Politechnika Wrocławska

Inżynieria Obrazów

Laboratorium nr 3

Programowe przetwarzanie obrazów

Szymon Datko & Mateusz Gniewkowski

szymon.datko@pwr.edu.pl , mateusz.gniewkowski@pwr.edu.pl

Wydział Elektroniki,

Politechnika Wrocławska

semestr letni 2020/2021

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

Cel ćwiczenia

1. Zapoznanie się z biblioteką OpenCV.
2. Nauczenie się w jaki sposób programowo przekształcać obrazy.
3. Zapoznanie się z podstawowymi modelami barw.

Politechnika Wrocławska

Wprowadzenie do tematu

Omówienie zagadnień

Politechnika Wrocławska

Instrukcja do zajęć

- Omówienie zagadnień znajduje się w pliku `instrukcja.ipynb`.
- Zachęcam do własnego eksperymentowania z zawartymi tam przykładami.
- Zawarte w instrukcji fragmenty kodów można użyć w rozwiązaniach.
- Proszę **nie zamieszczać** pliku `instrukcja.ipynb` w końcowym archiwum z rozwiązaniami, przesyłanymi do systemu ePortal!
  - Na koniec pracy, oszczędzając miejsce, proszę po prostu ten plik usunąć!
  - Oczywiście uważając, żeby się nie pomylić i nie usunąć rozwiązań... ;-)

Politechnika Wrocławska

Przygotowanie do zajęć

Skonfigurowanie środowiska

Politechnika Wrocławska

Rozpoczęcie pracy (1/2)

- Pobrać i rozpakować archiwum z plikami do zajęć ze strony:
  https://datko.pl/IOb/
- Wejść do pobranego katalogu z plikami źródłowymi.
- Stworzyć wirtualne środowisko:
  `python3 -m venv venv/`
- Aktywować wirtualne środowisko:
  - w systemie Linux:
    `. venv/bin/activate`
  - w systemie Windows:
    `venv/Scripts/activate`
- Zaktualizować narzędzia budujące wewnątrz wirtualnego środowiska:
  `pip3 install --upgrade pip setuptools wheel`
- Zainstalować potrzebne zależności:
  `pip3 install -r requirements.txt`

Rozpoczęcie pracy (2/2)

- Uruchomić serwer aplikacji **Jupyter**:
  `jupyter notebook`
- Zaczekać na załadowanie się przeglądarki internetowej.
- W aplikacji **Jupyter** załadować plik `rozwiązania.ipynb`.
- Zrealizować zadania w przygotowanych komórkach aplikacji **Jupyter**.
- Jeśli przygotowane rozwiązanie wymaga dodatkowych bibliotek, to należy koniecznie dopisać je w pliku `requirements.txt`.
- Wszelkie pliki pomocnicze powinny być w katalogu projektu.
  - Należy odwoływać się do nich przy pomocy ścieżek względnych.
- Dla pewności, że wszystko działa, przebudować wszystkie komórki.
  - W aplikacji **Jupyter** z menu `CELL` wybieramy pozycję `run all`.
- Na sam koniec pracy proszę pamiętać, aby w archiwum do zamieszczenia w systemie ePortal **nie dołączać** katalogu `venv/` – czyli usunąć go!

Politechnika Wrocławska

Koniec wprowadzenia

Zadania do wykonania...

Zadania do wykonania (1)

Na ocenę **3.0** należy wykonać filtr górnoprzepustowy – tzw. detektor krawędzi.

Wskazówki:

- zastosować poniższą maskę dla dowolnego obrazka,

$$h(k,l) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix},$$

- uwaga – w przypadku tego filtru nie należy uśredniać macierzy (czyli nie należy dzielić przez sumę wartości).

Zadania do wykonania (2)

Na ocenę **3.5** należy dokonać prostego przekształcenia kolorów obrazu.

Wskazówki:

- jako wejście można wybrać dowolny obrazek,
- skonwertować wartości do formatu zmiennoprzecinkowego (*float*),
  - zamiast wartości 0..255 będziemy pracować na wartościach 0.0..1.0,
- zrealizować przeliczenie wartości według poniższego wzoru,

$$\begin{bmatrix} R_{new} \\ G_{new} \\ B_{new} \end{bmatrix} = \begin{bmatrix} 0.393 & 0.769 & 0.189 \\ 0.349 & 0.689 & 0.168 \\ 0.272 & 0.534 & 0.131 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix},$$

- jeśli któraś z nowych wartości przekroczy **1.0**, należy ją przyciąć do **1.0**.

Zadania do wykonania (3)

Na ocenę **4.0** należy skonwertować dowolny obrazek do modelu barw YCbCr.

