

# Skalowanie Obrazów

Klaudia Lota

Numer indeksu: 272576

Grupa: Wtorek TNP 17:05

28 listopada 2023

## 1 Interpolacja funkcji

### 1.1 Cel ćwiczenia

Celem tego ćwiczenia jest zastosowanie jedno-wymiarowej konwolucji do interpolacji funkcji, takich jak  $\sin(x)$ , przy użyciu różnych jąder splotu. Analizowana jest jakość interpolacji za pomocą kryterium MSE, a także badany jest wpływ liczby punktów oraz ich rozmieszczenia na efektywność interpolacji. Dodatkowo, porównywane są interpolacje o dużej liczbie punktów z sekwencją interpolacji tworzącą mniej punktów, aby zrozumieć wpływ iteracyjnego procesu na jakość interpolacji.

### 1.2 Interpolacja 3 funkcji

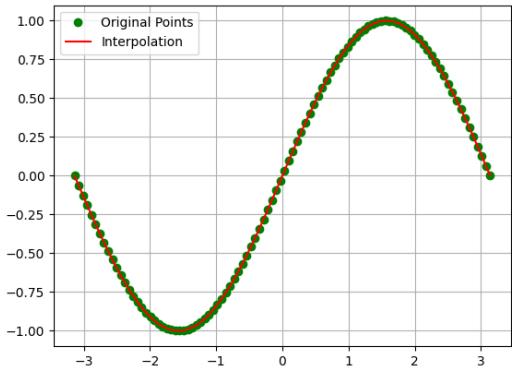
W celu przeprowadzenia interpolacji na trzech funkcjach sinusoidalnych, konieczne było wcześniejsze zdefiniowanie ich w programie zgodnie z poniższym kodem:

```
def sin(x):
    return np.sin(x)

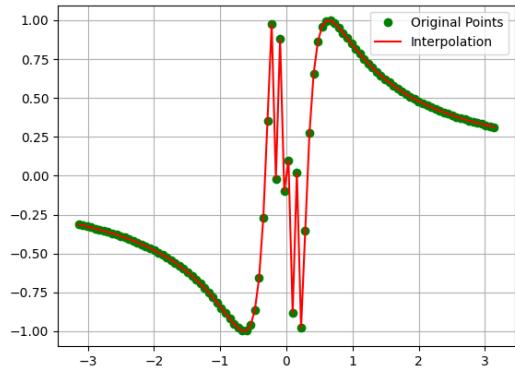
def sinx(x):
    return np.sin(1/x)

def sin8x(x):
    return np.sign(np.sin(8*x))
```

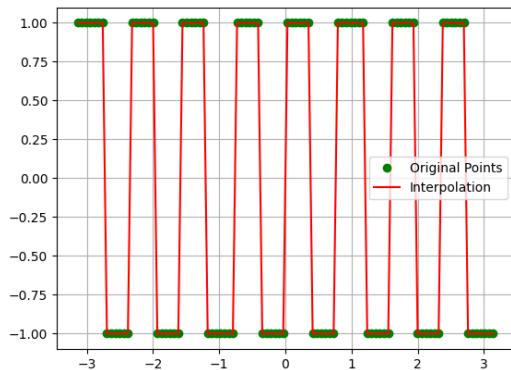
W kolejnym etapie przeprowadzono interpolację funkcji przy użyciu metody liniowej. Po stworzeniu funkcji interpolacyjnej na podstawie dostępnych danych  $x$  i  $y$ , dokonano zwiększenia liczby punktów na osi  $x$ . Następnie wykorzystano tę funkcję do uzyskania nowych wartości ( $y\_int$ ), które zostały porównane z rzeczywistą funkcją sinusoidalną. W rezultacie uzyskane wyniki zostały zwizualizowane i przedstawione poniżej:



Rysunek 1: Interpolacja funkcji  $f_1(x) = \sin(x)$



Rysunek 2: Interpolacja funkcji  $f_2(x) = \sin(x^{-1})$



Rysunek 3: Interpolacja funkcji  $f_3(x) = \text{sgn}(\sin(8x))$

W ramach analizy wpływu różnych jąder konwolucji przeprowadzono implementację czterech funkcji:  $h_1(x)$ ,  $h_2(x)$ ,  $h_3(x)$ , oraz  $h_4(x)$ , zgodnie z poniższym kodem:

```
def h1(x):
    if(x>=0 and x<1):
        return 1
    else:
        return 0

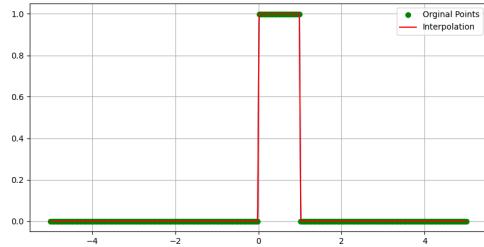
def h2(x):
    if(x>= - 1 / 2 and x < 1 / 2 ):
        return 1
    else:
        return 0

def h3(x):
    if( - 1 <= x <= 1):
        return 1 - abs(x)
    else:
        return 0

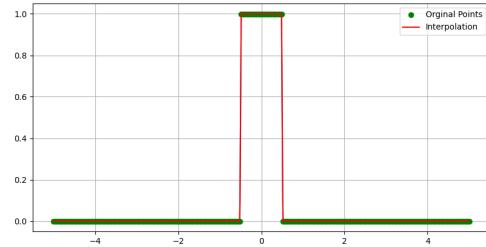
def h4(x):
    return (np.sin(x) / x)
```

Następnie utworzono wektor  $x$  zawierający 300 równomiernie rozmiieszczonych punktów w zakresie

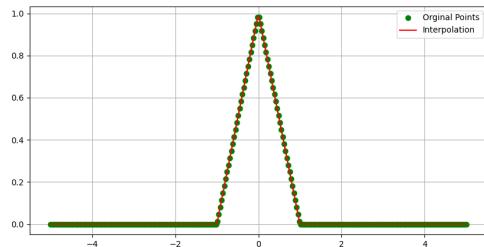
od -5 do 5. Dla każdego punktu  $x_0$  na osi  $x$ , obliczono wartości funkcji konwolucji  $k_1(x_0)$ ,  $k_2(x_0)$ ,  $k_3(x_0)$ , oraz  $k_4(x_0)$ . Następnie przeprowadzono interpolację otrzymanych wyników. Efekty analizy zostały przedstawione na poniższych wykresach:



