P O L I T E C H N I K A W A R S Z A W S K A

CYFROWE METODY PRZETWARZANIA OBRAZÓW

Projekt 2

Rozpoznawanie kart UNO

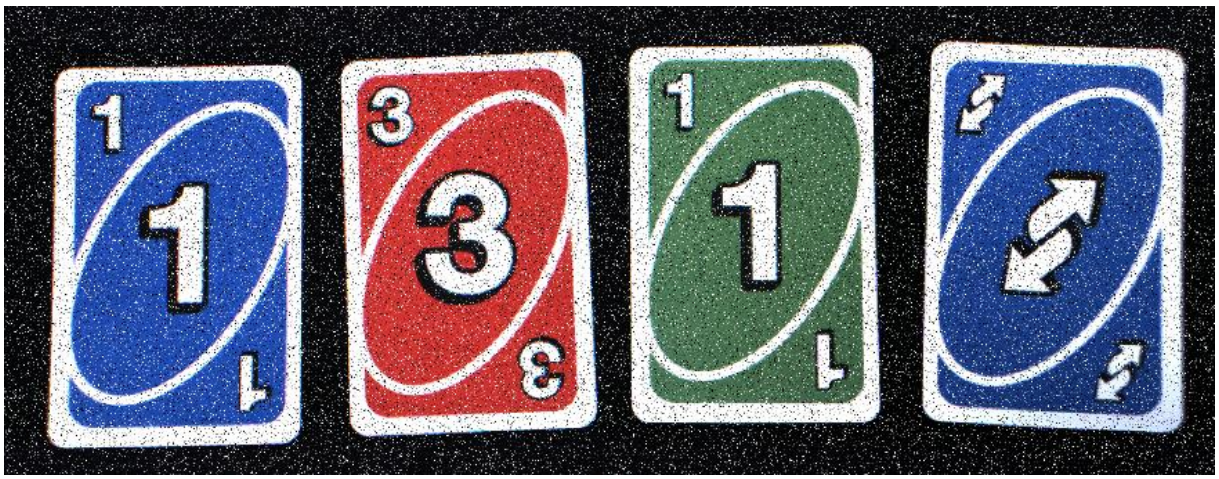
Wykonał:

Oleg Łyżwiński

Warszawa 2022/2023

1. Cel projektu.

Celem projektu było przygotowanie aplikacji w języku Python zdolnej do rozpoznawania kart uno znajdujących się na zdjęciach wykonanych podczas laboratorium. Na każdym zdjęciu znajdowały się 4 karty ułożone w linii, ponadto na zdjęcia nałożono filtry utrudniające rozpoznanie: gradient, salt_pepper i blur. Poniżej przedstawiono przykładowe zdjęcie:



2. Prezentacja algorytmu.

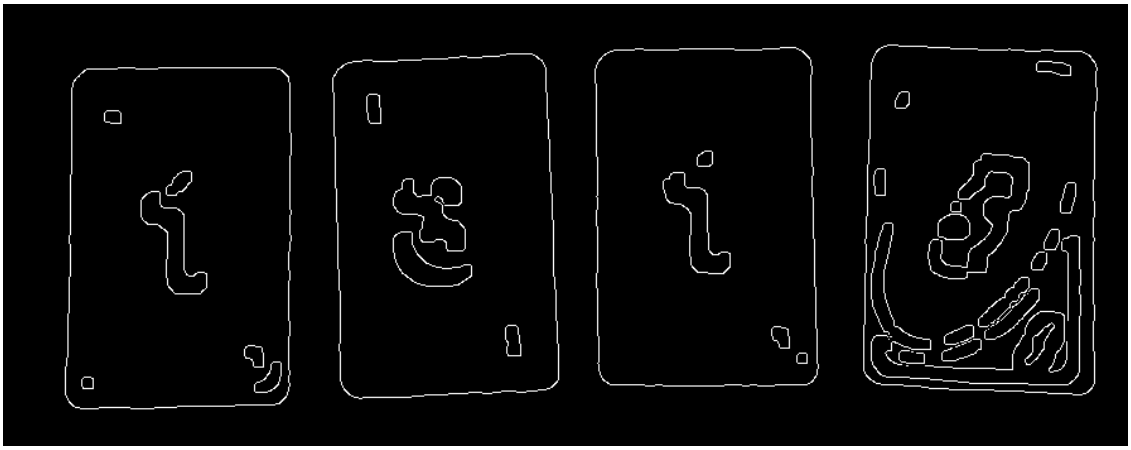
2.1. Wyszukiwanie kart.

