Kompresja Bezstratna: RLE i ByteRun - Sprawozdanie

Table of Contents

Wprowadzenie

Celem laboratorium było zaimplementowanie i zbadanie skuteczności dwóch algorytmów kompresji bezstratnej: Run-Length Encoding (RLE) oraz ByteRun. Algorytmy zostały zaimplementowane w języku Python z wykorzystaniem biblioteki NumPy. Analizę przeprowadzono na trzech różnych typach obrazów: rysunku technicznym (JPEG), zeskanowanym dokumencie (PNG) oraz kolorowej fotografii (JPEG), oceniając stopień kompresji oraz weryfikując bezstratność procesu.

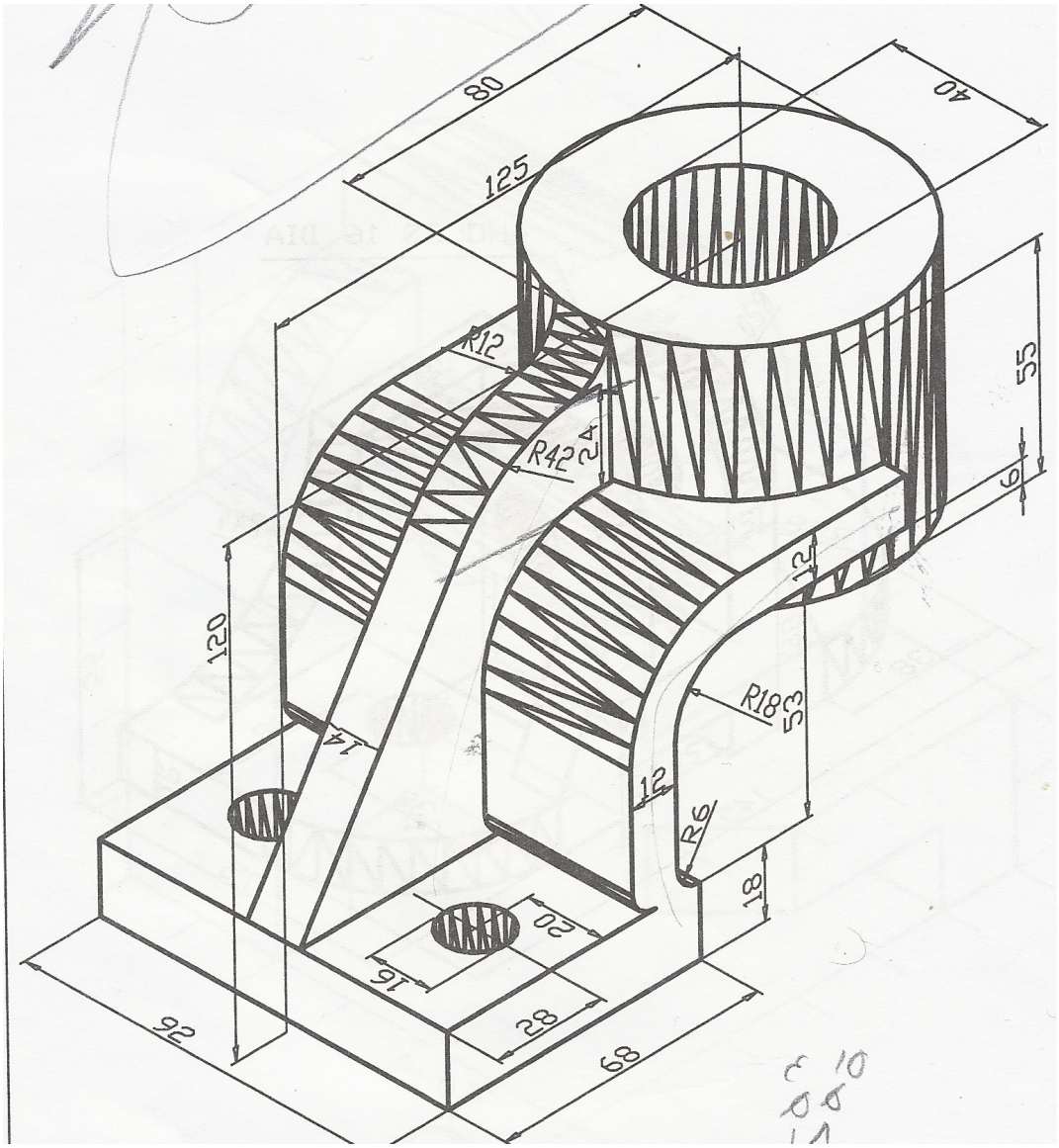
Zadania Zrealizowane

1. **Wybór Obrazów:** Wybrano trzy obrazy reprezentujące zadane kategorie (`technical.jpg`, `document.png`, `color.jpg`).
2. **Implementacja Algorytmów:** Zaimplementowano funkcje RLE i ByteRun (koder i dekoder) w Pythonie, wraz z mechanizmem zapisu/odczytu metadanych (kształt, typ danych).
3. **Raport i Analiza:** Przeprowadzono kompresję/dekompresję, zweryfikowano bezstratność, obliczono metryki CR i PR, opisano sposób działania metadanych i sformułowano wnioski na podstawie uzyskanych wyników.