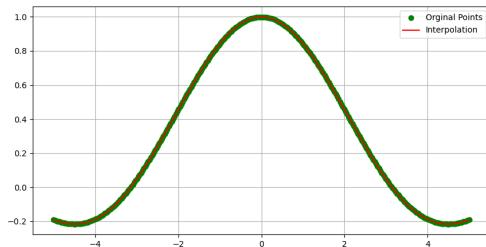
Rysunek 4: Interpolacja funkcji z jądrem  $h_1(x)$



Rysunek 5: Interpolacja funkcji z jądrem  $h_2(x)$



Rysunek 6: Interpolacja funkcji z jądrem  $h_3(x)$



Rysunek 7: Interpolacja funkcji z jądrem  $h_4(x)$

W przypadku interpolacji funkcji sinusoidalnych, metoda liniowej interpolacji okazała się efektywnym narzędziem do przybliżania funkcji takich jak  $f_1(x) = \sin(x)$ ,  $f_2(x) = \sin(x^{-1})$  i  $f_3(x) = \text{sgn}(\sin(8x))$ . Zwiększenie liczby punktów na osi  $x$  pozwoliło uzyskać bardziej dokładne przybliżenie pierwotnych funkcji. Wykresy interpolacji doskonale ilustrują, jak nowe wartości ( $y_{\text{int}}$ ) lepiej odzwierciedlają rzeczywiste funkcje sinusoidalne.

W kontekście różnych jąder konwolucji, zaimplementowane funkcje  $h_1(x)$ ,  $h_2(x)$ ,  $h_3(x)$  i  $h_4(x)$  wprowadziły różnice w procesie interpolacji. Każde z jąder miało swoje specyficzne efekty, takie jak wygładzanie danych, odsiewanie wartości czy nawet wprowadzanie oscylacji. Widać było istotność wyboru odpowiedniego jądra konwolucji, które miało kluczowy wpływ na ostateczne wyniki interpolacji.

Ogólnie rzecz biorąc, analiza ta podkreśla znaczenie interpolacji w przybliżaniu funkcji na podstawie ograniczonej liczby punktów. Dodatkowo, wybór odpowiedniego jądra konwolucji w procesie interpolacji jest kluczowy dla uzyskania precyzyjnych wyników. Skuteczność tych technik może mieć istotne znaczenie w analizie danych funkcji, zwłaszcza w kontekście przetwarzania sygnałów i ekstrapolacji informacji z ograniczonej liczby punktów pomiarowych.

### 1.3 Badanie jakości interpolacji za pomocą kryterium MSE

W celu wykonania tej części zadania należało dopisać do wcześniejszego kodu:

```
print(f"mse: {metrics.mean_squared_error(y_pred=y_int, y_true=y_true):.8f}")
```

co pozwalało obliczyć błąd średniokwadratowy (MSE) między przewidywanymi ( $y_{\text{int}}$ ) a rzeczywistymi ( $y_{\text{true}}$ ) wartościami. MSE to miara jakości modelu, wyrażająca średnią kwadratów różnic między przewidywaniami a rzeczywistymi wartościami obliczana wzorem:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Wynik MSE z dokładnością do 8 miejsc po przecinku został wyświetlony na ekranie co przedstawiono w poniższej tabeli:

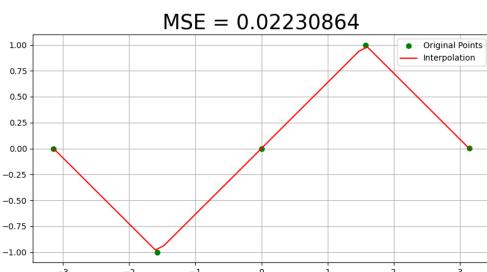
Tabela 1: Wyniki obliczeń MSE dla poszczególnych funkcji

Funkcje	Wartość MSE
$\sin(x)$	0.000 000 07
$\sin(x^{-1})$	0.126 434 91
$\text{sign}(\sin(8x))$	1.464 938 99

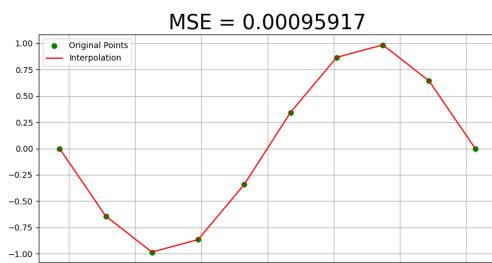
### 1.4 Badanie wpływu liczby oraz rozmieszczenia punktów na jakość interpolacji

W ramach badania wpływu ilości punktów na wskaźnik Mean Squared Error (MSE) w kontekście interpolacji funkcji sinusoidalnej, zastosowano specjalnie przygotowany kod, który został zaprojektowany w celu iteracyjnego obliczania wartości MSE dla różnych ilości punktów, co pozwala na analizę, jak jakość interpolacji zmienia się w zależności od liczby punktów kontrolnych.

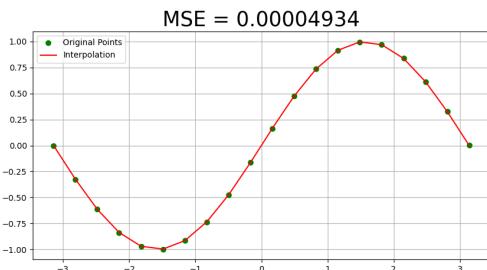
Wyniki tego badania zostały przejrzyście przedstawione na poniższych wykresach generowanych przez kod. Każdy wykres odpowiada konkretnej ilości punktów, a na osiach zaznaczone są zarówno oryginalne punkty, jak i krzywa interpolacyjna. Dodatkowo, wskaźnik MSE dla każdej ilości punktów został wypisany nad wykresem, umożliwiając pełniejszą analizę jakości interpolacji.



