

Politechnika Śląska
Wydział Automatyki Elektroniki i Informatyki
Kierunek: Automatyka i Robotyka sem. 7

Gliwice
Rok akademicki
2020/2021
Semestr zimowy

Laboratorium

Przetwarzania Informacji Wizyjnej

Rozpoznawanie płytek (kamieni) domina

Choiński Kamil
Stabla Oskar
Gr. I

Liczba oczek

1. Wstęp

Celem projektu w ramach przedmiotu o nazwie Przetwarzanie Informacji Wizyjnej było stworzenie programu będącego w stanie rozpoznać na podanym przez użytkownika obrazie kostkę domina, a następnie zliczyć jej liczbę oczek wraz z podziałem na segment górny i dolny. Środowisko programistyczne nie było z góry narzucone, jednak postanowiliśmy, iż całość zostanie wykonana w programie Matlab z uwagi na jego doskonale możliwości przetwarzania informacji w obrazie. W celu usprawnienia prac nad projektem wykorzystane zostało środowisko Git, które umożliwiło asynchroniczną pracę nad jednym kodem źródłowym.

2. Podobne rozwiązania

- Projekt Domino-Detection

<https://github.com/karthikVenkataramana/Domino-Detection>

Projekt używa języka Python i Google Object detection API

Dzięki użyciu Google Object detection API program jest w stanie rozpoznawać w jakim położeniu znajduje się domino. Po wykryciu domino program determinuje kierunek skierowania oczek domino według oznaczeń poniżej.

Oczka do dołu - A

Oczka do góry - B

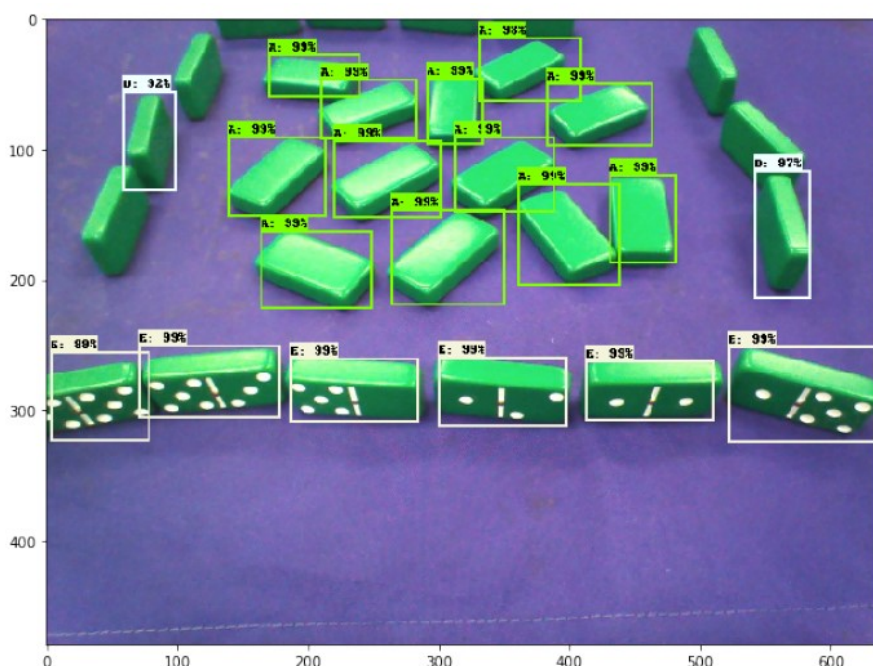
Stoi i patrzy w lewo - C

Stoi i patrzy w prawo - D

Stoi i patrzy w kamerę - E

Stoi i nie patrzy w kamerę - F

Nie jest to efekt jaki chcieliśmy osiągnąć w naszym projekcie, jednak jest to ciekawa implementacja wykrywania domino z użyciem Google Object detection API.