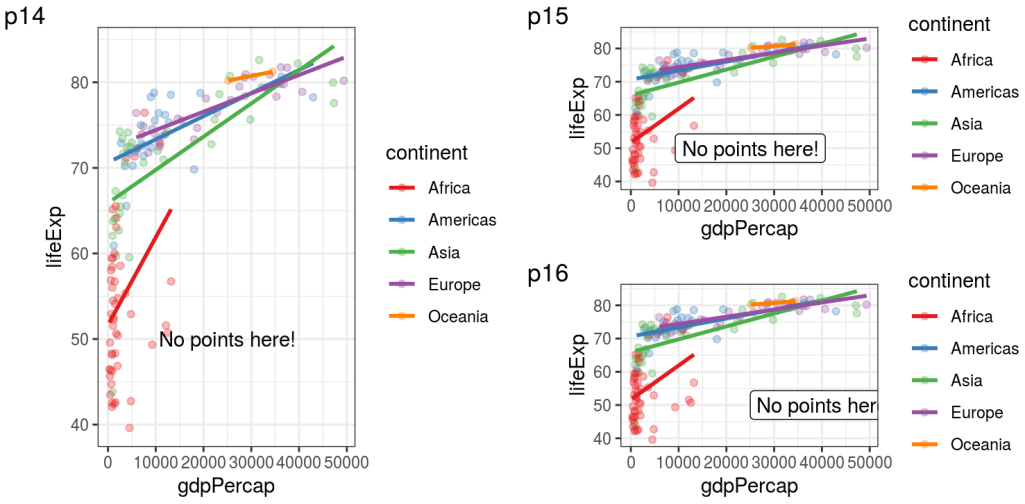
Wyniki Eksperymentalne

Obrazy Testowe

Poniżej przedstawiono obrazy wykorzystane do testów kompresji. Ścieżki odnoszą się do lokalizacji względem pliku sprawozdania.



Rysunek Techniczny (obrazy/technical.jpg)



Dokument (obrazy/document.png)

Zdjęcie Kolorowe (`obrazy/color.jpg`)

Implementacja i Przechowywanie Metadanych

Kluczowym elementem kompresji bezstratnej, szczególnie dla danych wielowymiarowych jak obrazy, jest zachowanie informacji o oryginalnej strukturze. W obu zaimplementowanych algorytmach (RLE i ByteRun) przed właściwymi skompresowanymi danymi umieszczane są metadane dotyczące kształtu i typu danych oryginalnej tablicy NumPy.

Wykorzystano następującą strategię: 1. Na początku skompresowanego wektora (tablicy NumPy) umieszczany jest nagłówek metadanych. 2. Pierwsza liczba w nagłówku (`metadata[0]`) określa liczbę wymiarów oryginalnej tablicy (`N = len(data.shape)`). 3. Druga liczba (`metadata[1]`) przechowuje numer typu danych (`data.dtype.num`). 4. Kolejne `N` liczb (`metadata[2]` do `metadata[N+1]`) przechowują rozmiary poszczególnych wymiarów (`data.shape[0]`, `data.shape[1]`, ...). 5. Do zapisu nagłówka użyto

typu `np.int64`, a typ danych jest odtwarzany za pomocą mapowania numeru typu na obiekt `np.dtype`. Właściwe dane RLE również zapisano jako `np.int64` (dla dużych liczników), a dane ByteRun jako `np.int16` (dla bajtów sterujących i danych). 6. Przed kompresją dane obrazu są spłaszczane do jednowymiarowego wektora (`data.flatten()`). 7. Po zdekompresowaniu danych z wektora wynikowego, jest on przekształcany z powrotem do oryginalnego kształtu i typu za pomocą `decoded_array.reshape(original_shape).astype(original_dtype)`.

Weryfikacja Poprawności Kodowania

Poprawność działania zaimplementowanych algorytmów została zweryfikowana dla każdego obrazu testowego i obu metod (RLE, ByteRun). Po wykonaniu kompresji i dekompresji, oryginalna tablica NumPy została porównana z tablicą uzyskaną po dekompresji za pomocą funkcji `np.array_equal()`.

