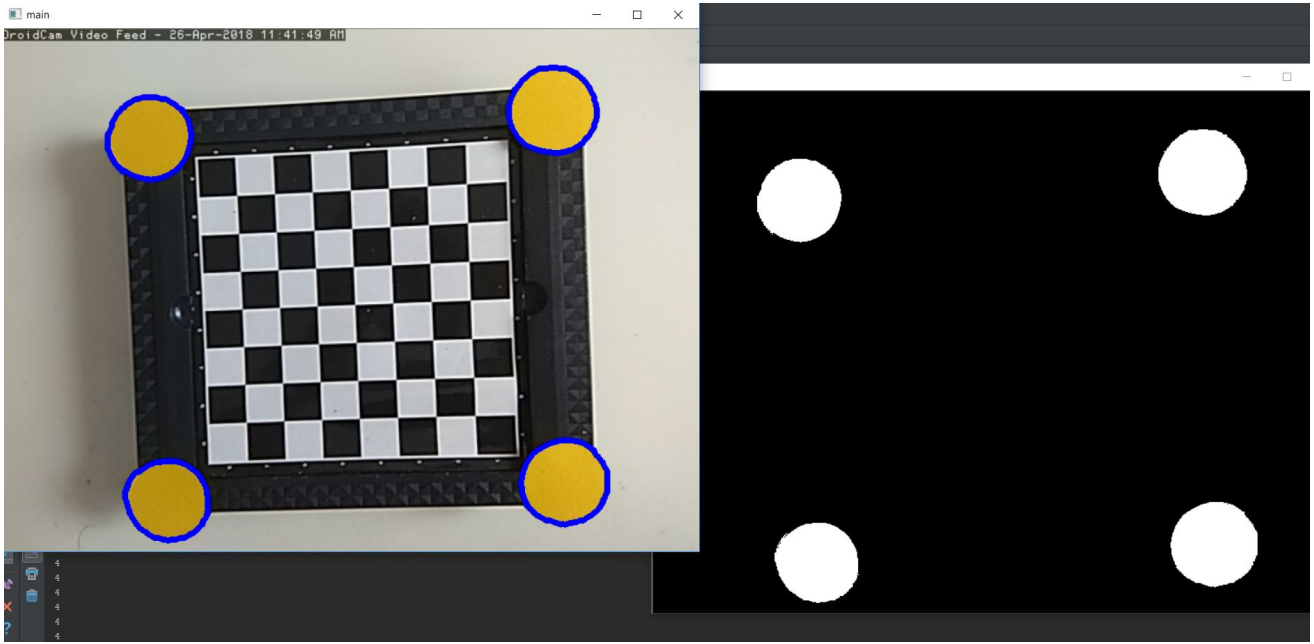
Podstawy teleinformatyki

Rozpoznawanie obrazu z gry w warcaby oraz wizualizacja stanu gry na komputerze

Część IV

Paulina Mrozek Kornel Krześlak Kamil Sagalara Hubert Springer

Sposoby wykrycia markerów - testy



Kontury

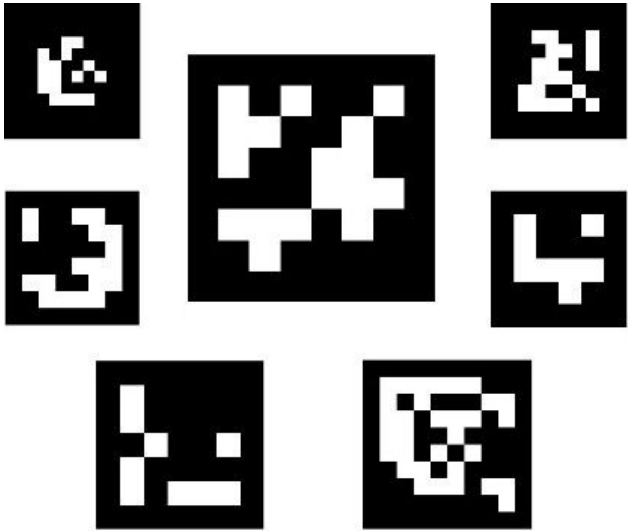
- Liczba wykrytych konturów zmienia się w każdej klatce, nawet jeśli obraz jest dobrze oświetlony a kolory są odpowiednio dobrane.
- Często w jednym okręgu znajduje się kilka konturów
- Można ignorować klatki, w których jest za dużo konturów
- Nadal pozostaje problem kolejności wykrycia konturów - trzeba je posortować według pozycji i dopiero wtedy można “wyciąć” planszę
- Ogólnie detekcja konturów działa “randomowo”

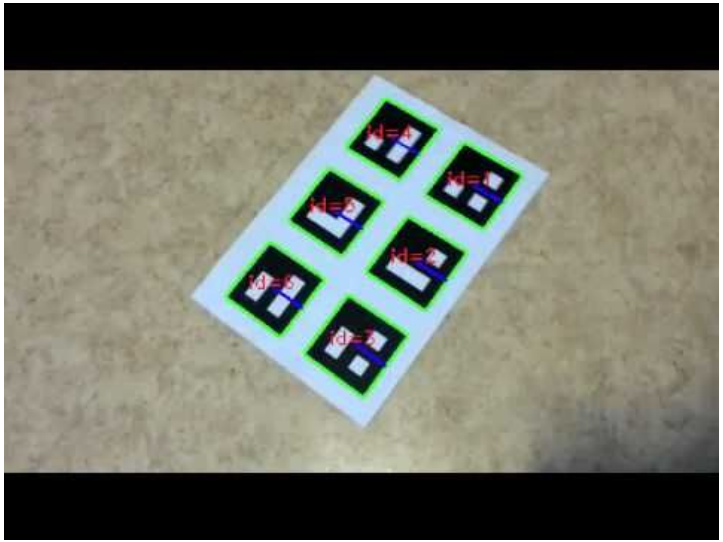
Okręgi - HOUGH CIRCLE TRANSFORM

- OpenCV oferuje funkcję *HoughCircles,* która pozwala na wykrycie okręgów
- W przypadku okrągłych markerów jej zastosowanie powinno dać lepsze efekty niż wykrycie konturów

Ar - markers

- Każdy marker ma swoje unikalne Id
- Do Id można przypisać wiele właściwości, np. pozycję

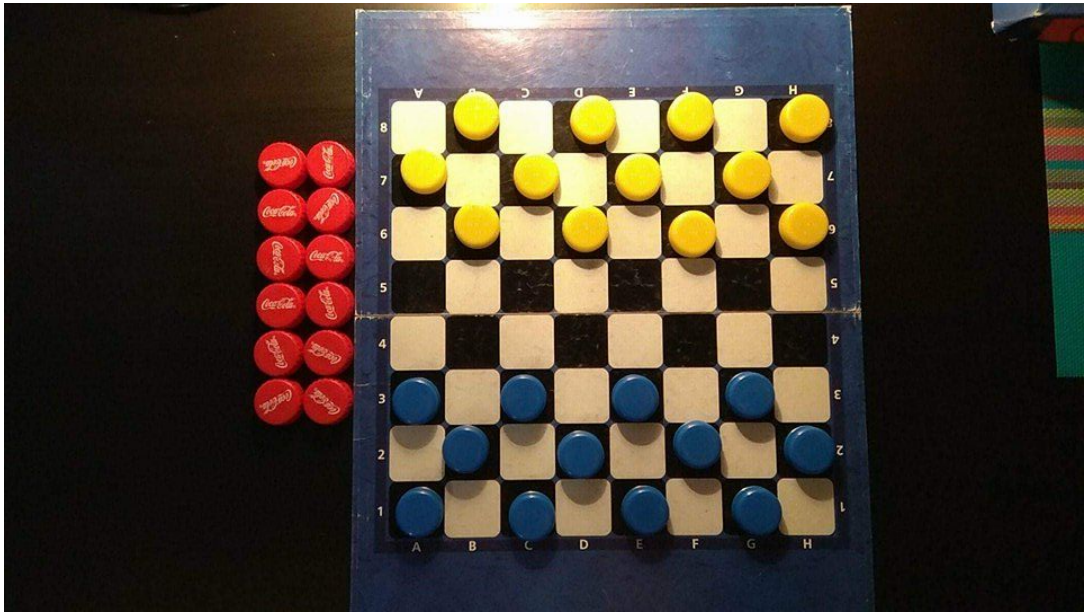




Detekcja kolorów na podzielonej planszy

- Po wykryciu planszy wystarczy podzielić ją na 64 bloki
- Każdy blok będzie przepuszczony przez algorytm wykrywający dominujący lub średni kolor (zależy, który okaże się lepszy)
- Klatki w których wykrycie planszy się nie powiodły, muszą być ignorowane, żeby nie powodować dodatkowych błędów
- Trzeba zdefiniować warunki, które mówią, kiedy dana klatka jest błędna (np. błędna liczba markerów, wzajemne odległości markerów)

Docelowa plansza z pionkami



Sprawdzanie poprawności ruchu

- wyświetlenie listy dostępnych ruchów dla aktywnego gracza
- zamiana współrzędnych (i,j) na postać bardziej przyjazną dla użytkownika litera+cyfra np. (0,0) -> A8
- przykład: sprawdzenie dostępnych ruchów z pola E3

```
--------------------

Możliwe ruchy z E3:
E3 C5 A7
E3 G5

--------------------
```

```
class Coordinates:
    def __init__(self, row, column):
        self.i = row
        self.j = column

    def to_string(self):
        if self.j == 0:
            return "H"+str(self.i + 1)
        elif self.j == 1:
            return "G"+str(self.i + 1)
        elif self.j == 2:
            return "F" + str(self.i + 1)
        elif self.j == 3:
            return "E" + str(self.i + 1)
        elif self.j == 4:
            return "D" + str(self.i + 1)
        elif self.j == 5:
            return "C" + str(self.i + 1)
        elif self.j == 6:
            return "B" + str(self.i + 1)
        elif self.j == 7:
            return "A" + str(self.i + 1)
        else:
            return None
```