- Projekt DominoesDetection

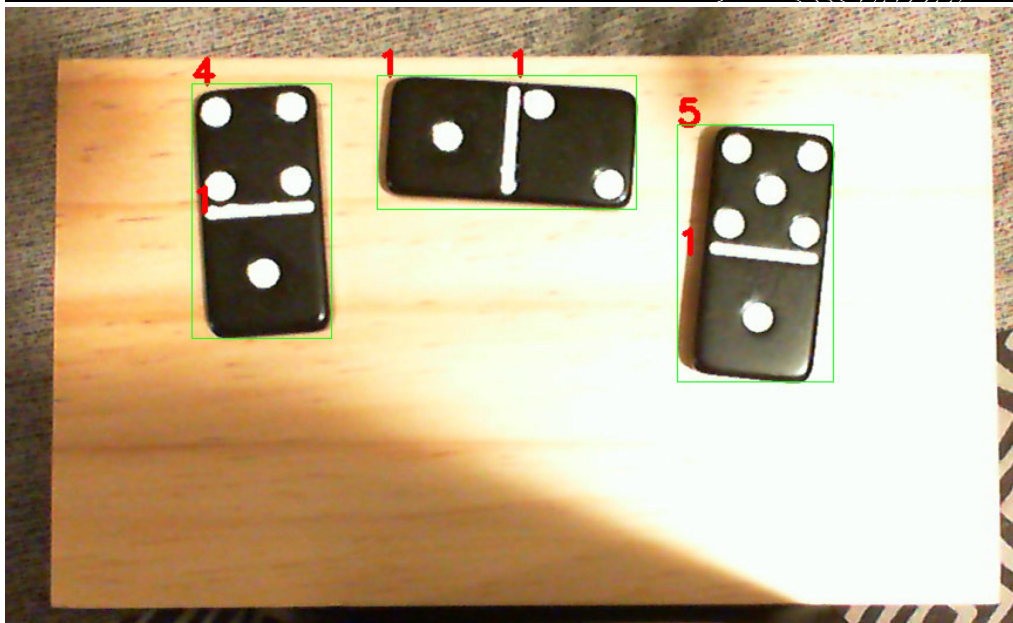
<https://github.com/HookJordan/DominoesDetection>

Projekt ten używa języka C++ wraz z biblioteką OpenCV 3.0

Jego celem jest wykrywanie kostek domino na załadowanym obrazie i wyświetlenie liczby oczek przy wykrytym domino.

Dzięki użyciu biblioteki OpenCV 3.0 możliwe jest zlokalizowanie kostki domino i jego wyodrębnienie z reszty obrazu. Pozwala to na dalszą analizę osobnego domino dzięki użyciu osobnej funkcji która dzieli domino na 2 części i osobno wypisuje liczbę oczek dla każdej części.

Projekt nie działa w każdych warunkach, jednak wyświetla poprawne wartości przy dobrym oświetleniu jak pokazane poniżej. Program ma też problem przy ułożonym domino w innej orientacji niż poziomo (przy pochylonym zdjęciu)



3. Implementacja projektu w Matlabie

Wykrywanie kostek domina i zliczanie oczek podzielone jest na parę etapów:

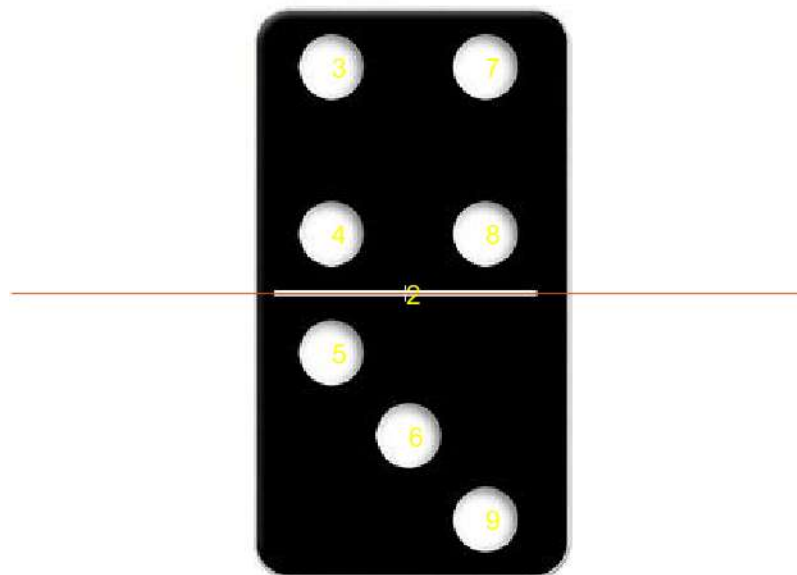
1. Wczytanie obrazu poprzez okno dialogowe, w którym użytkownik wybiera pożądaną próbkę, którą chce poddać analizie
2. Pozbycie się kolorystyki. W tym celu wykorzystano funkcję `rgb2gray()`, która zwraca obraz w skali szarości
3. Obliczenie progu binaryzacji, który stanowi wyznacznik, powyżej której wartości piksel ma wartość 1, a poniżej której 0. Wykorzystano do tego metodę OTSU, czyli metodę progowania globalnego, opartego na histogramie obrazu szarego. W Matlabie metodę tę wykonuje funkcja `graythresh()`, przyjmująca obraz szary i zwracająca próg binaryzacji.
4. Binaryzacja obrazu. Posiadając obraz szary jak i próg binaryzacji możemy uzyskać obraz czarno-biały poprzez wywołanie funkcji `im2bw()`
5. Wyłuskanie obszarów, należących do zbinaryzowanego obrazu. W idealnym przypadku będą to tylko i wyłącznie koła/elipsy, linie oraz cała kostka. W tym celu służy funkcja `bwboundaries()`
6. Analiza parametrów obszarów w celu ich dalszej klasyfikacji – Funkcja `regionprops()`, która zwraca strukturę, zawierającą 30 parametrów dla każdego obszaru, m.in. kąt nachylenia, położenie środka, stopień wypełnienia, kołowość i wiele innych

| STATS | | | | | | | | | | | | | | |
|---------------------------|---------|----------------|----------------------|-------------|-----------------|-----------------|--------------|-------------|----------------|-------------------|------------|-------------|-----------------|--|
| 9x1 struct with 30 fields | | | | | | | | | | | | | | |
| Fields | Area | Centroid | BoundingBox | SubarrayIdx | MajorAxisLength | MinorAxisLength | Eccentricity | Orientation | ConvexHull | ConvexImage | ConvexArea | Circularity | Image | |
| 1 | 5595468 | [1.2661e+03... | [0.5000,0.5000,25... | 1x2 cell | 3.0162e+03 | 2.9490e+03 | 0.2100 | -13.6335 | 10001x2 double | 2500x2500 logical | 6250000 | 0.7328 | 2500x2500 L... | |
| 2 | 6112 | [605.7349,1... | [539.5000,1.4895... | 1x2 cell | 131.4444 | 59.2293 | 0.8927 | 3.9902 | 132x2 double | 60x132 logical | 6204 | 0.8001 | 60x132 logical | |
| 3 | 2 | [757.5000,1... | [756.5000,1.5345... | 1x2 cell | 2.3094 | 1.1547 | 0.8660 | 0 | 7x2 double | [1, 1] | 2 | 6.5423 | [1, 1] | |
| 4 | 6723 | [841.1838,1... | [773.5000,1.5965... | 1x2 cell | 134.5044 | 63.6618 | 0.8809 | 2.1053 | 127x2 double | 65x135 logical | 6829 | 0.8197 | 65x135 logical | |
| 5 | 6417 | [871.2364,1... | [804.5000,1.4775... | 1x2 cell | 133.2235 | 61.3454 | 0.8877 | 1.7961 | 149x2 double | 61x133 logical | 6504 | 0.8090 | 61x133 logical | |
| 6 | 5547 | [899.2991,1... | [835.5000,1.3665... | 1x2 cell | 127.4187 | 55.4520 | 0.9003 | 1.5242 | 114x2 double | 56x128 logical | 5649 | 0.7698 | 56x128 logical | |
| 7 | 15394 | [1.0909e+03... | [885.5000,1.5115... | 1x2 cell | 507.4593 | 40.2683 | 0.9968 | 23.9935 | 59x2 double | 200x417 logical | 16069 | 0.2218 | 200x417 logi... | |
| 8 | 5986 | [1.1357e+03... | [1.0705e+03,1.46... | 1x2 cell | 129.5762 | 58.8366 | 0.8910 | 0.9102 | 136x2 double | 59x130 logical | 6076 | 0.8070 | 59x130 logical | |
| 9 | 7527 | [1.3355e+03... | [1.2655e+03,1.68... | 1x2 cell | 140.9081 | 68.0371 | 0.8757 | 0.6759 | 131x2 double | 68x140 logical | 7642 | 0.8277 | 68x140 logical | |

