

Politechnika Warszawska

Cyfrowe przetwarzanie obrazu

Projekt – zestaw 4

Wykonał:
Bartłomiej Guś,
nr albumu 297415

Warszawa 2021/2022

Spis treści

| | | |
|-------|--|----|
| 1. | Wstęp | 3 |
| 2. | Projekt..... | 3 |
| 2.1. | Ścieżka przetwarzania/problems i ich rozwiązania | 3 |
| 2.2. | Schemat blokowy ścieżki przetwarzania..... | 5 |
| 3. | Wyniki | 6 |
| 3.1. | Zdjęcie 1 | 6 |
| 3.2. | Zdjęcie 1_blur..... | 7 |
| 3.3. | Zdjęcie 1_gradient..... | 9 |
| 3.4. | Zdjęcie 1_sp | 10 |
| 3.5. | Zdjęcie 2 | 12 |
| 3.6. | Zdjęcie 2_blur..... | 13 |
| 3.7. | Zdjęcie 2_gradient..... | 15 |
| 3.8. | Zdjęcie 2_sp | 16 |
| 3.9. | Zdjęcie 3 | 18 |
| 3.10. | Zdjęcie 3_blur..... | 19 |
| 3.11. | Zdjęcie 3_gradient..... | 21 |
| 3.12. | Zdjęcie 3_sp..... | 22 |

1. Wstęp

Celem projektu było stworzenie programu w moim przypadku w języku Python, który miał posłużyć do wykrywania i rozpoznawania kart UNO. Projekt obejmował wykrycie kart z trzech scen w czterech różnych wersjach: oryginalnej, z blurem, z nałożonym gradientem oraz z zakłóceniami typu sól i pieprz. Na każdym zdjęciu były widoczne cztery karty. W projekcie zastosowano jedną ścieżkę przetwarzania do każdego zdjęcia.

2. Projekt

2.1. Ścieżka przetwarzania/problems i ich rozwiązania

Początkowo wczytane zdjęcie, było poddawane działaniu filtrowi Median Blur w celu przygotowania zdjęcia przed dalszym przetwarzaniem: za pomocą niego mogłem usunąć zakłóczenia typu sól i pieprz.

Następnie zdjęcie było zamieniane z przestrzeni barw RGB do przestrzeni barw w skali szarości, w celu możliwości w kolejnej fazie zastosowania progowania. Na podstawie sprogowanego zdjęcia istnieje możliwość wykrycia konturów kart na obrazie i tym samym wyznaczenia kart ze zdjęcia, przy znajomości wierzchołków karty oraz użycia transformacji perspektywicznej. Dodatkowo na podstawie tych samych konturów jest możliwość wyznaczenia środka karty na podstawie wartości momentów (składnia Python):

```
srodek_x = momenty['m10']/momenty['m00']  
srodek_y = momenty['m01']/momenty['m00']
```

Przykładowe zdjęcie pojedynczej karty, otrzymanej z zdjęcia po prawej stronie:



Ze względu na to, że niektóre zdjęcia miały na sobie nałożony gradient, to progowanie oparte o jedną wartość nie przynosiło zadowalających wyników. A zatem w celu wykrywania symboli, które w pierwszej kolejności wiązało się z oddzieleniem symboli od tła, zamieniałem piksele koloru karty na czarny przez co dobranie progu było już znacznie łatwiejsze.



Następnie w celu wyznaczenia kontur, które dotyczyły symbolu karty (ze względu na to, że trzy symbole na karcie są tym samym to chciałem zdobyć jedynie ten największy - położony na środku) wykonywałem progowanie tak jak w przypadku wczytywanego całego zdjęcia i nakładłem ograniczenie na wielkość pola kontur oraz położenie środka ciężkości, które powinno być zbliżone do środka karty. Dzięki takiemu podejściu mogłem otrzymywać jedynie jeden kontur (w przypadku UNO Revers Card dwa takie kontury), które dotyczyły właśnie symbolu karty i pomijały inne kontury na karcie takie jak np. kontur elipsy.

