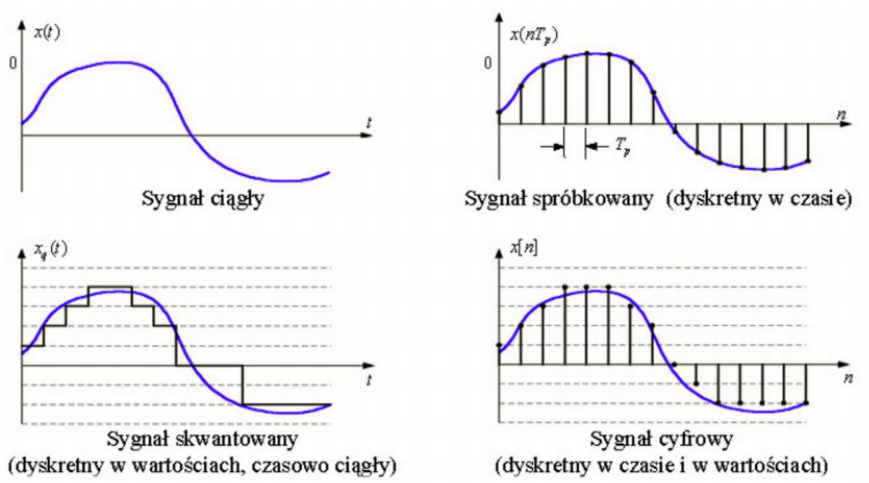
Laboratorium 4 – dyskretyzacja, kwantyzacja, binaryzacja

Streszczenie

Komputery przetwarzają i przechowują wszystkie sygnały w postaci cyfrowej. Ważnym elementem pracy z sygnałami (m.in. dźwięk, obraz) jest przetworzenie sygnału analogowego (ciągłego) na cyfrowy (dyskretny). Konwersja analogowo-cyfrowa składa się z kilku ważnych elementów takich jak dyskretyzacji (zwanej też próbkowaniem) oraz kwantyzacji (nadaniu wartości ze zbioru). Przebieg procesu przedstawia poniższy rysunek.



Rys. 1. Konwersja analogowo-cyfrowa.

1 CEL

Celem laboratorium jest poznanie zagadnień związanych z konwersją analogowo-cyfrową, oraz problemów które może ona dostarczyć. Dodatkowo poruszony jest temat binaryzacji, jako „uproszczonej kwantyzacji" do dwóch wartości [0, 1] na rzeczywistym przykładzie segmentacji obiektu z tła na obrazie.

2 Dyskretyzacja

1. Przygotuj funkcję generującą zdyskretyzowany sygnał sinus na podstawie parametrów:
   - Częstotliwość sygnału f
   - Częstotliwość próbkowania F_s
2. Funkcja powinna zwracać tablice:
   - t - tablicę "czasu" (zakres od 0 do 1 z krokiem równym częstości próbkowania: $dt = \frac{1}{F_s}$)
   - s - wartości wygenerowanego sygnału sinus: $s(t) = \sin(2\pi f t)$
3. Wyświetl wykresy zawierające sygnał o częstotliwości 10Hz za pomocą próbkowania:
   - 20Hz
   - 21Hz
   - 30Hz
   - 45Hz
   - 50Hz
   - 100Hz
   - 150Hz
   - 200Hz
   - 250Hz
   - 1000Hz
4. Czy istnieje twierdzenie, które określa z jaką częstotliwością należy próbkować, aby móc wiernie odtworzyć sygnał? Jak się nazywa?
5. Jak nazywa się zjawisko, które z powodu błędnie dobranej częstotliwości próbkowania powoduje błędną interpretację sygnału?
6. Znajdź za pomocą internetu obraz, na którym występuje wskazane zjawisko (przy wyświetlaniu go w znacznie mniejszej rozdzielczości).
   - Obrazek oryginalny powinien być wysokiej rozdzielczości
   - Plik powienien być zapisany w formacie png
7. Sprawdź jak python radzi sobie z wyświetleniem takiego obrazu i jakie opcje można przy tym użyć (`https://matplotlib.org/3.1.1/gallery/images_contours_and_fields/interpolation_methods.html`)

3 Kwantyzacja

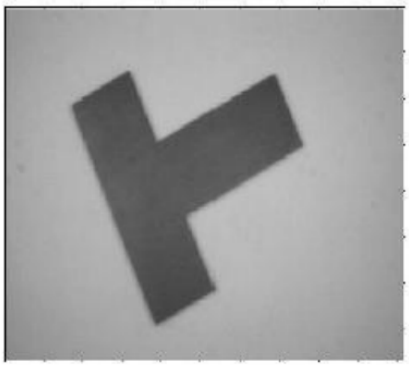
1. Wczytaj wcześniej znaleziony obrazek
2. Wykonaj polecenie, które zwróci ile wymiarów ma wczytana macierz (obrazek)
3. Wykonaj polecenie, które zwróci iloma wartościami jest opisywany pojedynczy piksel (inaczej: z ilu wartości składa się najgłębszy wymiar)
4. Przekształć obraz do skali szarości za pomocą 3 różnych metod (zapisz jako 3 różne macierze):
   - Wyznaczenie jasności piksela: $(max(R, G, B) + min(R, G, B))/2$
   - Uśrednienie wartości piksela: $(R + G + B)/3$
   - Wyznaczenie luminancji piksela: $0.21R + 0.72G + 0.07B$
5. Wygeneruj histogram dla każdego z otrzymanych „szarych" obrazów (funkcja histogram z pakietu numpy)
6. Dla dowolnego z wygenerowanych obrazów, za pomocą parametru bins zredukuj liczbę kolorów na histogramie do 16 i wyświetl zakresy nowych kolorów
7. Stwórz kolejną macierz (obrazek) ze zredukowaną liczbą kolorów (jako nową wartość piksela przyjmnij środek przedziału zwróconego przez funkcję histogramu)
8. Wyświetl wszystkie obrazy i ich histogramy

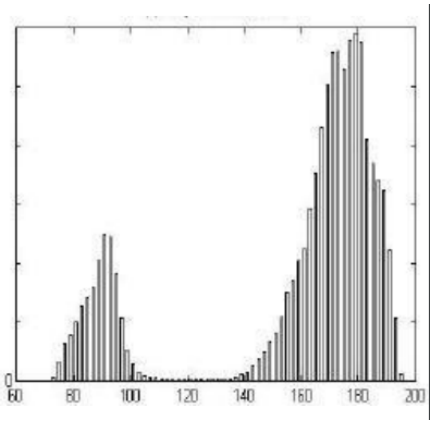
4 Binaryzacja

1. Pobierz z internetu obraz przedstawiający obiekt na niejednorodnym tle (np. gradiencie)

2. Wczytaj obraz, zamień go na skalę szarości za pomocą jednej z wcześniej stosowanych metod i wygeneruj histogram

3. Napisz funkcję, która na podstawie histogramu określi punkt progowania (zazwyczaj lokalne minimum między dwoma 'klasami' kolorów, tak jak na poniższym obrazku')





4. Zbinaryzuj (ustaw wartości na 0 lub 1) wartości pikseli obrazka zgodnie z otrzymanym progiem

5. Wyświetl obraz z wysegmentowanym obiektem (segmentacja, czyli usunięcie niepotrzebnych elementów, jak np. tła i uwypukleniu tych ważniejszych)