7. Wykrycie linii – jest wykrywana w dwóch przypadkach, albo szerokość głównej osi elipsy zbudowanej na podstawie wykrytego obszaru jest co najmniej 4 razy większa od jej krótszej osi, albo gdy stopień wypełnienia obszaru w stosunku do „pudełka”, w którym jest definiowany przez `regionprops()` jest większy niż 0.9, a szerokość i wysokość są od siebie różne (warunek na prostokąt, którym jest linia)
8. Utworzenie równania prostej – w celu dalszej analizy położenia oczek konstruujemy prostą przechodzącą przez punkt środka linii domina, o kącie nachylenia równym kątowi nachylenia głównej osi elipsy zbudowanej na tej linii.

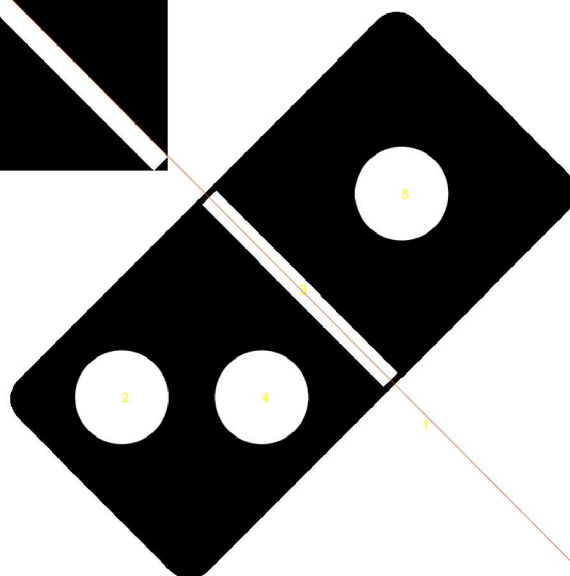
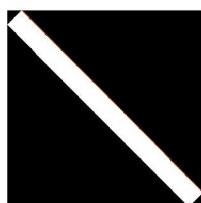
9. Wykrycie oczek i przyporządkowanie ich do określonego segmentu, górnego lub dolnego – W tym celu sprawdzamy, czy parametr Circularity (kołowość) danego obszaru zawiera się w przedziale od 0.76 do 1.1. Pozwala to przyporządkować zarówno koła jak i elipsy, które powstają przy przechylonej pozycji kostki domina. Następnie sprawdzamy, czy linia jest pionowa. Jeśli tak, to dochodzi do porównania współrzędnych x-owych środków kół/elips ze środkiem linii. W przypadku nachylenia innego niż poziome rozwiązywana jest nierówność $y > ax + b$, gdzie y i x to współrzędne środka koła/elipsy, a parametry „a” i „b” wyznaczone są z równania prostej. Jeśli y jest większe, to koło przypisywane jest do dolnego segmentu. Wynika to z tego, iż Matlab rozpoczyna indeksowanie drugiej współrzędnej od lewego górnego rogu ekranu.
10. Wyświetlenie otrzymanego rezultatu na obrazku