W celu odczytania symbolu z karty wykorzystałem HuMoments, na początku odczytałem wartości HuMoments z kart ze zdjęć oryginalnych i stworzyłem następującą tabelę, w której znajdują się jej wartości przybliżone (w samym projekcie użyłem dokładniejszych wartości):

| | HuMoment1 | HuMoment2 | HuMoment3 | HuMoment4 | HuMoment5 | HuMoment6 | HuMoment7 |
|------------------|-----------|-----------|-----------|-----------|------------|------------|------------|
| Karta Stop | 1.612e-01 | 6.554e-04 | 5.584e-08 | 3.204e-10 | -8.341e-19 | -5.669e-12 | -1.068e-18 |
| UNO Reverse Card | 2.525e-01 | 2.886e-02 | 2.095e-03 | 4.958e-04 | 4.949e-07 | 8.167e-05 | 1.023e-07 |
| 2 | 3.484e-01 | 2.857e-02 | 1.998e-04 | 2.014e-05 | -6.349e-10 | -3.169e-06 | -1.109e-09 |
| 4 | 2.066e-01 | 5.020e-03 | 3.408e-03 | 5.218e-05 | -1.642e-08 | -2.070e-06 | -1.464e-08 |
| 5 | 3.246e-01 | 1.725e-02 | 3.433e-04 | 2.151e-06 | -3.523e-11 | -2.670e-07 | -4.666e-11 |

Poniższy wzór:

$$HuMoments_{znormalizowany} = \log_{10}|HuMoments|$$

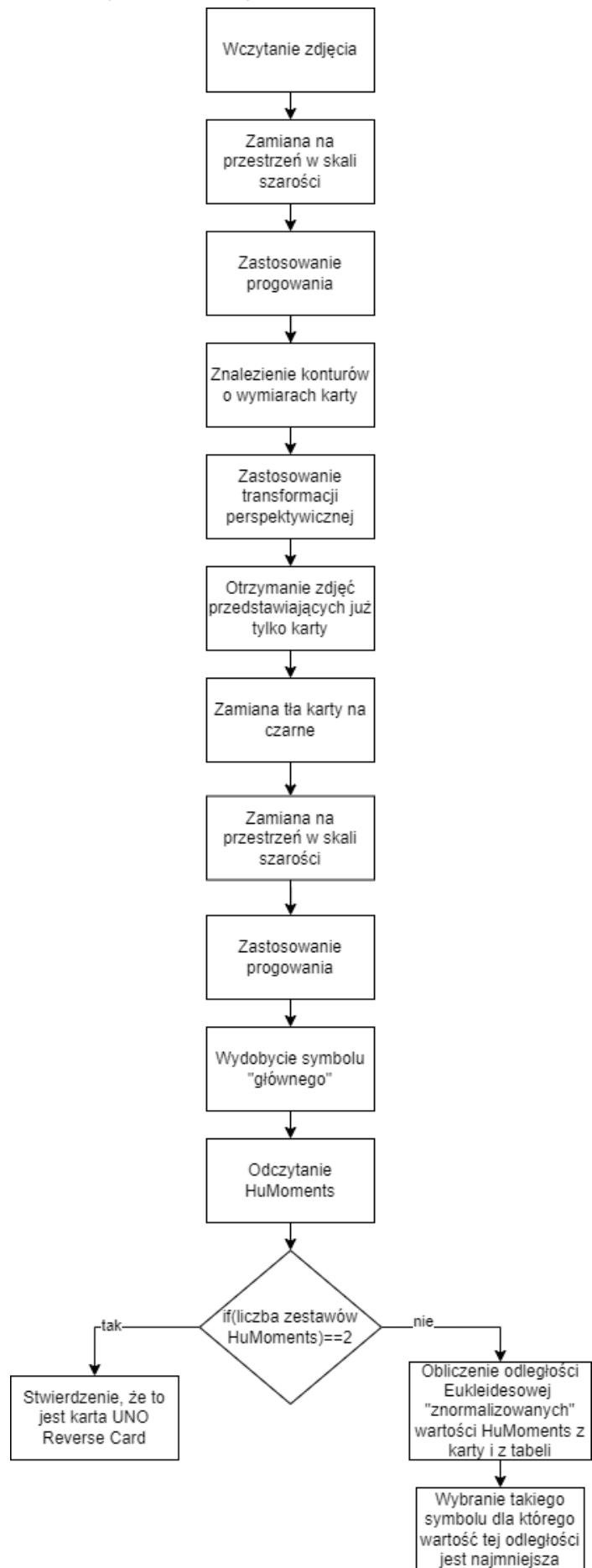
Pozwolił mi na „normalizację” wartości HuMoments odczytywanych z karty jak i tabeli. I następnie w celu dopasowania „znormalizowanych” wartości do „znormalizowanych” wartości z powyższej tabeli używałem odległości Euklidesowej i przypisywałem taki symbol do karty, gdzie wartość tej odległości była najmniejsza.