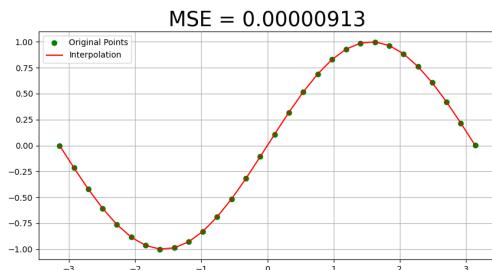
Rysunek 8: Wykres z 5 punktami kontrolnymi



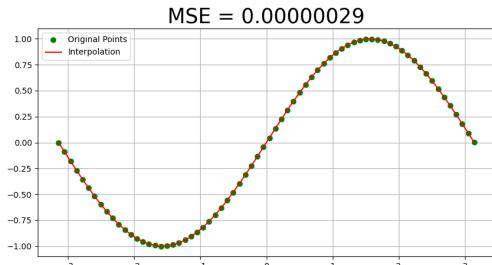
Rysunek 9: Wykres z 10 punktami kontrolnymi



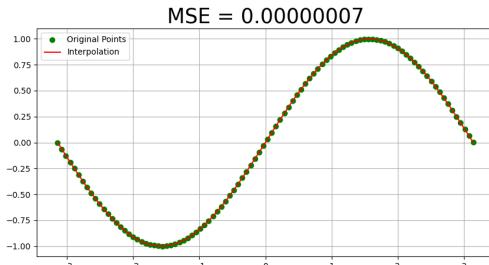
Rysunek 10: Wykres z 20 punktami kontrolnymi



Rysunek 11: Wykres z 30 punktami kontrolnymi

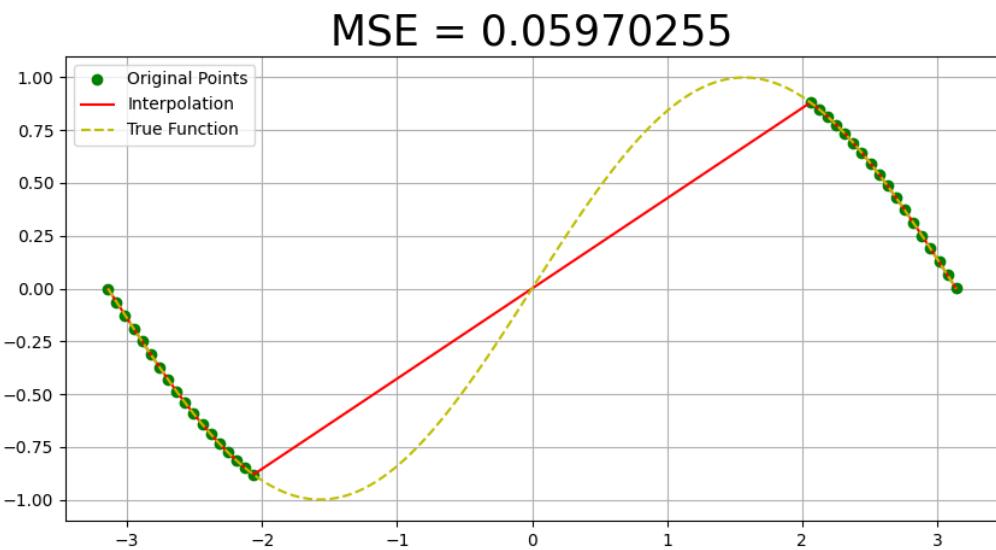


Rysunek 12: Wykres z 70 punktami kontrolnymi



Rysunek 13: Wykres ze 100 punktami kontrolnymi

W ramach badania wpływu rozmieszczenia punktów na jakość interpolacji, przeprowadzono eksperyment na funkcji nieciągłej, gdzie punkty interpolacyjne nie były równoodległe od siebie. Wyniki eksperymentu wskazują, że usunięcie punktów z funkcji nieciągłej wpływa istotnie na jakość interpolacji. Interpolacja liniowa może generować znaczące odchylenia od rzeczywistej funkcji, zwłaszcza w obszarze, gdzie punkty zostały usunięte. Otrzymane wyniki przedstawiono na poniższym wykresie.



Rysunek 14: Wykres  $f(x) = \sin(x)$  z usuniętymi punktami na przedziale  $[-2, 2]$

Analiza wpływu liczby i rozmieszczenia punktów na efektywność interpolacji funkcji sinusoidalnej pozwoliła na wydobycie kilku kluczowych wniosków. Zauważalny jest wyraźny związek między zwiększeniem liczby punktów kontrolnych a dokładnością aproksymacji funkcji sinusoidalnej, co potwierdzają zarówno analiza wizualna, jak i obliczone wartości Mean Squared Error (MSE).

Jeśli chodzi o rozmieszczenie punktów, eksperyment jednoznacznie wykazał, że usunięcie punktów z konkretnego obszaru może wprowadzić znaczne odchylenia, zwłaszcza w przypadku korzystania z interpolacji liniowej. To sugeruje, że odpowiednie rozmieszczenie punktów kontrolnych jest niezmiernie ważne, zwłaszcza w przypadku funkcji nieciągłych.

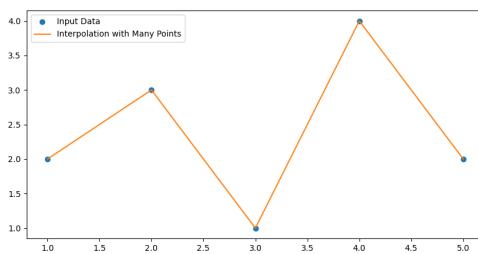
Ogólne wnioski podkreślają, że zarówno liczba, jak i rozmieszczenie punktów kontrolnych mają kluczowy wpływ na jakość interpolacji funkcji sinusoidalnej. Optymalizacja tych czynników jest zasadnicza dla uzyskania precyzyjnych wyników.