Pierwszym napotkanym problemem było podzielenie kart znajdujących się na zdjęciach na 4 oddzielne zmienne. W tym celu zastosowano przetwarzanie obrazu uwydatniające krawędzie, po czym na uzyskanych krawędziach opisano najmniejszy możliwy prostokąt.

Algorytm do uwydatniania krawędzi:

Najpierw zastosowano filtr unsharp mask w celu usunięcia niejednorodności filtru pieprz i sól. Następnym krokiem była binaryzacja zdjęcia o odwróconych kolorach. W tym celu zastosowano tresholding (cv.THRESH_BINARY). Po tym wykonano erozję karty filtrem o wymiarach 8 na 8 i wyznaczono same krawędzie funkcją cv.Canny.

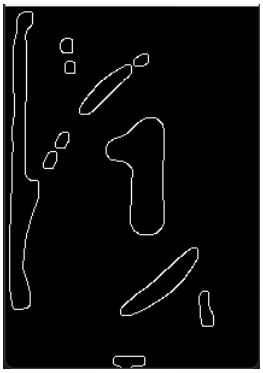
Poniżej przedstawiono wyniki działania algorytmu dla karty z nałożonym salt_pepper:



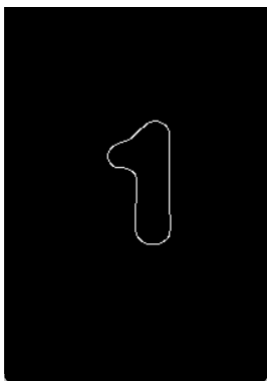
2.2. Rozpoznawanie symboli.

Kolejnym krokiem było opisanie na konturach kart (wydobytych cv.RETR_EXTERNA) na zdjęciu minimalnych prostokątów. Przy użyciu funkcji box wydobyto współrzędne wierzchołków, które uszeregowano w odpowiedniej kolejności. Przy użyciu funkcji cv.warpPerspective przypisano kartę do nowego obrazu, który przedstawiono poniżej:

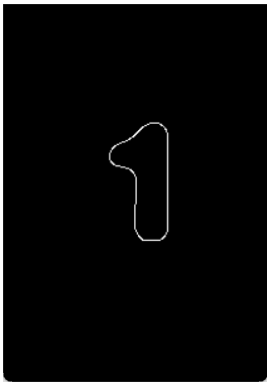
Następnie na tak przygotowanej karcie zastosowano przetwarzanie wyróżniające krawędzi na karcie, w szczególności symbol na środku. W tym celu najpierw rozmyto kartę przy użyciu filtrów medianBlur i GaussianBlur. Po tych operacjach wykonano wyrównanie histogramu na karcie o odwróconych kolorach. Kolejnym krokiem była binaryzacja karty funkcją cv.adaptiveThreshold przy zastosowaniu parametru (cv.ADAPTIVE_THRESH_MEAN_C). Następnie wykonano erozję macierzą o wymiarach 3 na 3, aby usunąć małe struktury utrudniające rozpoznanie karty. Wykonano również dylatację aby połączyć symbole takie jak zmiana kierunku czy plus 2. Na koniec pozostawiono jedynie krawędzie na karcie funkcją canny. Uzyskany efekt przedstawiono poniżej:



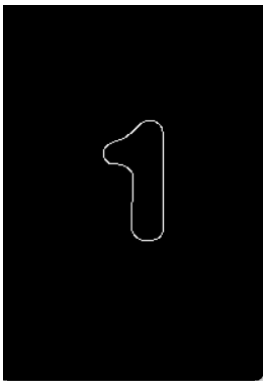
Następnie wykonano dwukrotne maskowanie kart najpierw zostawiając jedynie elipsę, w której znajduje się symbol na środku. Drugie maskowanie wykonano przy użyciu minimalnego prostokąta jaki można opisać na każdym symbolu. Działanie dla wszystkich filtrów przedstawiono poniżej:



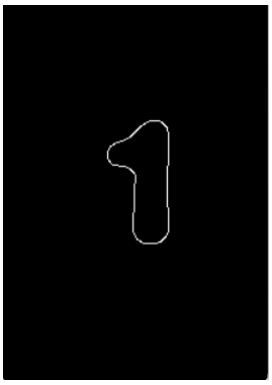
Zwykła karta



Karta z blurem



Karta z gradientem



Karta z salt_peper

Dzięki zastosowanemu przetwarzaniu obrazu cyfry na wszystkich kartach wyglądały bardzo podobnie co pozwoliło na przejście do następnego kroku jakim było wyznaczenie momentów hu dla każdej z kart. Porównując momenty kart wyznaczono przedziały, w których zawierały się dane karty. Napotkano tutaj problem ponieważ niektóre przedziały były bardzo blisko siebie, jednak na tyle minimalnie się różniły, że możliwe było ich rozpoznanie. W ten sposób uzyskano symbol każdej karty.