Wskazówki:

- konwersja wartości powinna nastąpić według poniższego wzoru (YCrCb),

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.229 & 0.587 & 0.114 \\ 0.500 & -0.418 & -0.082 \\ -0.168 & -0.331 & 0.500 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix},$$

  - uwaga na odwróconą kolejność składowych Cr i Cb w powyższym wzorze,
  - zastosowaną ją, aby zadziałała konwersja odwrotna z biblioteki OpenCV,
- tym razem operujemy na wartościach obrazu typu *uint8*, czyli 0..255,
  - proszę pamiętać o przycięciu uzyskanych wartości do tego przedziału,
- konwersję odwrotną będzie można później zrealizować za pomocą funkcji,

```
imageRGB = cv.cvtColor(imageYCrCb, cv.COLOR_YCrCb2RGB)
```

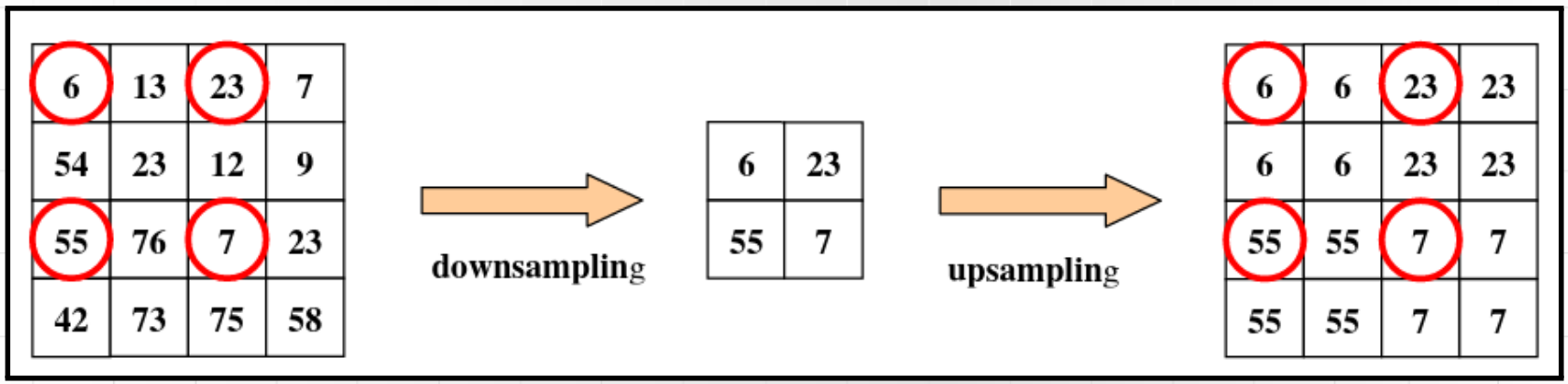
- wyświetlić oryginalny obraz (dla porównania) przed konwersją,
- wyświetlić obliczone składowe `Y`, `Cb`, `Cr` w odcieniach szarości.

Zadania do wykonania (4)

Na ocenę **4.5** należy wykonać symulację transmisji obrazu w systemie DVB.

Wskazówki:

- kroki do wykonania są następujące:
  1. przeprowadzić konwersję obrazu z modelu RGB do YCbCr (zadanie 3),
  2. zrealizować operację downsamplingu na kanałach `Cb` i `Cr`,
  3. przeprowadzić operację upsamplingu na macierzach kanałów `Cb` i `Cr`,
  4. złożyć obraz z otrzymanych wartości `Cb` i `Cr` oraz oryginalnej macierzy `Y`,
  5. wyświetlić otrzymany obraz (RGB) i poszczególne nowe składowe (YCbCr);
- punkty 1. i 2. imitują działanie kodera, zaś kroki 3. i 4. to dekoder,
- zmniejszenie rozmiaru macierzy `Cb` i `Cr` pozwala na zaoszczędzenie pasma w transmisji, bez znaczącego wpływu na jakość obrazu.



Zadania do wykonania (5)

Na ocenę **5.0** należy obliczyć różnicę między obrazkami z poprzednich zadań.

Wskazówki:

- wykorzystać obraz wejściowy z zadania 3 i wynikowy z zadania 4 (RGB),
- obliczyć tak zwany błąd średniokwadratowy (ang. *Mean Square Error*),

$$MSE = \frac{1}{m} \cdot \frac{1}{n} \cdot \sum_{i=1}^{3} \sum_{j=1}^{n} \left(X_{ij} - \hat{X}_{ij} \right)^2,$$

- gdzie:
  - n – liczba pikseli obrazu,
  - m – liczba kanałów (trzy w modelu RGB),
  - X_{ij} – wartość j-tego koloru i-tego piksela w obrazie wejściowym,
  - $\hat{X}_{ij}$ – wartość j-tego koloru i-tego piksela w obrazie wyjściowym;
- można też porównać jak wypada ten sam obraz w formacie JPEG i PNG.