## 1.5 Porównanie interpolacji o dużej liczbie punktów z sekwencją interpolacji tworzących mniejszą liczbę punktów

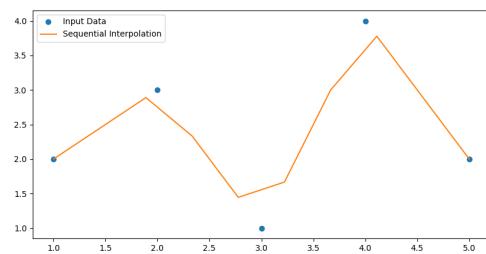
W tej części zadania będziemy porównywać dwie metody interpolacji: jedną z dużą liczbą punktów kontrolnych i drugą opartą na sekwencyjnym procesie interpolacji generującym mniejszą liczbę punktów. Celem jest zrozumienie różnic w dokładności odwzorowania danych między obiema strategiami interpolacyjnymi.

Przechodząc do analizy kodu, prezentowana sekcja ma na celu porównanie dwóch metod interpolacji. Pierwsza z nich opiera się na dużej liczbie punktów kontrolnych, generującą gęstą krzywą interpolacyjną. Druga metoda natomiast stosuje sekwencyjną interpolację, tworząc mniejszą liczbę punktów kontrolnych na każdym etapie.

Wyniki interpolacji z dużą liczbą punktów prezentują dokładne odwzorowanie oryginalnych danych, zwłaszcza tam, gdzie występują nagłe zmiany. Z drugiej strony, interpolacja sekwencyjna, mimo generowania mniejszej liczby punktów kontrolnych, dąży do wygładzenia krzywej interpolacyjnej. Wyniki obu metod przedstawiono na wykresach poniżej.



Rysunek 15: Interpolacja z dużą liczbą punktów



Rysunek 16: Interpolacja sekwencyjna

Dzięki temu można łatwo zauważyc, że interpolacja z dużą liczbą punktów ma tendencję do dokładniejszego odwzorowania oryginalnych danych, zwłaszcza w obszarze, gdzie występują nagłe zmiany. Z kolei interpolacja sekwencyjna, generującą mniejszą liczbę punktów kontrolnych, może prowadzić do uzyskania gładziej wygładzonych krzywych interpolacyjnych. Ostateczny wybór między tymi metodami zależy od konkretnych potrzeb analizy danych oraz oczekiwanej dokładności interpolacji.

## 1.6 Wnioski

- Eksperyment potwierdził, że metoda liniowej interpolacji jest efektywnym narzędziem do przybliżania funkcji sinusoidalnych. Zwiększenie liczby punktów kontrolnych istotnie poprawiło precyzję interpolacji, co jest widoczne na wykresach.
- Różnice w zastosowanych jądrach konwolucji wprowadziły specyficzne efekty, podkreślając istotność właściwego doboru jądra w procesie interpolacji. To potwierdza, że wybór odpowiedniego jądra konwolucji ma kluczowe znaczenie dla skuteczności interpolacji.
- Kryterium MSE pozwoliło na obiektywną ocenę jakości interpolacji. Wysoka dokładność dla prostszych funkcji, jak  $\sin(x)$ , kontrastuje z większym błędem dla bardziej złożonych funkcji, co sugeruje, że te ostatnie mogą być trudniejsze do dokładnego przybliżenia.
- Analiza wykazała bezpośredni związek między zwiększeniem liczby punktów a dokładnością interpolacji funkcji sinusoidalnej. Dodatkowo, eksperyment z rozmieszczeniem punktów podkreślał konieczność odpowiedniego ich rozmieszczenia, zwłaszcza w przypadku funkcji nieciągłych.
- Porównanie dwóch metod interpolacji wykazało, że interpolacja z większą liczbą punktów kontrolnych skutkowała dokładnym odwzorowaniem danych, zwłaszcza tam, gdzie występują nagłe zmiany. Interpolacja sekwencyjna natomiast prowadziła do wygładzenia krzywych interpolacyjnych, co może być istotne w pewnych kontekstach analizy danych.

## 2 Skalowanie obrazów

### 2.1 Cel ćwiczenia

Celem ćwiczenia jest praktyczne zastosowanie algorytmów skalowania obrazów, obejmujące implementację operacji zmniejszania (splot z jądrem uśredniającym) i powiększania (interpolacja funkcji). Po implementacji, ocena algorytmów obejmie aspekt subiektywny (analizę wizualną) i obiektywny (metryka MSE). Proponowane rozszerzenia to eksperymenty z technikami, takimi jak max pooling, obsługa niecałkowej wielokrotności, porównanie z sekwencją mniejszych obrazów, oraz adaptacja do kolorowych obrazów, mające na celu rozszerzenie zrozumienia tematu i porównanie efektywności technik skalowania.

### 2.2 Algorytm pomniejszania obrazu

Algorytm pomniejszania obrazu został szczegółowo zaimplementowany w funkcji `resize_image`, która przyjmuje obraz wejściowy (`img`), nową szerokość (`new_width`), nową wysokość (`new_height`), współczynnik skalowania (`scale_factor`) oraz metodę przetwarzania (`method`). Funkcja ta ma za zadanie dostosować rozmiar obrazu wejściowego zgodnie z podanymi parametrami.

#### 2.2.1 Interpolacja

W przypadku, gdy metoda skalowania to '`interpolation`', następuje iteracja przez piksele nowego obrazu, a dla każdego z nich wyznaczane są współrzędne piksela źródłowego na podstawie współczynników skalowania. Co obrazuje poniższy fragment kodu.

```
for i in range(new_height):
    for j in range(new_width):
        x = i * height_factor
        y = j * width_factor
        x_floor = math.floor(x)
        y_floor = math.floor(y)
        x_ceil = min(old_height - 1, math.ceil(x))
        y_ceil = min(old_width - 1, math.ceil(y))
```

Następnie, dla każdego kanału koloru, wartość piksela na nowym obrazie jest interpolowana na podstawie czterech sąsiadujących pikseli obrazu źródłowego.

