Kompresja stratna audio

Kompresja stratna (ang. lossy compression) – nieodwracalna metoda zmniejszenia ilości danych poprzez usunięcie przez koder-dekoder danych mniej istotnych dla zmysłów człowieka, dopuszczająca przy odtwarzaniu (dekompresji) skompresowanego sygnału, takie zniekształcenie sygnału w porównaniu do sygnału pierwotnego, powoduje, że jego percepcja przez człowieka (słuch, wzrok) będzie odbierana jako identyczna lub zbliżona do oryginału, ewentualnie będzie od niego odbiegać, lecz za świadomą zgodą użytkownika. Zaletą kompresji stratnej jest uzyskiwanie wysokiego stopnia kompresji, o wiele wyższego niż w metodach kompresji bezstratnej.

W ramach instrukcji proszę zapoznać się z poniższymi informacjami, przykładami i fragmentami kodu, na podstawie których powinniście zaimplementować funkcje. Kompresje Law nie wymagają użycia żadnych pętli. Do kompresji A-law trzeba zapoznać się z fragmentem na temat indeksowania logicznego, dzięki któremu da się zaimplementować kompresję A-law w 3-4 linijkach. Do kompresji DPCM macie gotowe kody, trzeba tylko na ich podstawie napisać dekodery, które są już realizowane w funkcjach kompresji.

Zadania

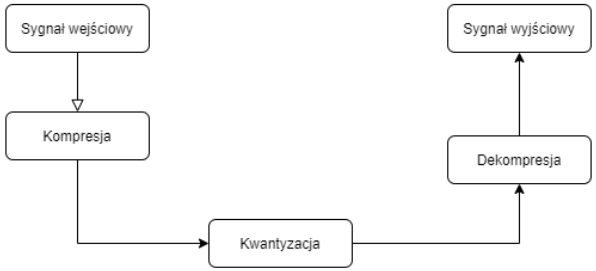
1. Do zaimplementowania pary osobnych funkcji koder i dekoder, każdy algorytm ma mieć osobny zestaw funkcji. Proszę nie łączyć kodera i dekodera w jedną funkcję. Proszę pokazać poprawność działania na przypadkach testowych z instrukcji:
    1. Metody A-law i μ-law, bez użycia pętli **(0,25 pkt)**,
    2. Metody DPCM bez predykcji i dowolną predykcją (opisać jaką predykcję wykorzystujecie, ile elementów itd.), wykorzystując wzorce z instrukcji **(0,25 pkt)**.
2. Przebadać wpływ poprawnie zaimplementowanych metod kompresji, na jakość plików dźwiękowych. Na podstawie co najmniej 3 różnych plików *sing_*, przynajmniej po jednym z każdej klasy low, medium oraz high:
    1. Przeanalizować działanie obu metod przemyśleć i krótko opisać **własnymi słowami** ich działanie. Co ulega kompresji etc. **(0,1 pkt)**
    2. Zbadać jakość dźwięku po działaniu domyślnym (kompresja do 8 bitów) wszystkich metod, dźwięk nie powinien brzmieć źle na tym etapie. Jeżeli wasze wrażenia odsłuchowe są bardzo złe, radzę sprawdzić kod. Jeżeli to nie pomoże to w ostateczności proszę wczytać pliki na *int16* i zrzutować do *float32*, ale nie powinno być to konieczne. **(0,1 pkt)**
    3. Sprawdzić, czy i do jak niskiej ilości bitów informacji jesteśmy w stanie skompresować nasz sygnał, aby być w stanie jeszcze rozpoznać jego zawartość. Zacząć od 8 bitów i obżąć ich ilość aż do momentu, którym przestajemy rozpoznawać zawartość (7,6,5,4,3,2 bity). Dane najlepiej zamieścić w formie tabeli (pliki na jednej osi ilość bitów na drugiej), nie trzeba się rozpisywać, tylko proszę analizować każdy plik osobno nie uogólniać wniosków — podpowiedź: wrażenia odsłuchowe dla różnych plików dźwiękowych powinny być trochę inne w zależności od ilości bitów. **(0,3 pkt)**

Do oddania

- kod źródłowy (jeden plik *.py*)
- sprawozdanie z obserwacjami i wynikami (format *PDF*)

Kompresja stratna audio — metody LAW

Algorytmy A-law i μ-law pracują na sygnałach *float* ($< -1, 1 >$). Praca z algorytmami law będzie przebiegała według schematu:



schemat kompresji law

Algorytm A-law

Algorytm A-law jest standardem powszechnie stosowanym w 8-bitowej kompresji sygnału w Europie. Kompresja wykorzystuje zmniejszenie zakresu dynamicznego dla pewnych częstotliwości. Kompresja odbywa się za pomocą równania $y = F(x)$, natomiast do dekompresji wykorzystujemy równanie $x' = F^{-1}(y)$. W obu przypadkach wykorzystujemy parametr $A = 87.6$, który jest standardem. Sposób, w jaki wyglądają poszczególne krzywe kompresji oraz sygnał po dekompresji zaprezentowano pod koniec instrukcji.

$$F(x) = sign(x) \begin{cases} \dfrac{A|x|}{1+\ln(A)}, & \text{gdy } |x| < \dfrac{1}{A} \\ \dfrac{1+\ln(A|x|)}{1+\ln(A)}, & \text{gdy } \dfrac{1}{A} \leq |x| \leq 1 \end{cases}$$

$$F^{-1}(y) = sign(y) \begin{cases} \dfrac{|y|(1+\ln(A))}{A}, & \text{gdy } |y| < \dfrac{1}{1+\ln(A)} \\ \dfrac{\exp(|y|(1+\ln(A))-1)}{A}, & \text{gdy } \dfrac{1}{1+\ln(A)} \leq |y| \leq 1 \end{cases}$$

Algorytm μ-law (lub mu-law)

Algorytm μ-law (mu-law lub u-law) jest standardem powszechnie stosowanym w 8-bitowej kompresji sygnału w stosowany w Stanach Zjednoczonych oraz Japonii. Kompresja wykorzystuje zmniejszenie zakresu dynamicznego dla pewnych częstotliwości. Kompresja odbywa się za pomocą równania $y = F(x)$, natomiast do dekompresji wykorzystujemy równanie $x' = F^{-1}(y)$. W obu przypadkach wykorzystujemy parametr $\mu = 255$, który jest standardem. Sposób, w jaki wyglądają poszczególne krzywe kompresji oraz sygnał po dekompresji zaprezentowano pod koniec instrukcji.

$$F(x) = sign(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)}, \quad \text{gdy } -1 \leq x \leq 1$$

$$F^{-1}(y) = sign(y)\frac{1}{\mu}((1+\mu)^{|y|}-1), \quad \text{gdy } -1 \leq y \leq 1$$

Funkcje, które potrzebujecie: funkcja znaku, logarytm naturalny, funkcja eksponencjalna:

```
    01. np.sign()
02. np.log()
03. np.exp()
```

Uwagi końcowe i prezentacja działania

Wszystkie metody kompresji powinny działać po ich kwantyzacji do 8-bitów, chyba że jest popełniany jakiś błąd zaokrągleń. Jeżeli chodzi o testowanie prawidłowego działania