Wzór:

$$symbol = \min \left(\sqrt{\sum (HuMoment_{z\ karty:znormalizowany} - HuMoment_{z\ tabeli:znormalizowany})^2} \right)$$

W przypadku kart UNO Reverse Card otrzymywałem dwa zestawy takich HuMoments (po jednej dla strzałki), więc jeżeli zestaw HuMoments był dwa to od razu przypisywałem tej karcie symbol: UNO Reverse Card.

2.2. Schemat blokowy ścieżki przetwarzania



3. Wyniki

Za pomocą powyżej wspomnianej metody udało mi się przyporządkować poprawnie wszystkie karty i tym samym otrzymać następujące wyniki (w przypadku podawania współrzędnych środka pierwsza wartość to współrzędna y a druga współrzędna to x):

3.1. Zdjęcie 1



Wyniki:

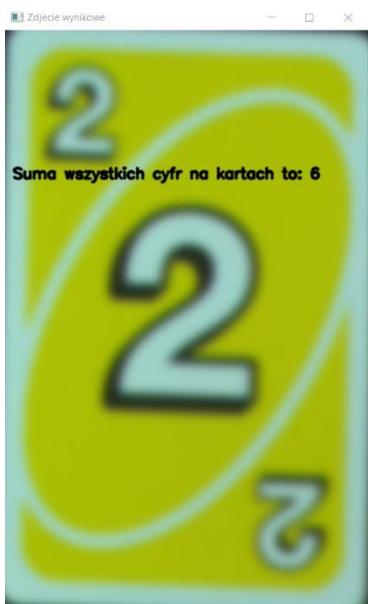




3.2. Zdjęcie 1.blur



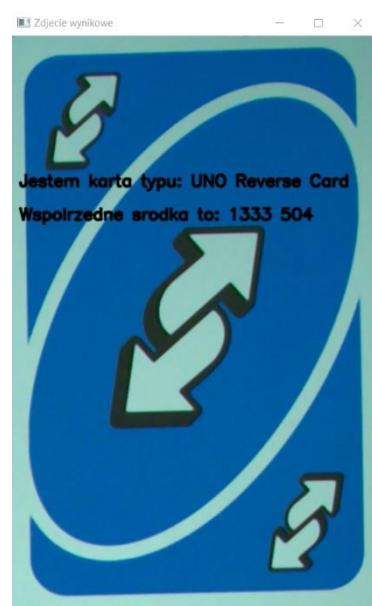
Wyniki:



3.3. Zdjęcie 1_gradient



Wyniki:





3.4. Zdjęcie 1_sp



Wyniki:



3.5. Zdjęcie 2



Wyniki:

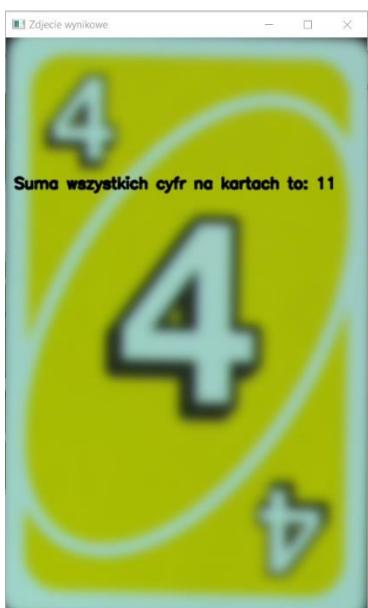
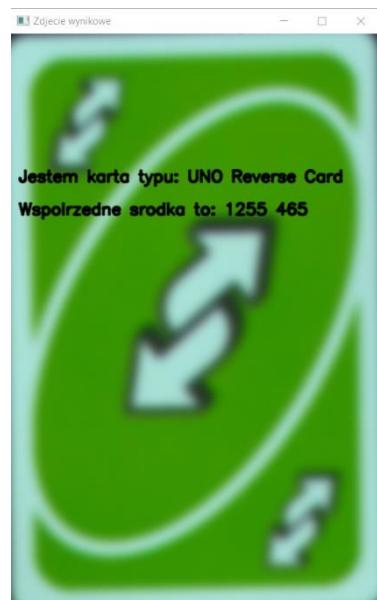
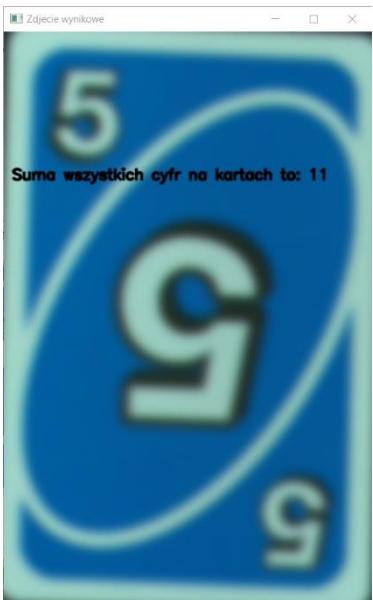




3.6. Zdjęcie 2 blur



Wyniki:

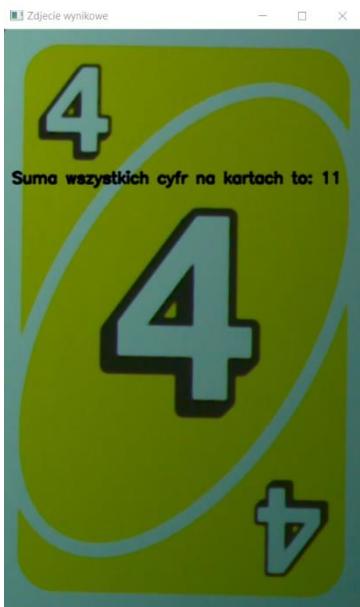


3.7. Zdjęcie 2_gradient



Wyniki:





3.8. Zdjęcie 2_sp



Wyniki:



3.9. Zdjęcie 3



Wyniki:





3.10. Zdjęcie 3.blur



Wyniki:

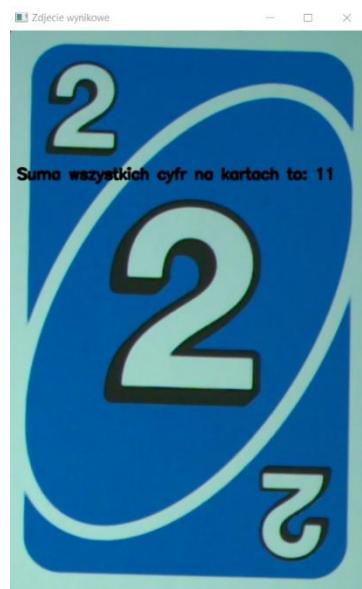


3.11. Zdjęcie 3_gradient



Wyniki:





3.12. Zdjęcie 3_sp



Wyniki:

