Modelowanie matematyczne i symulacje komputerowe

Projekt 1

CZĘŚĆ II

Mapy chaotyczne to podtypy nieliniowych układów dynamicznych o charakterze deterministycznym, których rozwiązania mają zachowanie podobne do sygnału szumu. Wysoka wrażliwość mapy chaotycznej na wartość początkową jest jedną z jej najważniejszych cech - *efekt motyla*. Efekt motyla oznacza, że dokonanie niewielkiej zmiany w wartości początkowej, skutkuje gwałtowną zmianą w wyniku. Własność ta daje wysoką skuteczność i nieprzewidywalność mapy chaotycznej w zastosowaniach, między innymi, z punktu widzenia bezpieczeństwa w sieci internetowej. Mapy chaotyczne są szeroko stosowane do kodowania informacji przesyłanych przez sieć, np. kodowania obrazów, czym się Państwo zajmą w drugiej części projektu pierwszego.

Kodowania obrazów z wykorzystaniem map chaotycznych można dokonywać na wiele różnych sposobów. To zagadnienie wciąż intensywnie się rozwija i coraz więcej znajdujemy w literaturze proponowanych algorytmów kodowania, które uzyskują coraz lepsze wyniki w przeprowadzanych testach.

W projekcie wykorzystają Państwo mapę chaotyczną do zakodowania obrazka `papugi_res.png` (dostępny na stronie kursu w materiałach do projektu) poprzez tasowanie pozycji intensywności kolorów. Każdy kolorowy obraz to $N \cdot M \cdot 3$ pikseli, czyli `wysokość·szerokość·kanały`, obraz jest przechowywany jako tablica wymiaru $N \times M \times 3$. Oznacza to, że każdy piksel przechowuje własne intensywności RGB. W opisie zadania podane są wskazówki wykonania projektu w `R`.

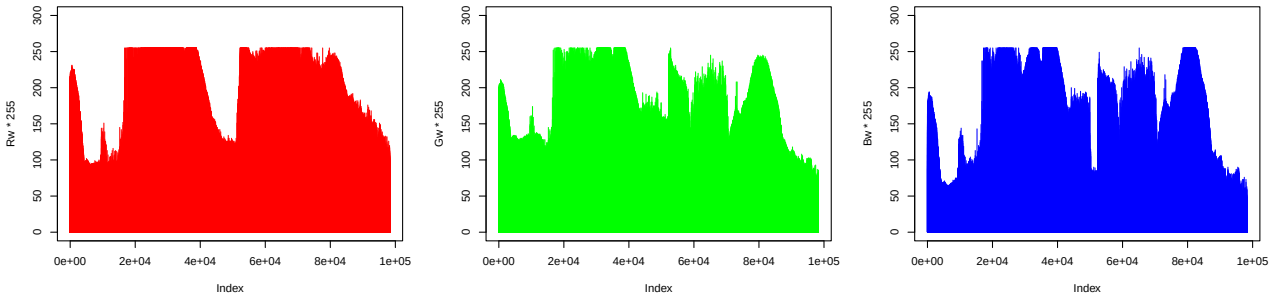
Algorytm postępowania można znaleźć w artykule [1].

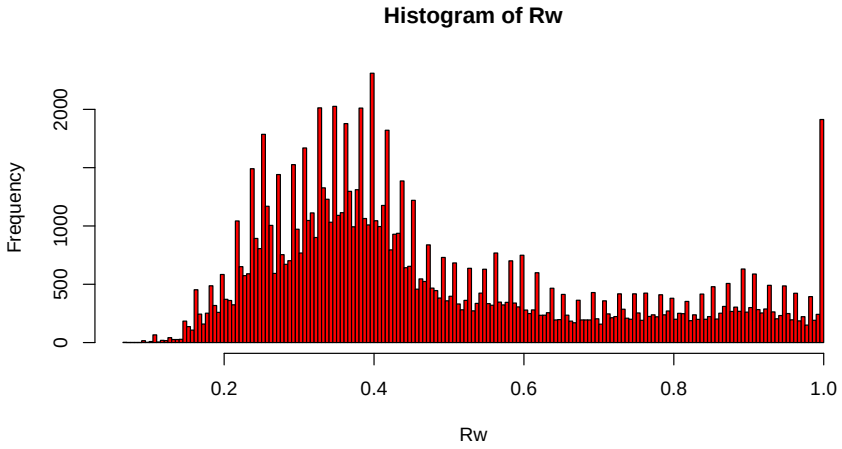
Kodowanie i dekodowanie obrazów:

1. Zapisujemy i ładujemy obrazek (można wykorzystać funkcję `readImage()` z pakietu `OpenImageR`, np. `img = readImage("papugi_res.png")`).

2. Wyodrębniamy trzy macierze $\{R, G, B\}$ (np. przy wykorzystaniu `readImage()` macierz intensywności koloru czerwonego dla każdego piksela dostajemy za pomocą `img[,,1]`). Każdą macierz $\{R, G, B\}$ zapisujemy w postaci wektora o długości $N \cdot M$, co daje $\{Rw, Gw, Bw\}$.

3. Dla każdego koloru oryginalnego obrazka możemy zobrazować intensywność (w formie słupkowej, przykład dla wszystkich kolorów) oraz przedstawić histogramy (przykład dla koloru czerwonego).