Kontakt

Materiały

Przetwarzanie Obrazów

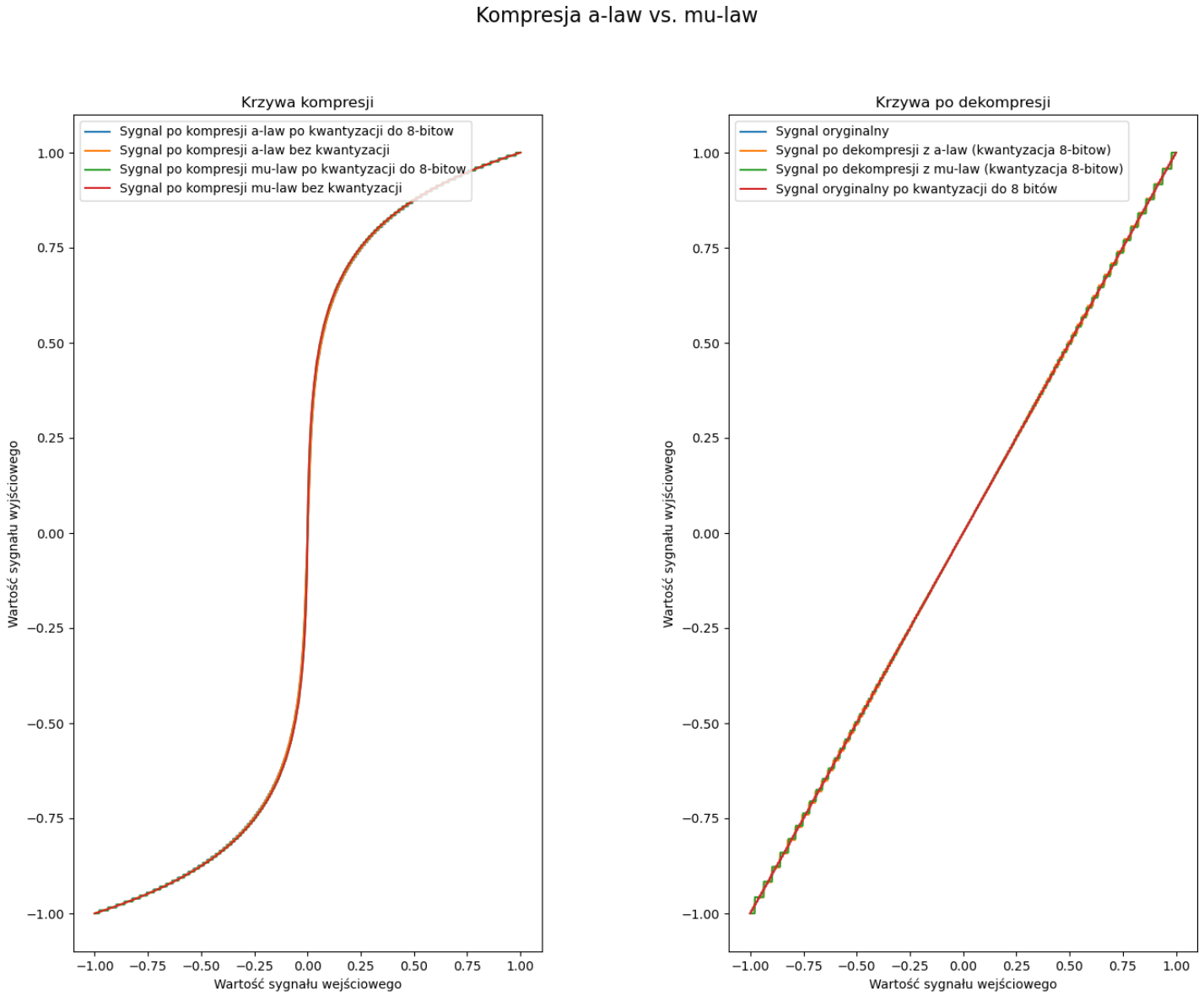
Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

metod law, to możemy je sprawdzić, wykorzystując tu aspekt, że kodują one całą przestrzeń sygnału. Na pełnej przestrzeni sygnału (np. `x=np.linspace(-1,1,1000)`) powinny się one prezentować jak na poniższych wykresach.