2.3. Wyznaczenie środka karty

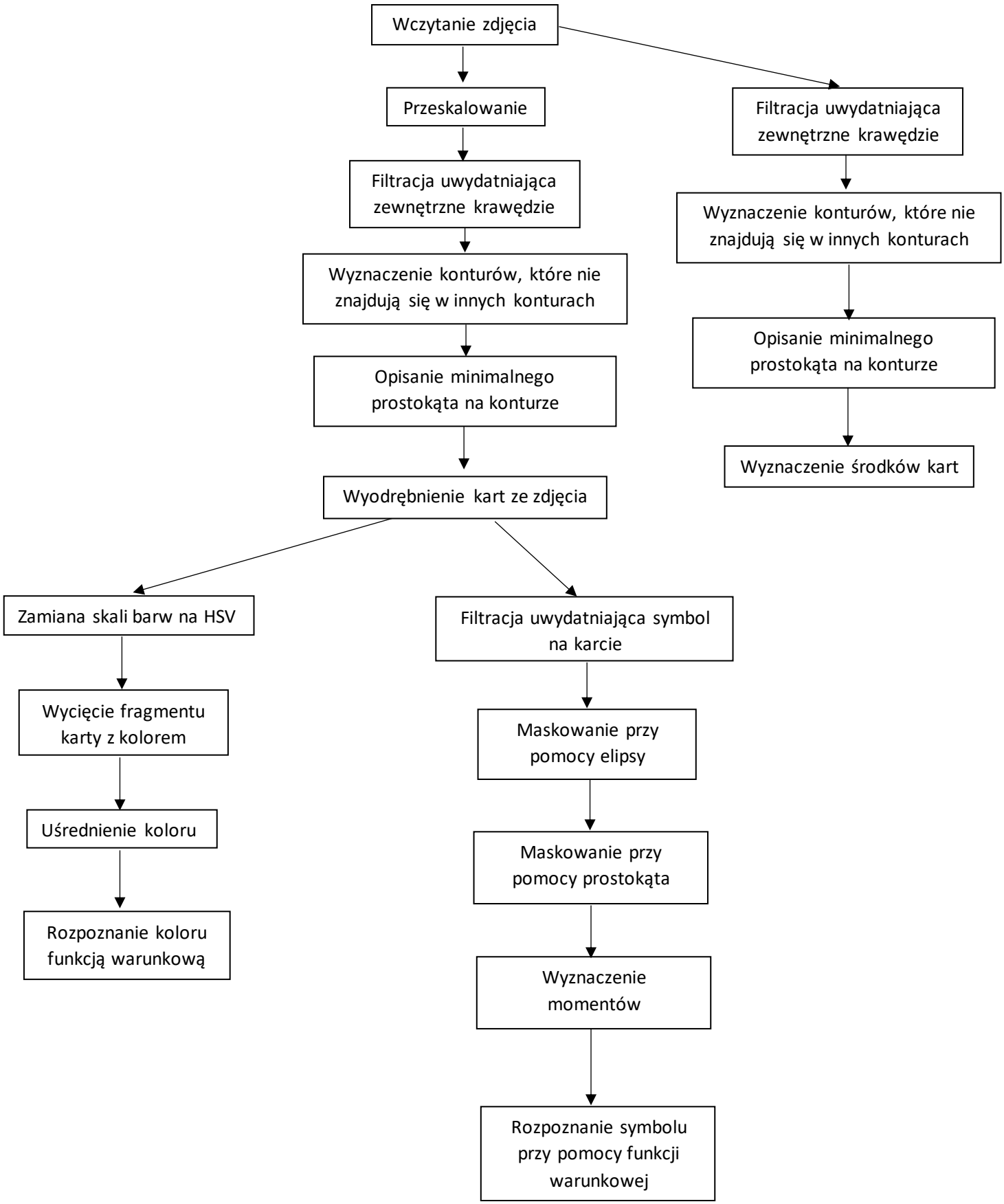
Pewnym problemem było wyznaczenie środka karty, ponieważ w trakcie przetwarzania pracowano na zdjęciu pomniejszonym o 40%, aby zdjęcie mieściło się na ekranie monitora. W związku z tym, aby znaleźć koordynaty środków kart dodano dodatkową funkcję mimo, że koordynaty środka każdej karty mogła zwracać już funkcja wycinająca karty. Parametry środka karty zapisane są w funkcji cv.minAreaRect, jednak przez korzystanie ze zmniejszonego zdjęcia nie można było ich zwrócić już w funkcji wycinającej karty (praca na większym zdjęciu spowodowała zmiany momentów dla poszczególnych symboli).