4. Wykorzystując mapę logistyczną

$$x_{n+1} = rx_n(1 - x_n), \quad n = 0, \ldots, K,$$

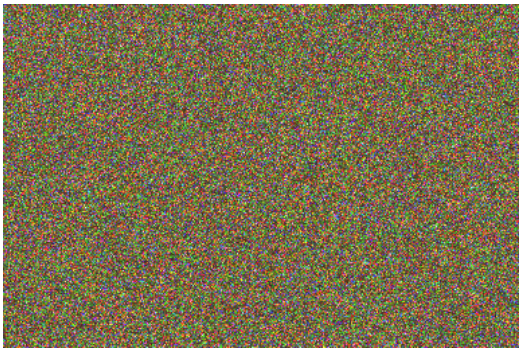
dla wybranego $r \in [3.6, 4]$ i $x_0 \in (0, 1)$ konstruujemy wektor wartości o długości $K = 3 \cdot N \cdot M$. Otrzymany wektor dzielimy na trzy podwektory $\{x_R, x_G, x_B\}$. Klucz kodowania składa się z dwóch informacji

$$(x_0, r).$$

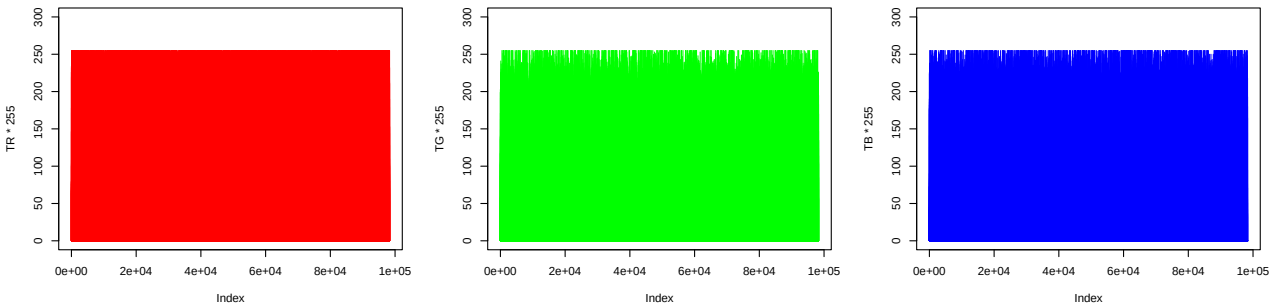
5. Sortujemy (malejąco lub rosnąco) osobno wektory $\{x_R, x_G, x_B\}$ otrzymując $\{x_R^s, x_G^s, x_B^s\}$. Następnie tworzymy trzy wektory pozycji $\{P_R, P_G, P_B\}$, które przechowują numery pozycji elementów z $\{x_R^s, x_G^s, x_B^s\}$ w wektorach $\{x_R, x_G, x_B\}$.

7. (Kodowanie) Dokonujemy tasowania wartości wektorów $\{Rw, Gw, Bw\}$ za pomocą wektorów pozycji $\{P_R, P_G, P_B\}$, tzn. tworzymy nowe wektory $\{T_R, T_G, T_B\}$, do których wstawiamy wartości z $\{Rw, Gw, Bw\}$ w następujący sposób: bierzemy element z $i-$tej pozycji wektora Rw, czyli $Rw[i]$, szukamy na jakiej pozycji stoi i w wektorze P_R (załóżmy, że na j), wówczas na pozycji j wektora T_R, czyli $T_R[j]$ wstawiamy $Rw[i]$.

8. Łączymy wektory kolorów $\{T_R, T_G, T_B\}$ w tablicę (funkcja `array()`) o wymiarze $N \times M \times 3$. Zakodowany obrazek wygląda tak:



9. Wykreślamy intensywność każdego koloru zakodowanego obrazka



10. (Dekodowanie) Odbiorca mając klucz kodowania (x_0, r) może stworzyć wektory pozycji $\{P_R, P_G, P_B\}$, które pozwolą poustawiać w nowych wektorach (odkodowane kolory) $\{D_R, D_G, D_B\}$ wartości z $\{T_R, T_G, T_B\}$. Następnie łączymy $\{D_R, D_G, D_B\}$ w tablicę uzyskując odkodowany obraz:

11. Wyznaczamy współczynnik korelacji między poszczególnymi kolorami oryginalnego obrazu a kolorami zakodowanego obrazu, np. dla koloru czerwonego mamy

$$cor_R = \frac{\sum_{i=1}^{N\cdot M}\left(Rw_i - \overline{Rw}\right)\left((T_R)_i - \overline{T_R}\right)}{\sqrt{\sum_{i=1}^{N\cdot M}\left(Rw_i - \overline{Rw}\right)^2}\sqrt{\sum_{i=1}^{N\cdot M}\left((T_R)_i - \overline{T_R}\right)^2}}$$

12. Sprawdzamy wrażliwość kodowania względem doboru klucza, przykładowo dokonujemy kodowania obrazka przy kluczu $(x_0 = 0.1, r = 3.8)$, następnie próbujemy odkodować obrazek przy użyciu klucza $(x_0 = 0.1000000001, r = 3.8)$ oraz $(x_0 = 0.1, r = 3.800000001)$.

Literatura

[1] M. Prasad, K.L. Sudha, *Chaos Image Encryption using Pixel shuffling* (artykuł umieszczony na stronie kursu w materiałach do projektu).