Kontakt

Materiały

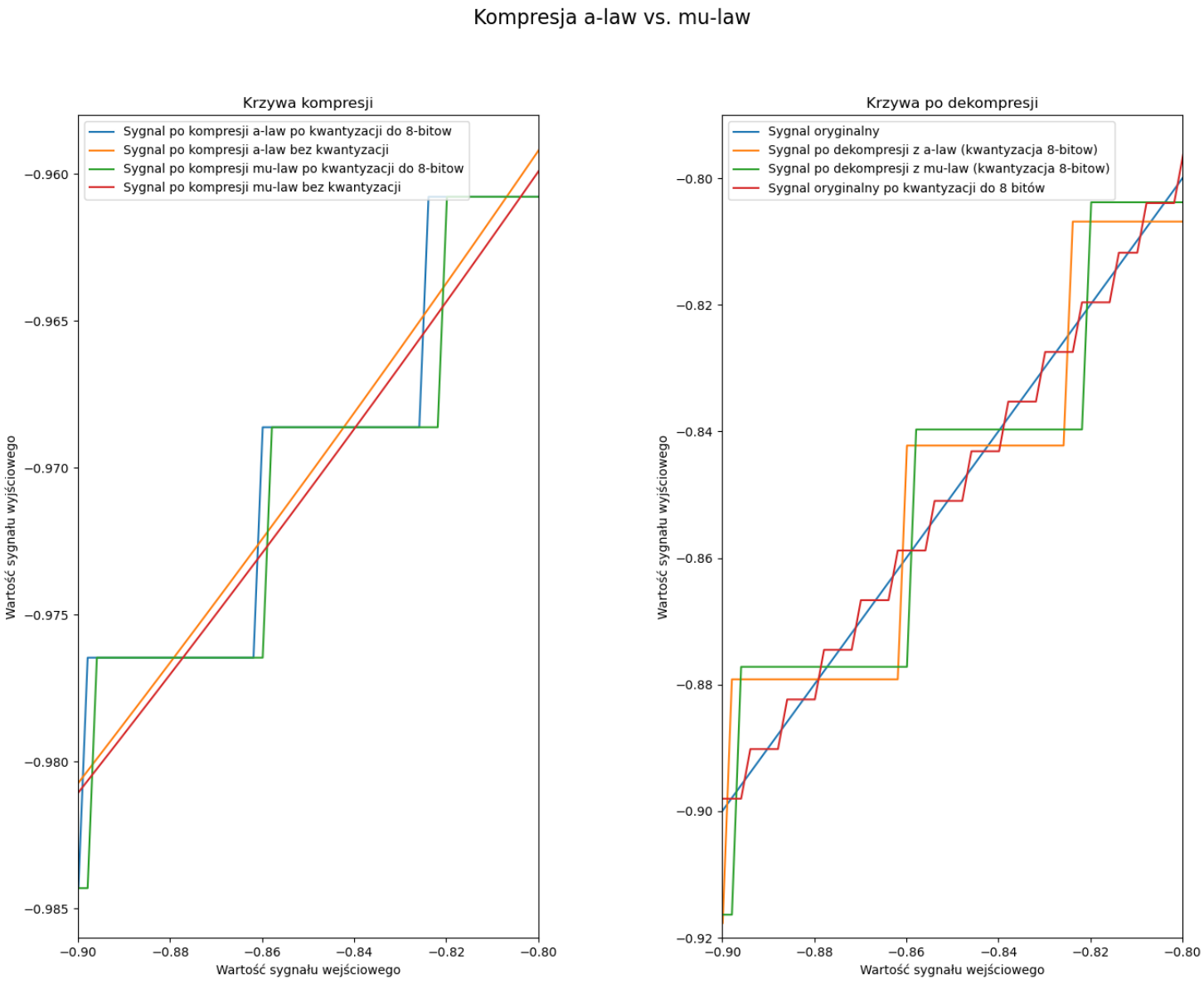
Przetwarzanie Obrazów

Systemy Multimedialne

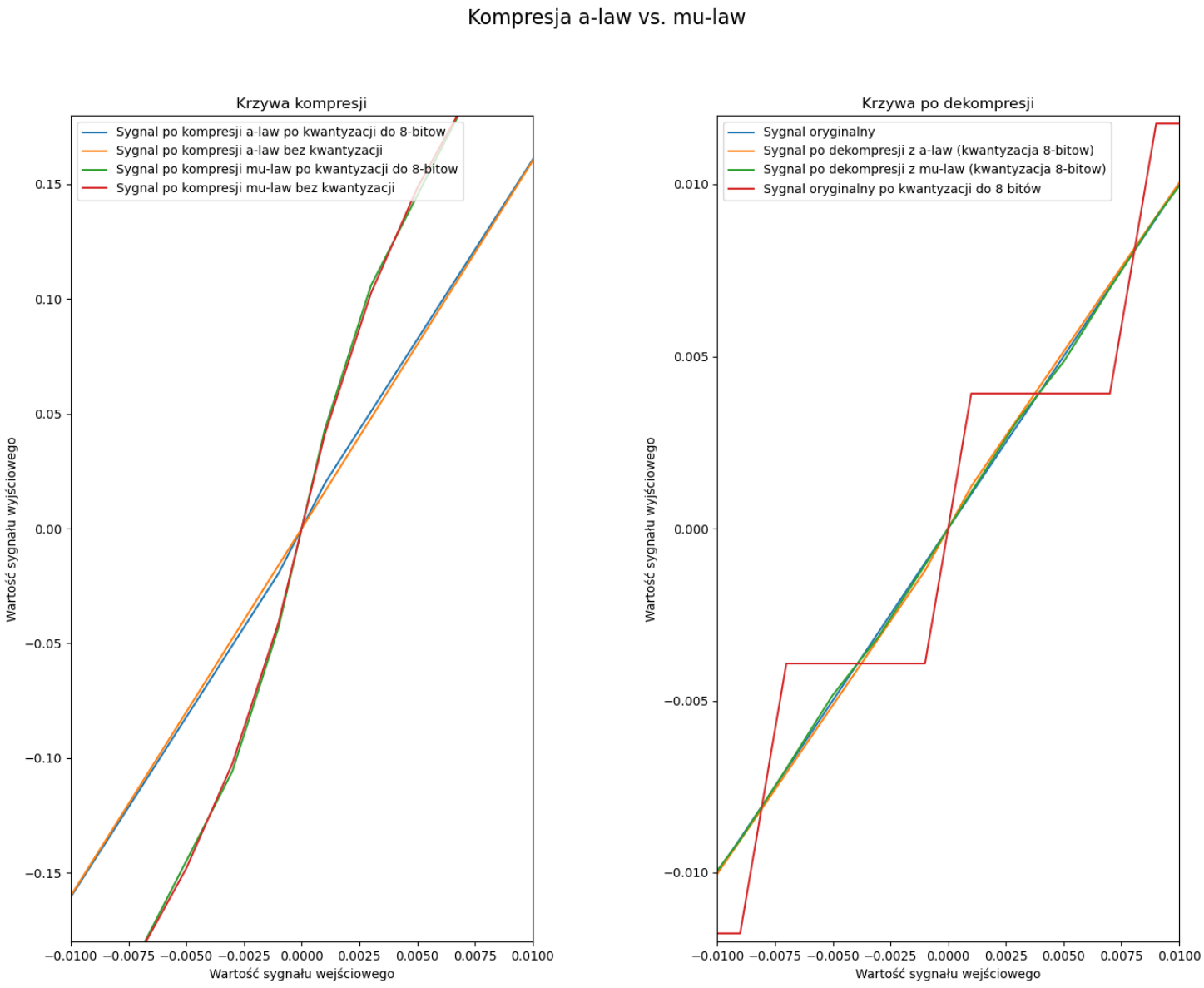
Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania



Kompresja a-law, a μ-law - rzut szczegółowy: wartości skrajne



Kompresja a-law, a μ-law - rzut szczegółowy: okolice zera

Jeżeli chodzi o porównanie wszystkich metod, to możemy [illegible] np.pi*x*4) x z poprzedniego przykładu) jak poniżej. ***Możemy również za pomocą tego przykładu zobaczyć czy dobrze wykonujemy*** [illegible], gdy będzie coś nie tak, to będziemy widzieć obcinanie wartości lub inne dziwne

spłaszczenia albo błędy.

Kontakt

Materiały

Przetwarzanie Obrazów

Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

Kontakt

Materiały

Przetwarzanie Obrazów

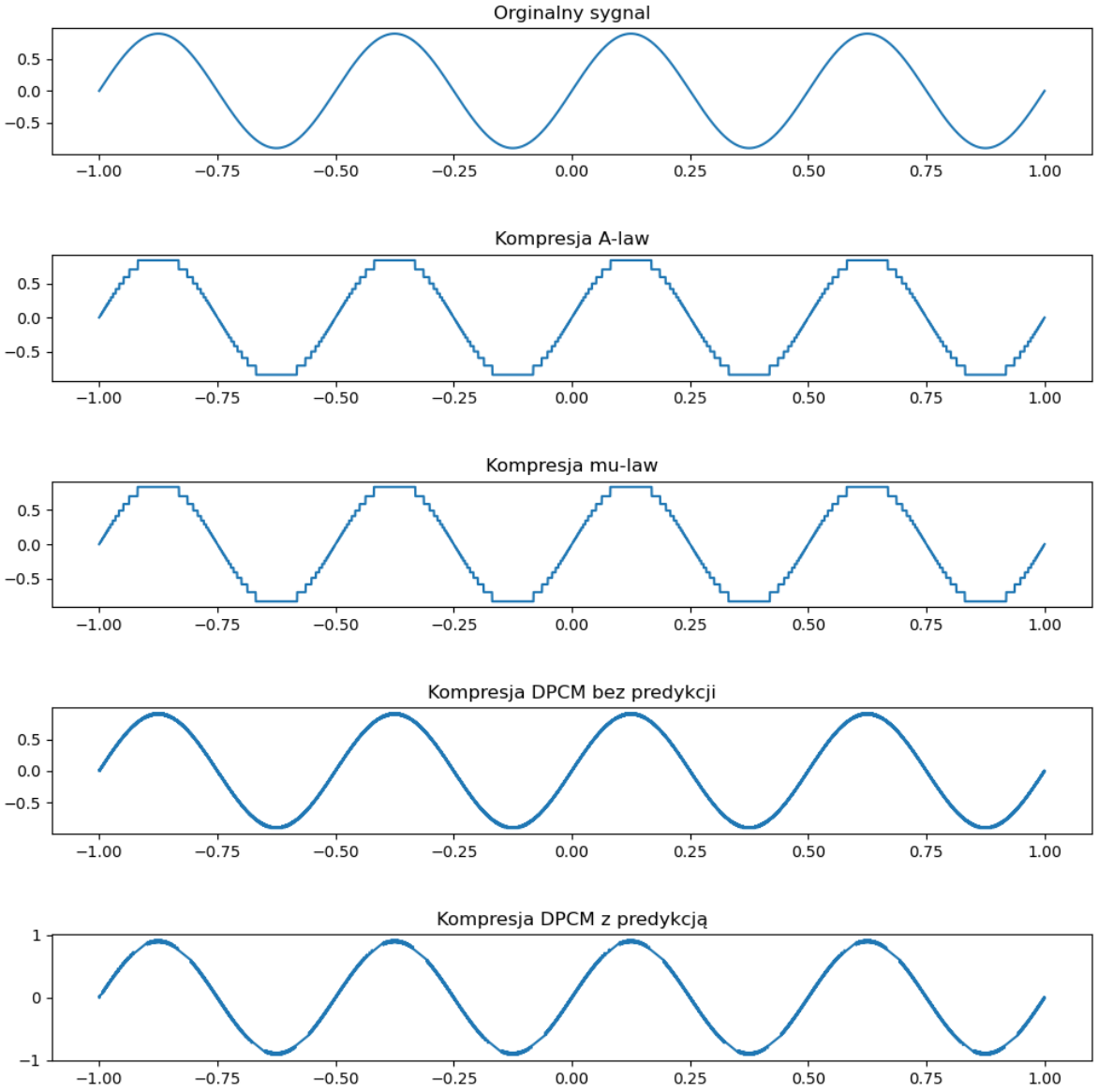
Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

Przyklad A kwnatyzacja do 6 bitów



Porównanie kompresji wersja A

Kontakt

Materiały

Przetwarzanie Obrazów

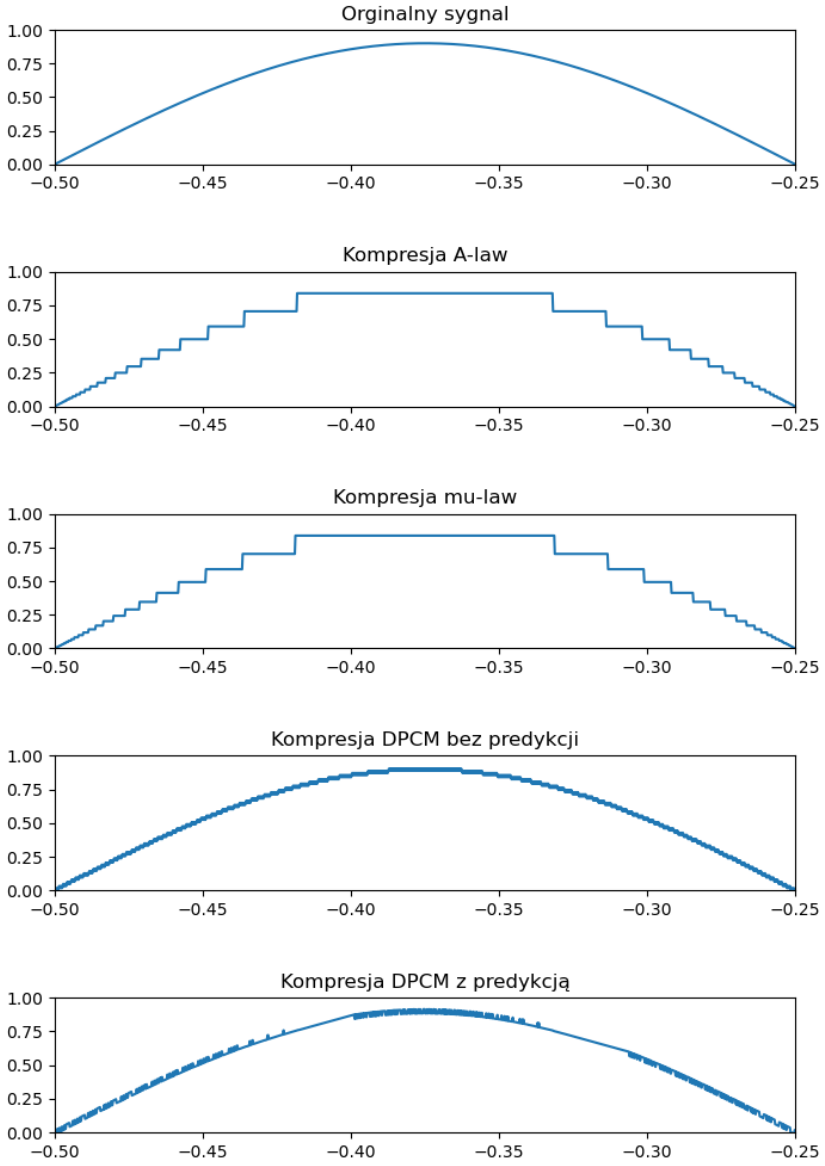
Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

Przyklad A kwnatyzacja do 6 bitów



Kontakt

Materiały

Przetwarzanie Obrazów

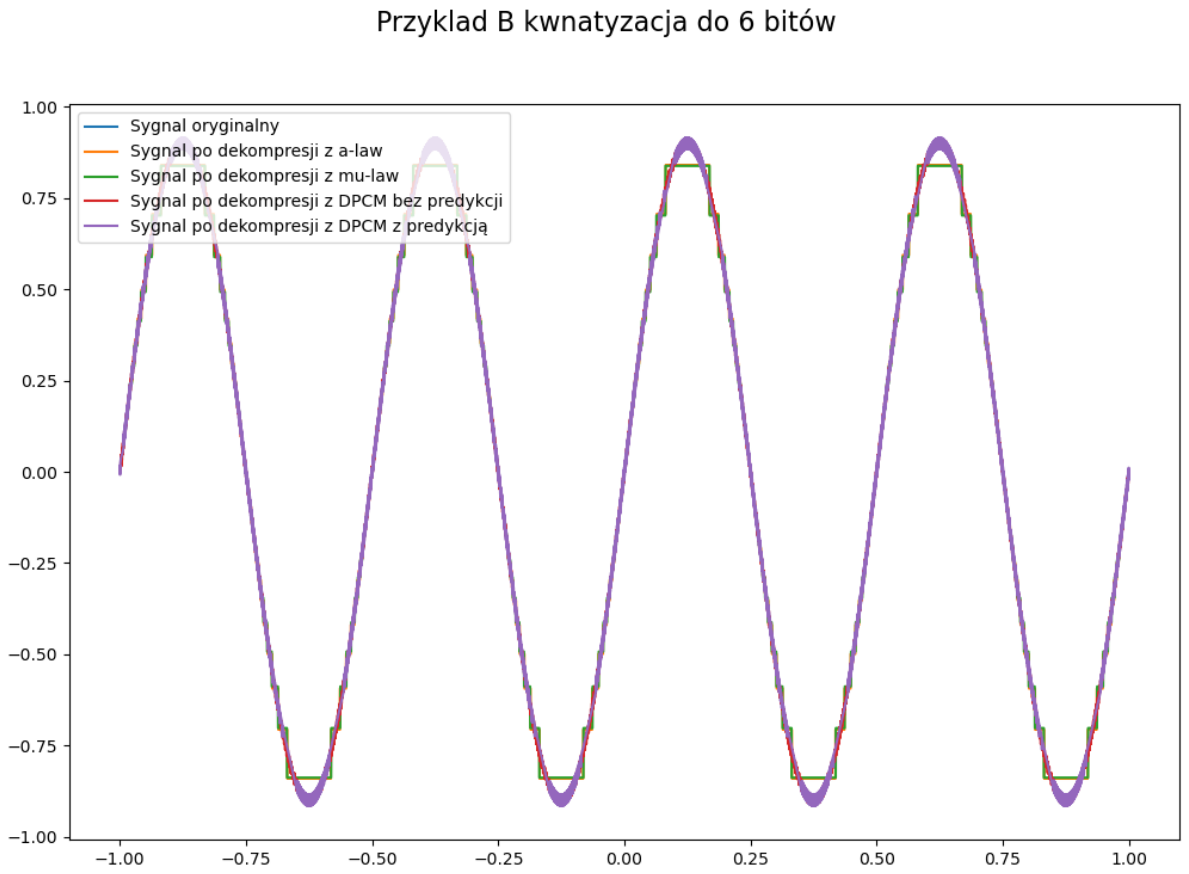
Systemy Multimedialne

Interaktywne Systemy Multimedialne

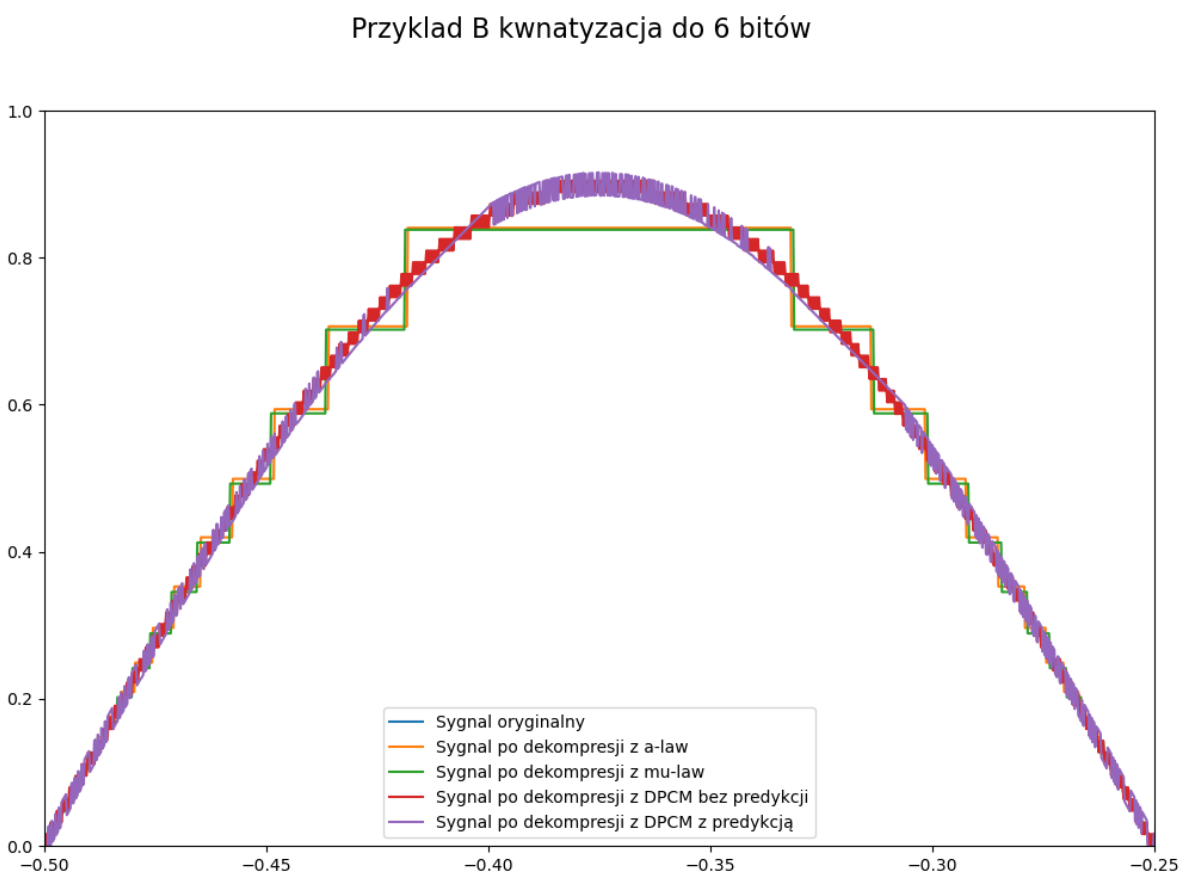
Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

Porównanie kompresji wersja A



Porównanie kompresji wersja B



Porównanie kompresji wersja B

Indeksowanie logiczne (jak łatwo zrobić A-law)

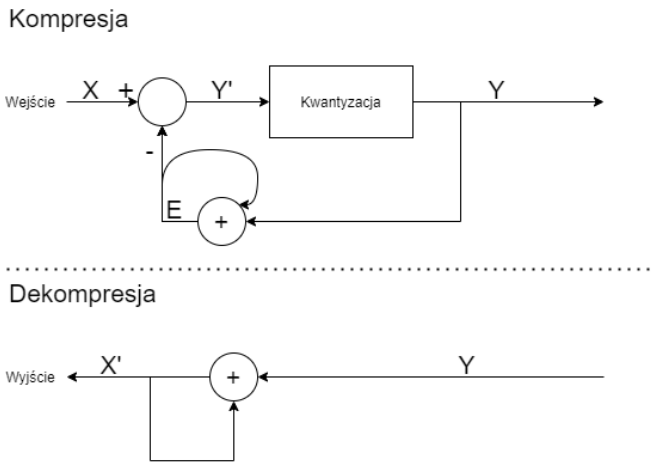
Indeksowanie logiczne to sposób, w jaki NumPy pozwala nam wykonywać operacje na części tabeli bez konieczności wykorzystywania pętli. Poniżej kilka linijek kodu, które pozwolą wam to zrozumieć. Uruchomcie sobie poniższe linijki w Pythonie i odpowiedzcie sobie pytanie, co dzieje się w zaznaczonych linijkach:

```
    01. R=np.random.rand(5,5)
02. A=np.zeros(R.shape)
03. B=np.zeros(R.shape)
04. C=np.zeros(R.shape)
05.
06.
07. idx=R<0.25
08. A[idx]=1 # <-
09. B[idx]+=0.25 # <-
10. C[idx]=2*R[idx]-0.25 # <-
```

```
11. C[np.logical_not(idx)]=4*R[np.logical_not(idx)]-0.5 # <-
12. print(R)
13. print(A)
14. print(B)
15. print(C)
```

Kompresja stratna audio — metody DPCM

DPCM (ang. Differential Pulse-Code Modulation) – metoda kompresji stratnej przeznaczona głównie dla sygnałów dźwiękowych. Normalny algorytm wykorzystuje predykcję kolejnych wartości sygnałów i koduje jedynie różnice pomiędzy wartością szacowaną a rzeczywistą. My zajmiemy się jego uproszczonymi wersjami, kodującymi zmiany pomiędzy kolejnymi wartościami, zamiast samych wartości. Zmiany te zwykle są mniej dynamiczne niż same wartości sygnału. Schemat kodowania został zaprezentowany na poniższym schemacie.



Schemat uproszczonej wersji DPCM

Na początku wyjaśnijmy oznaczenia:

- `X` - sygnał wejściowy,
- `Y'` - różnica do zakodowania,
- `Y` - zakodowana wartość,
- `E` - estymanta/wartość, jaka zostanie odkodowana.

DPCM bez predykcji

Jeżeli potrzebujecie jakieś wcześniejsze wartości początkowe (np. krok -1) zakładamy, że będą one równe 0 W każdym kroku algorytmu (dopóki mamy nowe próbki sygnału) będziemy odejmować od bieżącej wartości `X` wartość estymowaną `E`, otrzymując w ten sposób wstępną wartość różnicy `Y'`. Następnie na wartości `Y'` dokonujemy kwantyzacji (na przykład do wartości 8-bitowej), otrzymując naszą zakodowaną różnicę `Y`. Następnie dodajemy ją do naszej wartości estymowanej `E` w celu jej odświeżenia.

Dekompresja odbywa się w podobny sposób. Tu również zakładamy, że `X'[0]==Y[0]`. Następnie dla każdej kolejnej zakodowanej różnicy dodajemy wartość naszego odtworzonego sygnału z poprzedniego kroku. Należy jednak pamiętać, że nasz sygnał nie jest już zapisany na pierwotnym rozmiarze danych, nie tym po kwantyzacji!!!

Przedstawmy teraz kilka przypadków:

Zakładamy, że chcemy zakodować ciąg $X = [15, 16, 20, 14, 5, 10, 15, 13, 11, 7, 10, 11, 20, 1, 23]$. Załóżmy, że zmienna ta będzie zapisana w typie `int8` i będziemy stosować dla niej kwantyzacji do konkretniej ilości bitów (7 i 6), przy użyciu naszej standardowej funkcji do kwantyzacji. Poniżej dwa przykłady jak wartości jakie generuje zasza fukcja.:

- Wartości oryginalne: $X = [-10, -9, -8, -7, -6, -5, -4, -3, -2, \ldots, 6, 7, 8, 9, 10]$

Kontakt

Materiały

Przetwarzanie Obrazów

Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

- Wartości po kwantyzacji do 7 bitów: $X_7 = [-9, -9, -7, -7, -5, -5, -3, -3, -1, -1, 0, 0, 2, 2, 4, 4, 6, 6, 8, 8, 10]$
- Wartości po kwantyzacji do 6 bitów: $X_6 = [-10, -10, -6, -6, -6, -6, -2, -2, -2, -2, 1, 1, 1, 1, 5, 5, 5, 5, 9, 9, 9]$

W pierwszym przypadku wykorzystamy funkcję do *7-bitów*. Zacznijmy od dwóch pierwszych wartości. Kodujemy wartość 15, ponieważ w kroku zerowym nasze $e = 0$ nasza różnica będzie wynosić $Y' = 15$. Po kwantyzacji otrzymamy $Y = K_7(15) = 14$, za pomocą tego odświeżymy naszą estymowaną wartość na wyjściu:
$e = X' = Y + e = 16 + 0 = 16$. Przejdźmy do kodowania kolejnej wartości 16. Zaczynamy od wyliczenia różnicy $Y' = 16 - 14 = 2$, czyli po kwantyzacji również będziemy mieć $Y = K_7(2) = 2$. Teraz odświeżamy naszą wartość estymowaną $e = 2 + 16 = 16$. Reszta kroków została przedstawiona w tabeli.