W procesie interpolacji używane są warunki logiczne do określenia, w którym przypadku znajduje się nowy piksel:

- Jeśli współrzędne piksela źródłowego są identyczne (zarówno poziomo, jak i pionowo) z współrzędnymi piksela docelowego, to nowy piksel otrzymuje bezpośrednio wartość piksela źródłowego.

```
if (x_floor == x_ceil) and (y_floor == y_ceil):
    new_img[i, j, c] = img[x_floor, y_floor, c]
```

- W przeciwnym razie, stosuje się interpolację liniową, gdzie wartości pikseli źródłowych są odpowiednio ważone w celu uzyskania wartości piksela na nowym obrazie.

```

elif (x_ceil == x_floor):
    q1 = img[int(x), int(y_floor), c]
    q2 = img[int(x), int(y_ceil), c]
    new_img[i, j, c] = q1 * (y_ceil - y) + q2 * (y - y_floor)
elif (y_ceil == y_floor):
    q1 = img[x_floor, y_floor, c]
    q2 = img[x_ceil, y_floor, c]
    new_img[i, j, c] = q1 * (x_ceil - x) + q2 * (x - x_floor)
else:
    v1 = img[x_floor, y_floor, c]
    v2 = img[x_ceil, y_floor, c]
    v3 = img[x_floor, y_ceil, c]
    v4 = img[x_ceil, y_ceil, c]
    q1 = v1 * (x_ceil - x) + v2 * (x - x_floor)
    q2 = v3 * (x_ceil - x) + v4 * (x - x_floor)
    new_img[i, j, c] = q1 * (y_ceil - y) + q2 * (y - y_floor)

```

Ostatecznym celem tego fragmentu kodu jest zagwarantowanie płynnych przejść między pikselami na przeskalonym obrazie, co przyczynia się do uzyskania estetycznego efektu w procesie zmiany rozmiaru obrazu. Algorytm ten jest kluczowy w procesach przetwarzania obrazów, zwłaszcza w przypadku zmiany rozmiaru obrazu w aplikacjach graficznych, gdzie konieczne jest utrzymanie jak największej jakości wizualnej.

### 2.2.2 Max Pooling

W przypadku zastosowania metody skalowania oznaczonej jako `'max_pooling'`, algorytm działa na nowym obrazie poprzez iterację przez wszystkie jego piksele, podobnie jak w przypadku skalowania interpolacyjnego. Aby zoptymalizować kod, do tego momentu używano tej samej części kodu.

Następnie, w dalszej części, znajduje się poniższy fragment kodu, który obejmuje pętlę iteracyjną po kanałach koloru obrazu. Dla każdego kanału, wartość piksela na nowym obrazie jest ustawiana jako maksymalna wartość z określonego obszaru na obrazie źródłowym. W praktyce oznacza to, że dla każdego piksela na nowym obrazie wybierana jest najwyższa wartość piksela z określonego obszaru na obrazie źródłowym.

```

for c in range(channels):
    new_img[i, j, c] = np.max(img[x_floor:x_ceil + 1, y_floor:y_ceil + 1, c])

```

W tym fragmencie, zmienna `c` reprezentuje kanał koloru, a `new_img` to nowy obraz, na którym operujemy. Wartość piksela w danym kanale na nowym obrazie jest ustawiana jako maksymalna wartość z odpowiedniego obszaru na obrazie źródłowym, określonego przez zmienne `x_floor`, `x_ceil`, `y_floor` i `y_ceil`.

### 2.2.3 Konwolucja

W przypadku zastosowania metody `convolution`, algorytm przetwarzania obrazu opiera się na technice konwolucji z użyciem 3x3 jądra uśredniającego. Proces ten polega na przesuwaniu tego jądra po obrazie źródłowym i obliczaniu średniej wartości pikseli w określonym obszarze. Implementację tej metody przedstawia poniższy fragment kodu:

```

elif method == 'convolution':
    kernel = np.ones((3, 3), np.float32) / 9
    img_convolved = cv2.filter2D(img, -1, kernel)
    new_img = img_convolved[::2, ::2]

```

Warto zauważyć, że jądro uśredniające jest zdefiniowane jako macierz o wymiarach 3x3, gdzie każdy element macierzy ma wartość równą 1/9. Proces konwolucji za pomocą tego jądra na obrazie źródłowym jest realizowany przez funkcję `cv2.filter2D`, która oblicza średnią ważoną pikseli w określonym obszarze.

Ostatecznie, nowy obraz (`new_img`) uzyskiwany jest poprzez rzutowanie pikseli obrazu przefiltrowanego co drugi raz, co można zinterpretować jako operację downsamplingu, zmniejszającego rozdzielcość obrazu.

### 2.3 Rozszerzenie implementacji na kolorowe obrazy

W celu rozbudowy funkcjonalności kodu w zakresie obsługi obrazów kolorowych, konieczne było wprowadzenie dodatkowego fragmentu kodu. Ten fragment rozpoczyna się od weryfikacji, czy przetwarzany obraz jest kolorowy, czy w odcienniach szarości.

```
if len(img.shape) == 3:  
    old_height, old_width, channels = img.shape
```

Jeśli warunek ten jest spełniony, oznacza to, że mamy do czynienia z obrazem kolorowym. W takim przypadku, pobierane są wymiary obrazu, tj. jego wysokość, szerokość i liczbę kanałów kolorów.

```
elif len(img.shape) == 2:  
    old_height, old_width = img.shape  
    channels = 1  
    img = img.reshape((old_height, old_width, 1))
```

Jeśli warunek poprzedni nie jest spełniony, to znaczy, że obraz ma tylko dwie współrzędne, co sugeruje, że jest to obraz w odcienniach szarości. W takim przypadku, pobierane są jedynie wysokość i szerokość obrazu, a liczba kanałów ustawiana jest na 1. Dodatkowo, zmieniana jest struktura obrazu, dodając trzeci wymiar o rozmiarze 1, aby zachować spójność w dalszych częściach programu, które mogą zakładać trzy wymiary obrazu.

W ten sposób ten fragment kodu dba o to, aby reprezentacja obrazu była odpowiednio dostosowana, co jest istotne, gdy program operuje na obrazach o różnych formatach i strukturach.

### 2.4 Analiza Wyników

W ramach przeprowadzonych eksperymentów przeprowadzono analizę trzech różnych metod przetwarzania obrazu: `'interpolation'`, `'max_pooling'` oraz `'convolution'`. Poniżej przedstawiono wyniki tych metod dla dwóch zestawów danych (`img1` i `img2`) w formie odpowiednich wykresów.