We wszystkich przeprowadzonych testach porównanie zwróciło wynik `True`, co potwierdza, że zaimplementowane metody kompresji i dekompresji są **bezstratne**.

Wyniki Skuteczności Kompresji

Skuteczność kompresji dla poszczególnych obrazów i metod przedstawiono w tabeli poniżej. Zastosowano dwie metryki:

- **Stopień Kompresji (CR):** `Rozmiar_Przed / Rozmiar_Po`. Wartość > 1 oznacza kompresję, < 1 oznacza ekspansję danych.
- **Procent Pozostałych Danych (PR):** `(Rozmiar_Po / Rozmiar_Przed) * 100`. Mniejsza wartość oznacza lepszą kompresję.

Rozmiary podano w bajtach (`.nbytes` tablicy NumPy).

Obraz (Kategoria, Plik)	Algorytm	Rozmiar Oryginalny (B)	Rozmiar Skompresowany (B)	Stopień Kompresji (CR)	Pozostało (PR %)
Technical (`technical.jpg`)	RLE	7629741	80039512	0.0953	1049.05 %
Technical (`technical.jpg`)	ByteRun	7629741	14788512	0.5159	193.83 %
Document (`document.png`)	RLE	3538944	3923592	0.9020	110.87 %
Document (`document.png`)	ByteRun	3538944	741594	4.7721	20.96 %
Photo (`color.jpg`)	RLE	1440000	22353144	0.0644	1552.30 %
Photo (`color.jpg`)	ByteRun	1440000	2961006	0.4863	205.63 %

Podsumowanie / Wnioski

Analiza wyników liczbowych i weryfikacja bezstratności pozwalają na sformułowanie następujących wniosków:

1. **Skuteczność RLE:** Algorytm RLE okazał się **bardzo nieskuteczny** dla wszystkich testowanych obrazów, prowadząc we wszystkich przypadkach do **znacznej ekspansji danych** (PR > 100%, CR < 1). Największa ekspansja (PR ~1049% i ~1552%) wystąpiła dla obrazu technicznego (JPG) i kolorowej fotografii (JPG), a mniejsza, ale nadal znacząca (PR ~111%), dla dokumentu (PNG). Przyczyną jest brak bardzo długich serii identycznych wartości pikseli w tych typach obrazów. Nawet w rysunku technicznym, zapis w formacie JPG wprowadza artefakty zakłócające jednolitość. W obrazach naturalnych i skanach szum oraz drobne różnice w kolorach/odcieniach również przerywają serie. Narzut związany z zapisywaniem pary (licznik, wartość) dla krótkich serii (nawet długości 1) znacząco przewyższa rozmiar oryginalnych danych.
2. **Skuteczność ByteRun:** Algorytm ByteRun wykazał **zróżnicowaną skuteczność**. Osiągnął **bardzo dobrą kompresję** (CR ~4.77, PR ~21%) dla obrazu dokumentu (PNG). Świadczy to o obecności w tym pliku zarówno długich serii powtórzeń (prawdopodobnie białe tło), które ByteRun efektywnie koduje, jak i obszarów zmiennych (tekst, linie), które są kodowane jako sekwencje literalne bez dużego narzutu. Dla obrazu technicznego i fotografii (obu JPG) ByteRun, podobnie jak RLE, spowodował **ekspansję danych** (PR ~194% i ~206%), jednak **wielokrotnie mniejszą** niż RLE. Zdolność do kodowania sekwencji literalnych zapobiega katastrofalnej ekspansji obserwowanej w RLE dla danych bez długich powtórzeń. Mimo to, dla tych złożonych obrazów narzut związany z bajtami sterującymi i segmentacją danych nadal prowadził do powiększenia plików.
3. **Wybór Algorytmu i Charakter Danych:** Wyniki potwierdzają, że skuteczność algorytmów kompresji strumieniowej silnie zależy od **charakterystyki danych**. RLE nadaje się tylko do specyficznych danych z bardzo długimi seriami powtórzeń (np. proste mapy bitowe, nieskompresowane dane medyczne). ByteRun jest znacznie bardziej **uniwersalny i odporny** na zmienność danych, efektywnie kompresując dane mieszane (jak dokumenty) i ograniczając negatywne skutki (ekspansję) dla danych "trudnych" (jak zdjęcia). Dla testowanego zestawu obrazów, ByteRun okazał się zdecydowanie lepszym algorytmem, zwłaszcza dla dokumentu PNG.
4. **Bezstratność i Metadane:** Poprawna implementacja obu algorytmów zapewnia bezstratność, co zostało potwierdzone. Mechanizm zapisywania metadanych (kształt, typ danych) jest niezbędny do poprawnego odtworzenia oryginalnej struktury danych.