Id	X	$X - e$	Y'		Y	$X' = Y + e$		e
0	15	15-0	15	$\xrightarrow{K_7}$	14.0	14.0+0	$\xrightarrow{e}$	14.0
1	16	16-14.0	2	$\xrightarrow{K_7}$	2.0	2.0+14.0	$\xrightarrow{e}$	16.0
2	20	20-16.0	4	$\xrightarrow{K_7}$	4.0	4.0+16.0	$\xrightarrow{e}$	20.0
3	14	14-20.0	-6	$\xrightarrow{K_7}$	-5.0	-5.0+20.0	$\xrightarrow{e}$	15.0
4	5	5-15.0	-10	$\xrightarrow{K_7}$	-9.0	-9.0+15.0	$\xrightarrow{e}$	6.0
5	10	10-6.0	4	$\xrightarrow{K_7}$	4.0	4.0+6.0	$\xrightarrow{e}$	10.0
6	15	15-10.0	5	$\xrightarrow{K_7}$	4.0	4.0+10.0	$\xrightarrow{e}$	14.0
7	13	13-14.0	-1	$\xrightarrow{K_7}$	-1.0	-1.0+14.0	$\xrightarrow{e}$	13.0
8	11	11-13.0	-2	$\xrightarrow{K_7}$	-1.0	-1.0+13.0	$\xrightarrow{e}$	12.0
9	7	7-12.0	-5	$\xrightarrow{K_7}$	-5.0	-5.0+12.0	$\xrightarrow{e}$	7.0
10	10	10-7.0	3	$\xrightarrow{K_7}$	2.0	2.0+7.0	$\xrightarrow{e}$	9.0
11	11	11-9.0	2	$\xrightarrow{K_7}$	2.0	2.0+9.0	$\xrightarrow{e}$	11.0
12	20	20-11.0	9	$\xrightarrow{K_7}$	8.0	8.0+11.0	$\xrightarrow{e}$	19.0
13	1	1-19.0	-18	$\xrightarrow{K_7}$	-17.0	-17.0+19.0	$\xrightarrow{e}$	2.0
14	23	23-2.0	21	$\xrightarrow{K_7}$	20.0	20.0+2.0	$\xrightarrow{e}$	22.0

Po całym kodowaniu dostajemy na wyjściu zakodowany łańcuch $Y = [14, 2, 4, -6, -10, 4, 5, 4, 6, 2, 2, -1, -1, -5, 2, 2, 8, -17, 20]$. Teraz zabieramy się dekodowanie tego łańcucha.

Kontakt

Materiały

Przetwarzanie Obrazów

Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

Zaczynamy od wartości 14, ponieważ nie było kroku -1-go zakładamy, że poprzednia wartość X' miała wartość 0, więc X' w tym kroku będzie wynosić $X' = 14 + 0 = 14$. Kolejna wartość to 2, więc nowa wartość to $X' = 2 + 14 = 16$. Kontynuacja obliczeń w tabeli.

Id	Y	$Y + X_{-1}$	X'
0	14	14+0	14
1	2	2+14	16
2	4	4+16	20
3	-5	-5+20	15
4	-9	-9+15	6
5	4	4+6	10
6	4	4+10	14
7	-1	-1+14	13
8	-1	-1+13	12
9	-5	-5+12	7
10	2	2+7	9
11	2	2+9	11
12	8	8+11	19
13	-17	-17+19	2
14	20	20+2	22

Na wyjściu otrzymaliśmy, więc wektor $X' = [14, 16, 20, 15, 6, 10, 14, 13, 12, 7, 9, 11, 19, 2, 22]$, który jest zbliżony wartościami do $X = [15, 16, 20, 14, 5, 10, 15, 13, 11, 7, 10, 11, 20, 1, 23]$.

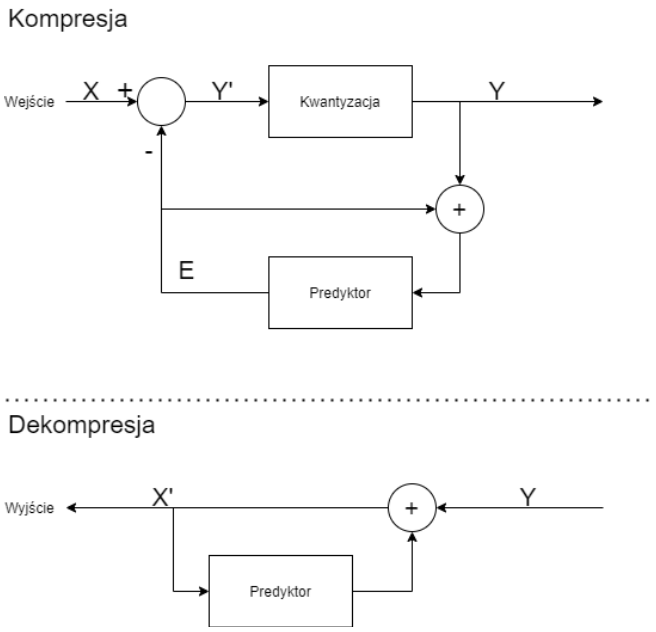
▸ Przykład dla kwantyzacji do 6 bitów

DPCM z predykcją

Trochę bardziej zaawansowana wersja DPCM zawiera predykcję, czyli wnioskowanie nie tylko na podstawie różnicy i poprzedniego elementu, ale również kilku poprzednich. Istnieją różnego rodzaju mniej lub bardziej zaawansowane predykatory. Najprostszy, jaki będziemy wykorzystywać to średnia ostatnich n elementów. Czyli zamiast odejmować ostatni estymant będziemy odejmować średnią np. 3 ostatnich sedymentów.

Ogólnie algorytm dla wyliczenia pojedynczego przejścia składa się z kilku kroków:

- Pobieramy wartość do zakodowania X_i
- Wyliczamy naszą wartość do zakodowania, czyli odejmujemy oczekiwaną wartość z poprzedniego kroku E_{i-1}, więc wykonujemy równanie $Y_i' = X_i - E_{i-1}$
- Dokonujemy kwantyzacji, czyli dostajemy na wyjście naszej kompresji, czyli Y
- Wyliczamy wartość, jaką otrzymamy na wyjściu n[illegible]
- Dokonujemy predykcji na podstawie naszych danych, czyli ostatnich [illegible] przykładowo licząc z nich średnią albo stosując algorytm predykcji liniowej.



Schemat uproszczonej wersji DPCM

Teraz przejdźmy do przykładów. Zaczniemy od naszego przypadku $X = [15, 16, 20, 14, 5, 10, 15, 13, 11, 7, 10, 11, 20, 1, 23]$ i predyktorem średnią z 3 elementów, a wartości będziemy poddawać kwantyzacji do *7-bitów*. Poniżej przykład bez rzutowania wartości estymowanej na liczbę całkowitą. Dalej również ten sam przykład, ale z zaokrągleniem.