2.4. Rozpoznawanie koloru.

Rozpoznawanie koloru przeprowadzono na wyciętych kartach, których przestrzeń barw przeniesiono do HSV. Następnie pobrano mały wycinek karty z kolorem i wyznaczono średni kolor na tym wycinku. Uzyskane w ten sposób wyniki porównano przy pomocy funkcji warunkowej i przypisano do odpowiednich kolorów.

3. Schemat blokowy

Poniżej przedstawiono schemat blokowy opisanych wyżej algorytmów



4. Prezentacja działania algorytmu.

Zestaw 1

Zestaw 2


NIEBIESKA 5
ZIELONA 5
CZERWONA 1
CZERWONA +2
[1363 560]


NIEBIESKA 5
ZIELONA 5
CZERWONA 1
CZERWONA +2
[1364 560]

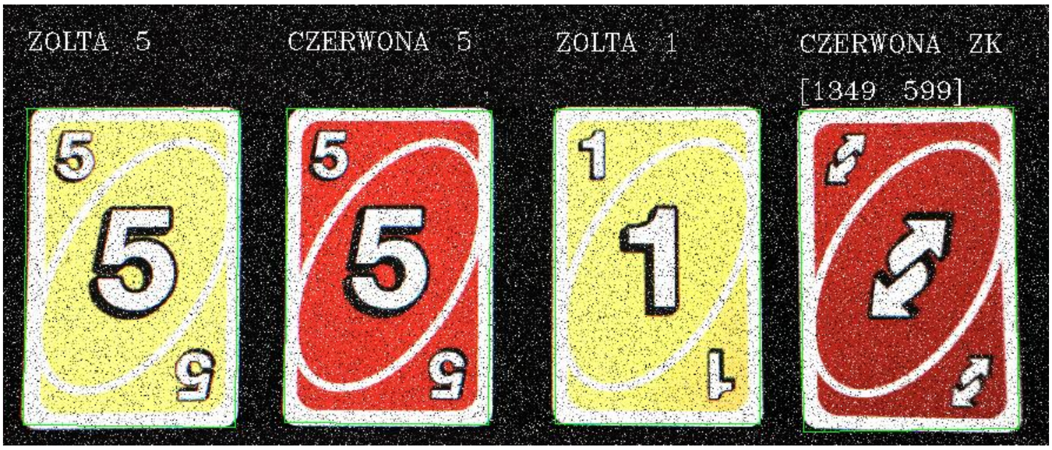

NIEBIESKA 5
ZIELONA 5
CZERWONA 1
CZERWONA +2
[1363 560]


NIEBIESKA 5
ZIELONA 5
CZERWONA 1
CZERWONA +2
[1363 560]