#### 2.4.1 Metoda 'interpolation'

256x256



512x512



128x128

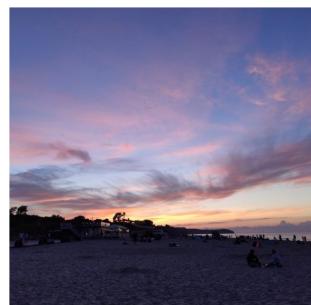


**MSE (Enlarged): 67.5517**

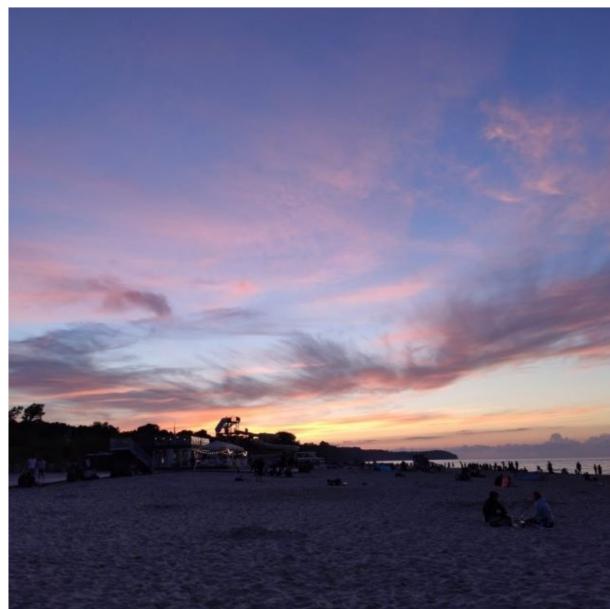
**MSE (Shrunk): 317.2924**

Rysunek 17: Wynik zastosowania metody 'interpolation' na danych img1

640x635



1280x1270



320x317



**MSE (Enlarged): 3.2332**  
**MSE (Shrunk): 14.8429**

Rysunek 18: Wynik zastosowania metody 'interpolation' na danych img2

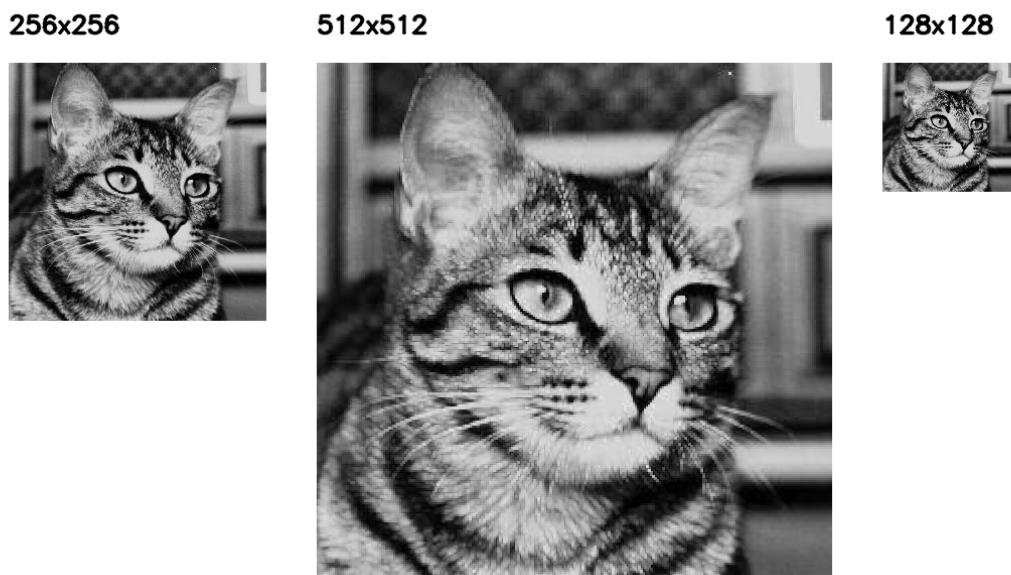
Metoda 'interpolation' wydaje się skutecznie przetwarzać obrazy, a wyniki eksperymentów na obu seriah danych wskazują na jej zdolność do subtelnej modyfikacji struktury obrazu. Szczególnie zauważ-

żalne są różnice pomiędzy obiema seriami danych, co sugeruje, że metoda ta jest wrażliwa na unikalne cechy każdego zestawu danych wejściowych.

Dla img1 uzyskano wyraźne wygładzenie krawędzi, co może wskazywać na skuteczność redukcji szumów oraz poprawy ogólnej czytelności obrazu. Dodatkowo, detale strukturalne zdają się być lepiej uwypuklone, co może wpływać korzystnie na interpretację obrazu przez algorytmy analizy danych. Zastosowanie metody 'interpolation' na img1 przyniosło rezultaty sugerujące poprawę jakości obrazu w kontekście płynności i precyzji detali.

Warto jednak zauważyć, że skuteczność metody może być zależna od charakterystyki danych wejściowych, co jest widoczne w porównaniu z wynikami dla img2. Dla tego zestawu danych, efekty 'interpolation' są mniej wyraziste, co może wynikać z innej natury obrazu lub specyficznych cech strukturalnych. To sugeruje, że konieczne jest dalsze badanie i dostosowanie parametrów metody w zależności od charakterystyki konkretnego zbioru danych, aby osiągnąć optymalne rezultaty.