- Przykład: DPCM z predykcją na podstawie średniej 3 ostatnich wartości z bez rzutowania jej na liczbę całkowitą
- Przykład: DPCM z predykcją na podstawie średniej 3 ostatnich wartości z zaokrągleniem jej do wartości całkowitej

Inne metody predykcji

1. Średnia geometryczna (*scipy.stats.mstats.gmean*) - może wymagać obudowania własną funkcją
2. Średnia harmoniczna (*scipy.stats.mstats.hmean*) - może wymagać obudowania własną funkcją
3. Mediana
4. Predykcja liniowa

$$
\begin{aligned}
p = 1 \quad &: \quad \hat{x}(n) = 1x(n-1) \\
p = 2 \quad &: \quad \hat{x}(n) = 2x(n-1) - 1x(n-2) \\
p = 3 \quad &: \quad \hat{x}(n) = 3x(n-1) - 3x(n-2) + 1x(n-3) \\
p = 4 \quad &: \quad \hat{x}(n) = 4x(n-1) - 6x(n-2) + 4x(n-3) - 1x(n-4)
\end{aligned}
$$

5. Regresja liniowa

- Przykład: DPCM z predykcją na podstawie predykcji liniowej dla n do 4 elementów

Przykładowy kod

1. DPCM bez predykcji

```
01. def DPCM_compress(x,bit):
02.     y=np.zeros(x.shape)
```

Kontakt

Materiały

Przetwarzanie Obrazów

Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

```
03.     e=0
04.     for i in range(0,x.shape[0]):
05.         y[i]=kwant(x[i]-e,bit)
06.         e+=y[i]
07.     return y
```

2. DPCM z predykcją

```
    01. def DPCM_compress(x,bit,predictor,n):
02.     y=np.zeros(x.shape)
03.     xp=np.zeros(x.shape)
04.     e=0
05.     for i in range(0,x.shape[0]):
06.         y[i]=kwant(x[i]-e,bit)
07.         xp[i]=y[i]+e
08.         idx=(np.arange(i-n,i,1,dtype=int)+1)
09.         idx=np.delete(idx,idx<0)
10.         e=predictor(xp[idx])
11.     return y
```

3. Predyktor bez predykcji:

```
    01. def no_pred(X):
02.     return X[-1]
```

Jak testować?

Jeżeli chodzi o testowanie to błędy najłatwiej zaobserwować na fragmencie sinusa:

```
    01. x=np.linspace(-1,1,1000)
02. y=0.9*np.sin(np.pi*x*4)
```

Kontakt

Materiały

Przetwarzanie Obrazów

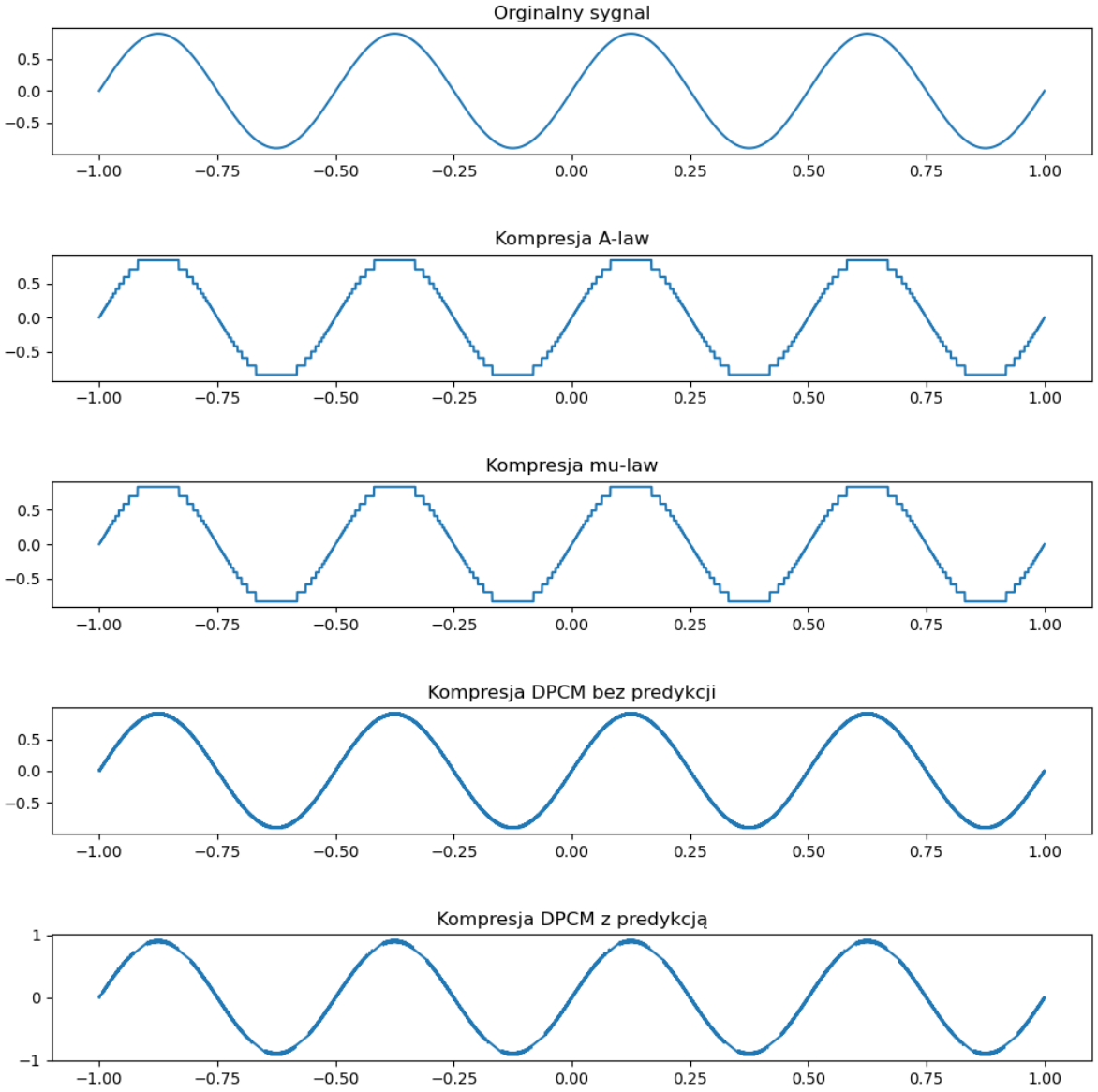
Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

Przyklad A kwnatyzacja do 6 bitów



Porównanie kompresji wersja A

Kontakt

Materiały

Przetwarzanie Obrazów

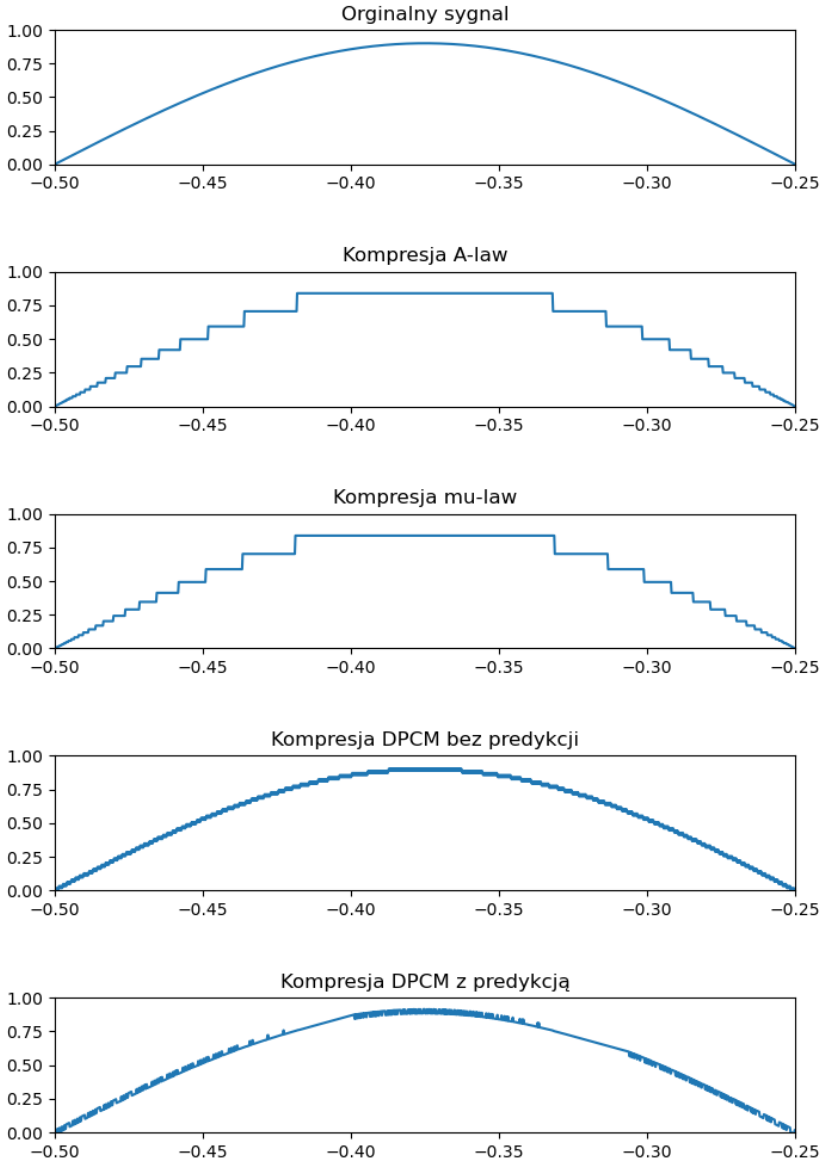
Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

