto zapamiętać:

- Wczytywanie obrazu z biblioteki *scikit-image*.

- Uśrednienie kanałów barwnych obrazu.
- Redukcja rozmiaru obrazu przez *podpróbkowanie*.
- Określenie współrzędnych pikseli.
- Łączenie tabeli (`np.column_stack` lub `np.concatenate`)
- Losowanie wartości za pomocą `np.random.choice`.