Zestaw 3

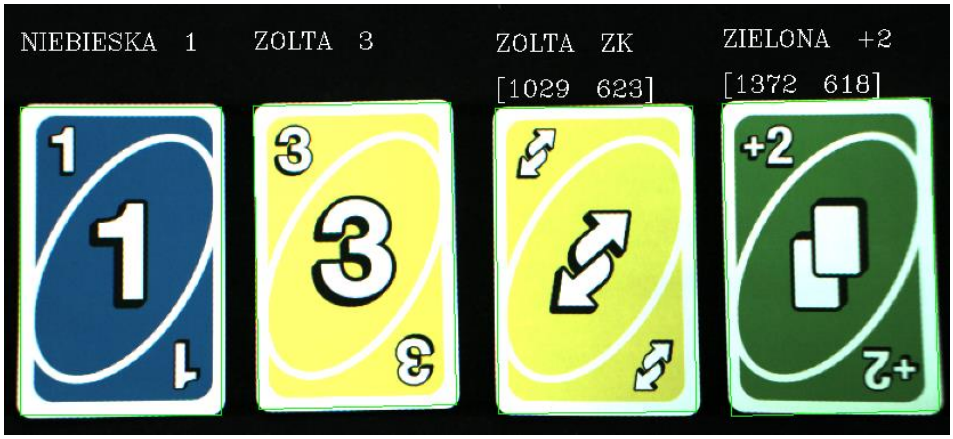

ZOLTA 5
CZERWONA 5
ZOLTA 1
CZERWONA ZK
[1349 599]


ZOLTA 5
CZERWONA 5
ZOLTA 1
CZERWONA ZK
[1349 599]

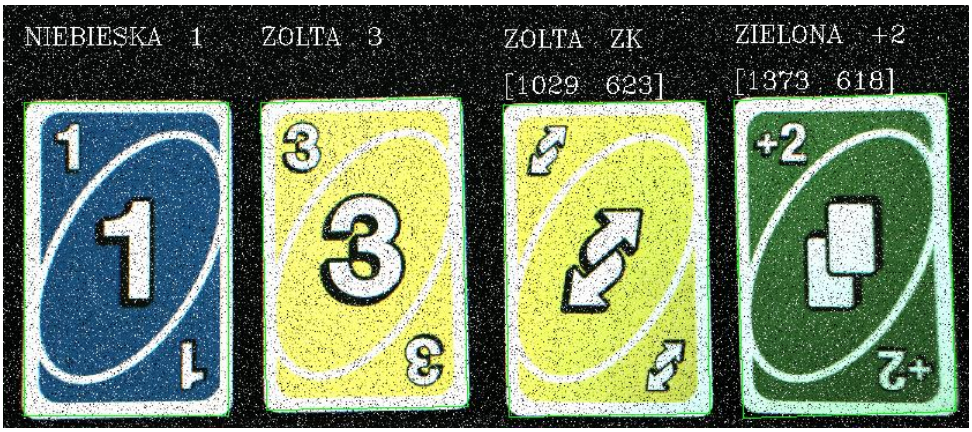

ZOLTA 5
CZERWONA 5
ZOLTA 1
CZERWONA ZK
[1349 599]

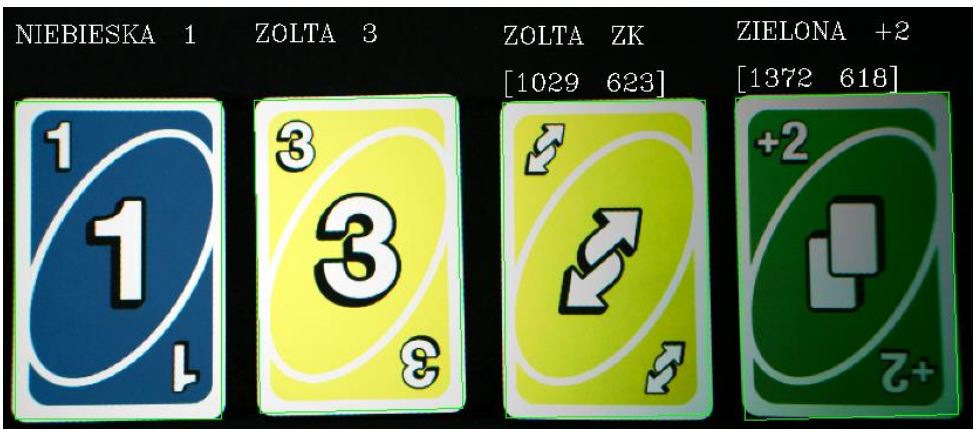

ZOLTA 5
CZERWONA 5
ZOLTA 1
CZERWONA ZK
[1349 599]

Zestaw 4


NIEBIESKA 1
ZOLTA 3
ZOLTA ZK
[1029 623]
ZIELONA +2
[1372 618]


NIEBIESKA 1
ZOLTA 3
ZOLTA ZK
[1028 624]
ZIELONA +2
[1372 618]


NIEBIESKA 1
ZOLTA 3
ZOLTA ZK
[1029 623]
ZIELONA +2
[1373 618]


NIEBIESKA 1
ZOLTA 3
ZOLTA ZK
[1029 623]
ZIELONA +2
[1372 618]