Przyklad A kwnatyzacja do 6 bitów



Kontakt

Materiały

Przetwarzanie Obrazów

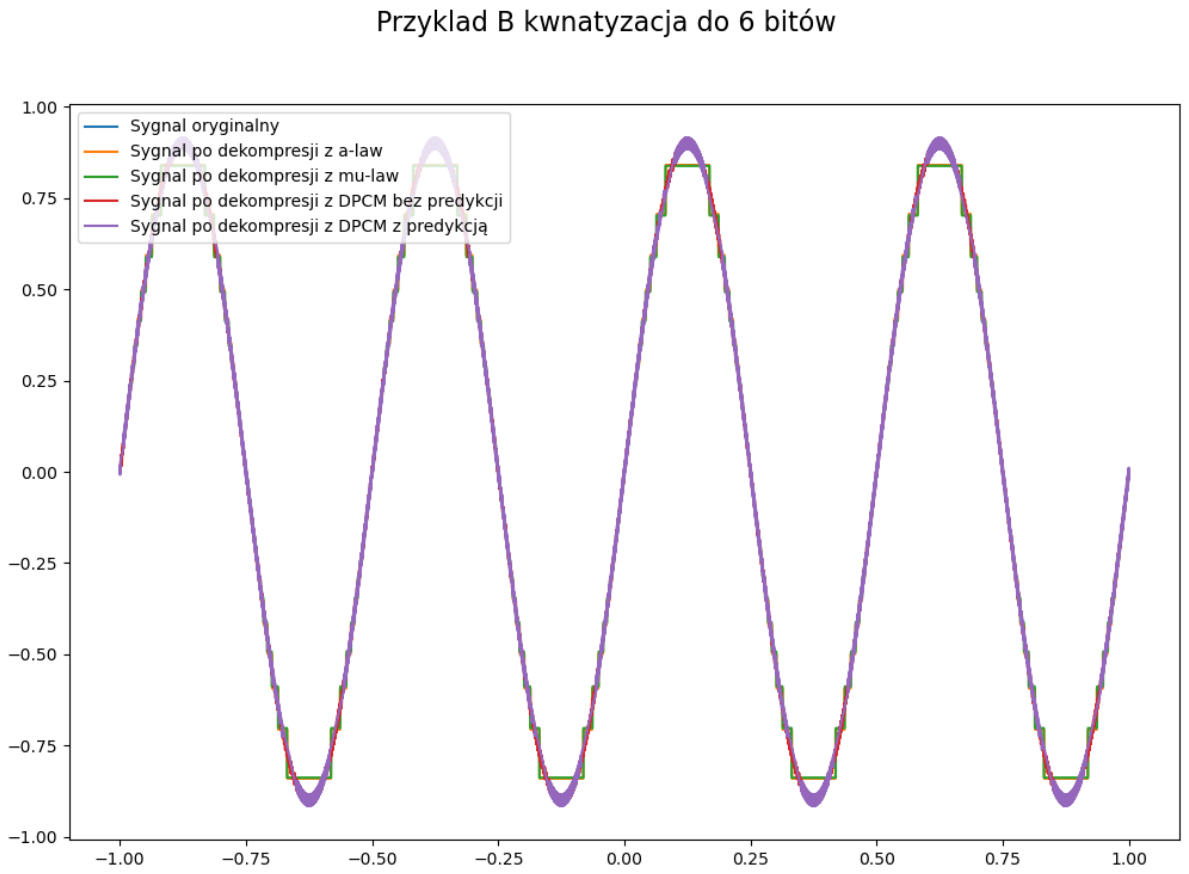
Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania

Porównanie kompresji wersja A



Porównanie kompresji wersja B

Kontakt

Materiały

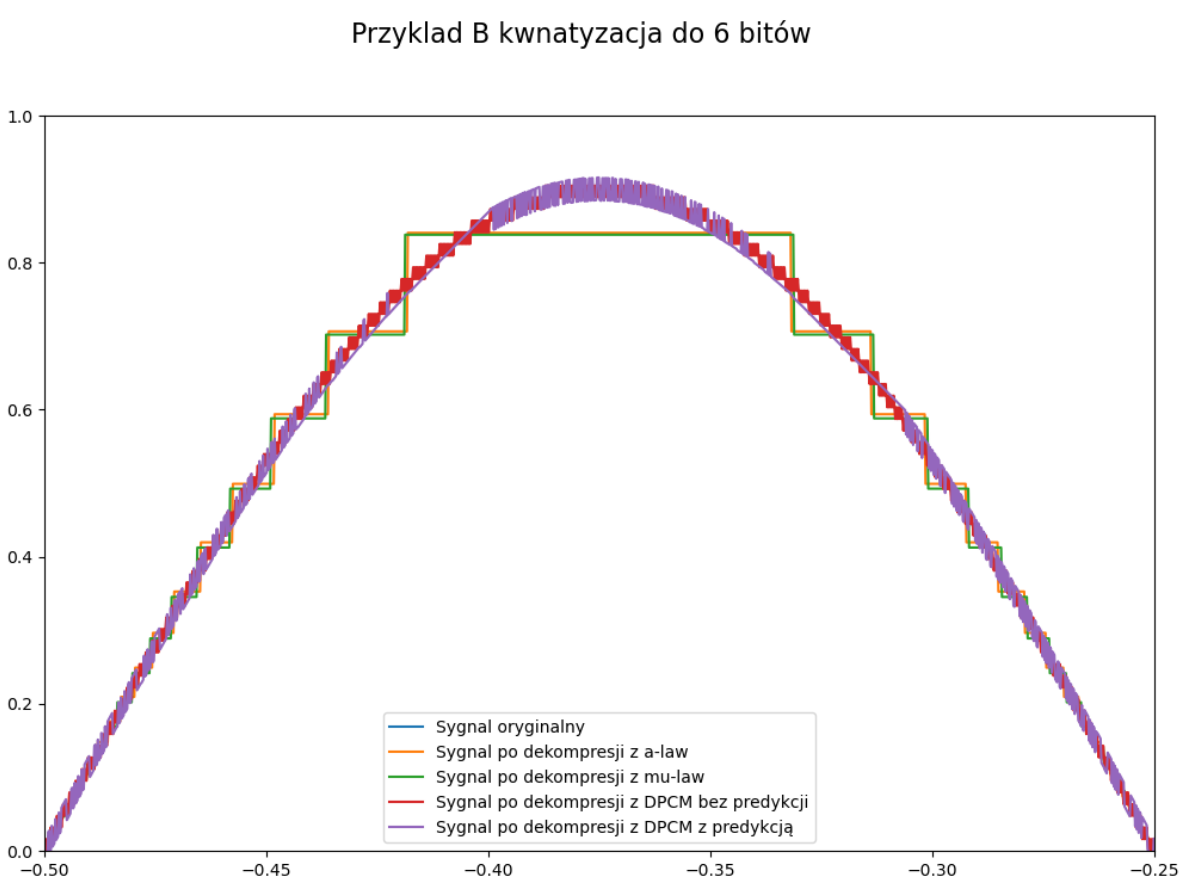
Przetwarzanie Obrazów

Systemy Multimedialne

Interaktywne Systemy Multimedialne

Przetwarzanie i analiza danych

FAQ - najczęściej zadawane pytania



Porównanie kompresji wersja B