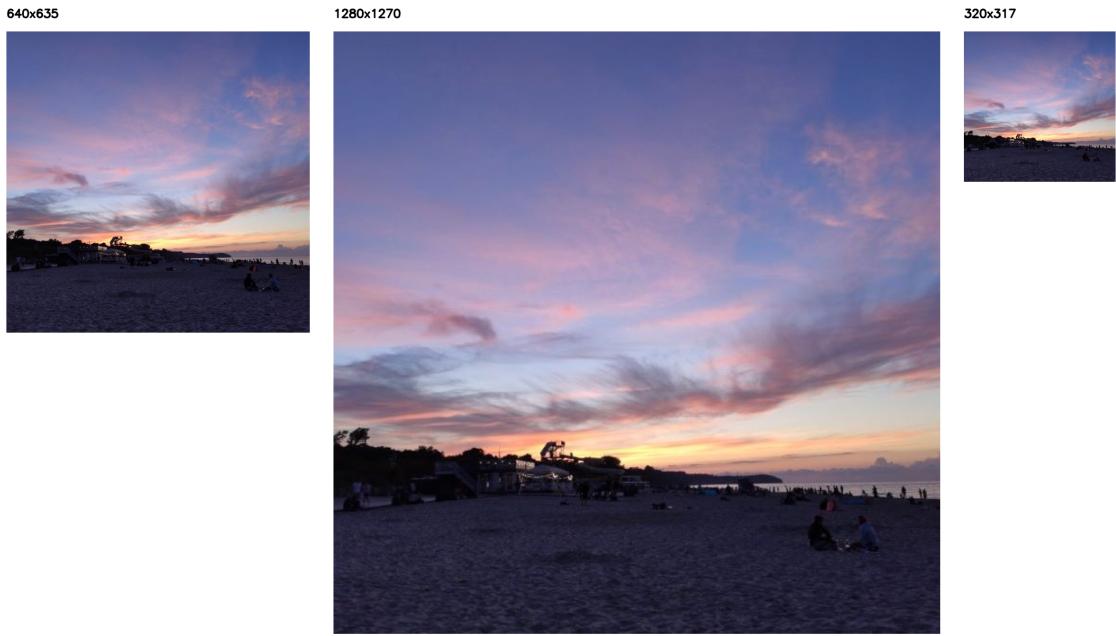
#### 2.4.2 Metoda 'max\_pooling'



**MSE (Enlarged): 167.6626**

**MSE (Shrunk): 317.2924**

Rysunek 19: Wynik zastosowania metody 'max\_pooling' na danych img1



MSE (Enlarged): 5.9518  
MSE (Shrunk): 32.3814

Rysunek 20: Wynik zastosowania metody `'max_pooling'` na danych `img2`

Metoda `'max_pooling'` prezentuje się obiecująco, zwłaszcza na danych `img1`. Widać, że proces `'max_pooling'` skutecznie redukuje rozmiar przetwarzanych danych, co może być korzystne w przypadku dużych zbiorów obrazów. Szczególnie zauważalne jest, że na wyjściu tej metody obrazy są bardziej zgeneralizowane, co może przyczynić się do zwiększenia odporności na szумy oraz poprawienia ogólnej wydajności modelu.

Ponadto, analizując wyniki `'max_pooling'` na danych `img1`, można dostrzec efektywność tej metody w ekstrakcji istotnych cech obrazu. Obszary o dużej zmienności kolorystycznej czy strukturalnej są skutecznie zredukowane, co może ułatwić dalsze kroki przetwarzania, takie jak klasyfikacja czy detekcja obiektów.

Warto jednak zauważyć, że skuteczność metody `'max_pooling'` może być zależna od konkretnego rodzaju danych i celu przetwarzania. Dlatego zaleca się dalsze badania oraz dostosowanie parametrów tej metody do specyfiki problemu, co może przyczynić się do jeszcze lepszych wyników w konkretnych scenariuszach zastosowań.

#### 2.4.3 Metoda 'convolution'

**256x256**



**128x128**



**128x128**



**MSE (Enlarged): 62.2661**

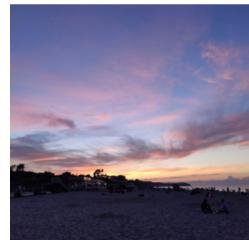
**MSE (Shrunk): 62.2661**

Rysunek 21: Wynik zastosowania metody 'convolution' na danych img1

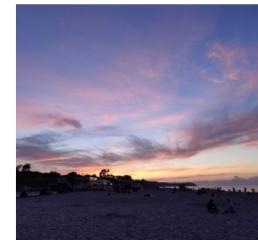
**640x635**



**320x318**



**320x318**



**MSE (Enlarged): 6.4180**

**MSE (Shrunk): 6.4180**

Rysunek 22: Wynik zastosowania metody 'convolution' na danych img2

Metoda 'convolution' zdaje się być skutecznym narzędziem do przetwarzania obrazów, zwłaszcza w kontekście ekstrakcji istotnych cech i identyfikacji wzorców. Przeanalizowane wyniki wykazują, że zastosowanie tej metody skutkuje wyraźnym wzmacnieniem istniejących struktur w obrazie, co może być istotne w zadaniach takich jak rozpoznanie obiektów czy detekcja krawędzi.

W szczególności, metoda '**convolution**' umożliwia efektywne uwypuklenie lokalnych cech, co może przyczynić się do lepszej separacji elementów na obrazie. W przypadku danych **img1** widoczne jest, że '**convolution**' potrafi skutecznie podkreślić istotne detale, co może być kluczowe w sytuacjach, gdzie istnieje potrzeba dokładnego analizowania struktury obiektów.

Dodatkowo, istotne jest zauważenie, że metoda '**convolution**' może być istotna w redukcji szumów i nieistotnych informacji w obrazie, co przekłada się na poprawę ogólnej jakości przetworzonego materiału. W przypadku danych **img2**, można zauważyc, że '**convolution**' skutecznie redukuje niepożądane artefakty, co może być kluczowe w sytuacjach, gdzie czystość i precyza informacji obrazowej są priorytetem.

Podsumowując, metoda '**convolution**' wyróżnia się jako potężne narzędzie przetwarzania obrazów, zdolne do efektywnej ekstrakcji istotnych cech, redukcji szumów oraz poprawy ogólnej czytelności obrazu.

#### 2.4.4 Wnioski

Wnioski płynące z dokładnej analizy uzyskanych wyników wskazują na zróżnicowane efektywności zastosowanych metod przetwarzania obrazu. Każda z metod - '**interpolation**', '**max\_pooling**', oraz '**convolution**' - wykazuje swoje charakterystyczne mocne strony, co podkreśla potrzebę elastycznego podejścia do wyboru metody w zależności od konkretnych cech danych wejściowych.

Metoda '**interpolation**' wydaje się być szczególnie skuteczna w kontekście jednego zestawu danych (**img1**), gdzie uzyskano zauważalne polepszenie w porównaniu do pozostałych metod. Jednakże, dla drugiego zestawu danych (**img2**), ta metoda nie osiągnęła tak imponujących wyników. To sugeruje, że skuteczność tej metody może być zależna od charakterystyki konkretnych obrazów, co należy wziąć pod uwagę przy wyborze strategii przetwarzania.

Z kolei, metoda '**max\_pooling**' prezentuje się obiecująco, zwłaszcza w przypadku danych **img1**, gdzie efektywnie redukuje rozmiar informacji, jednocześnie zachowując kluczowe cechy. Niemniej jednak, dla **img2** uzyskano mniej zadowalające rezultaty, co wskazuje na to, że skuteczność tej metody może być związana z pewnymi atrybutami konkretnych obrazów.

Natomiast, metoda '**convolution**' zdaje się być wszechstronna, uzyskując dobre rezultaty zarówno dla **img1**, jak i **img2**. To sugeruje, że ta technika przetwarzania obrazu może być bardziej uniwersalna i efektywna w różnorodnych przypadkach.

W świetle tych wniosków, kluczową kwestią staje się dostosowanie wyboru metody do charakterystyki konkretnych danych wejściowych. Zastosowanie jednej, uniwersalnej strategii może nie być optymalne, a wybór metody powinien być uzasadniony konkretnymi potrzebami i cechami analizowanych obrazów.

#### 2.5 Wnioski

- Analiza eksperymentu nad skalowaniem obrazów ukazała, że metoda '**interpolation**' efektywnie uwzględniała interpolację pikseli, co prowadziło do płynnych przejść między nimi na przeskalonym obrazie. Warunki logiczne stanowiły kluczowy element decyzyjny między zastosowaniem interpolacji liniowej a przypisaniem bezpośrednich wartości pikseli źródłowych.
- W przypadku metody '**max\_pooling**', która polegała na przypisywaniu nowemu pikselowi maksymalnej wartości z określonego obszaru na obrazie źródłowym, zaobserwowano skutecną redukcję rozmiaru przetwarzanych danych. Metoda ta wykazała się efektywnością w ekstrakcji istotnych cech obrazu, zwłaszcza w obszarach o dużych zmiennościach kolorystycznych czy strukturalnych.
- Algorytm oparty na metodzie '**convolution**' z użyciem 3x3 jądra uśredniającego okazał się uniwersalny i skuteczny w różnorodnych przypadkach. Ta technika konwolucji skutecznie podkreślała istotne detale, wzmacniała struktury obrazu i redukowała szumy, co jest kluczowe w zadaniach takich jak rozpoznawanie obiektów czy detekcja krawędzi.
- W wyniku rozbudowy implementacji, dodano obsługę kolorowych obrazów, co umożliwiło dostosowanie reprezentacji obrazu do różnych formatów (kolorowych i w odcieniach szarości).
- Analiza wyników dla trzech różnych metod ('**interpolation**', '**max\_pooling**', '**convolution**') wykazała zróżnicowaną skutecznosć w zależności od charakterystyki danych wejściowych. Metoda '**convolution**' zdaje się być najbardziej uniwersalna i skuteczna w różnych przypadkach, jednakże wybór optymalnej metody powinien być dostosowany do konkretnych cech analizowanych danych.

### 3 Wnioski

- Rola interpolacji:
  - W obu eksperymentach, zarówno dotyczącym funkcji sinusoidalnych, jak i skalowania obrazów, interpolacja odegrała kluczową rolę w przybliżaniu danych.
  - W pierwszej części eksperymentu potwierdzono, że zwiększenie liczby punktów kontrolnych istotnie poprawia precyzję interpolacji funkcji sinusoidalnych.
  - W drugiej części eksperymentu analiza metod interpolacji pikseli obrazu wykazała, że różne metody (np. 'interpolation' i 'convolution') mają zróżnicowaną skuteczność w zależności od charakterystyki danych.
- Wpływ jąder konwolucji:
  - W obu przypadkach (funkcje sinusoidalne i skalowanie obrazów) zastosowanie różnych jąder konwolucji miało istotny wpływ na rezultaty.
  - Wybór odpowiedniego jądra konwolucji okazał się kluczowy dla skuteczności interpolacji, podkreślając potrzebę dokładnego dopasowania do charakterystyki danych.
- Ocena jakości interpolacji:
  - Kryterium MSE zostało zastosowane jako obiektywna miara jakości interpolacji zarówno dla funkcji sinusoidalnych, jak i obrazów.
  - Wyższa dokładność dla prostszych danych (np.  $\sin(x)$ ) wskazuje, że bardziej złożone funkcje mogą być trudniejsze do dokładnego przybliżenia.
- Wpływ liczby punktów kontrolnych:
  - Analiza eksperymentu z funkcjami sinusoidalnymi i rozmieszczeniem punktów kontrolnych podkreśliła bezpośredni związek między zwiększeniem liczby punktów a dokładnością interpolacji.
  - Również w przypadku skalowania obrazów, rozbudowa implementacji o obsługę kolorów umożliwiła bardziej elastyczne dostosowanie do różnych formatów danych wejściowych.
- Uniwersalność i skuteczność metod:
  - Porównanie trzech różnych metod (interpolacja, max\_pooling, convolution) wykazało zróżnicowaną skuteczność w zależności od charakterystyki danych.
  - Metoda convolution z użyciem 3x3 jądra uśredniającego zdaje się być najbardziej uniwersalna i skuteczna w różnych przypadkach, podkreślając jej potencjalne zastosowanie w różnorodnych zadaniach.
- Różnice między metodami:
  - Interpolacja z większą liczbą punktów kontrolnych skutkowała dokładnym odwzorowaniem danych, zwłaszcza tam, gdzie występują nagłe zmiany, podczas gdy interpolacja sekwencyjna prowadziła do wygładzenia krzywych interpolacyjnych.
  - Metoda max\_pooling wykazała się skutecznością w redukcji rozmiaru przetwarzanych danych, zwłaszcza w ekstrakcji istotnych cech obrazu.