Część górną: 4 część dolną: 3



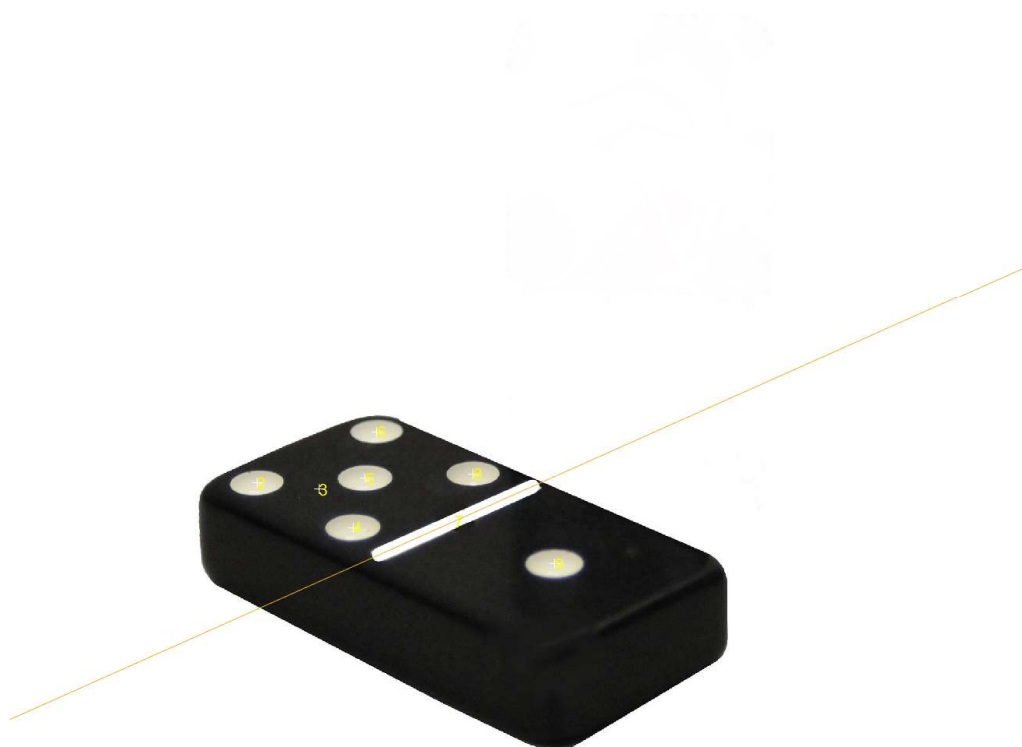
Przykład nr 1

Część góra: 1 część dolna: 2



Przykład nr 2

Część góra: 5 część dolna: 1



Przykład nr 3

4. Kod programu

```
5. clear all
6. clc
7. close all
8. %RGB = imread('domino_4.jpg');
9. [fn,pn]=uigetfile({'*.png'}, 'Wybierz obraz');
10.     RGB = imread([pn,fn]);
11.     figure,
12.     imshow(RGB),
13.     title('Original Image');
14.
15.     GRAY = rgb2gray(RGB);
16.
17.     threshold = graythresh(GRAY);
18.
19.
20.     BW = im2bw(GRAY, threshold);
21.
22.
23.     [B,L] = bwboundaries(BW, 'noholes');
24.
25.     STATS = regionprops(L, 'all'); % we need 'BoundingBox' and
    'Extent'
26.
27.     figure,
28.     imshow(RGB),
29.     title('Results');
30.     hold on
31.
32.     circles_1 = 0;
33.     circles_2 = 0;
34.     random_1 = 0;
35.     is_line = 0;
36.     lineSTAT=0;
37.     %bounding box defined for each shapes as
    [x_cordinate,y_cordinate,x_width,y_width]
38.     for i = 1 : length(STATS)
39.         %for rectangles, we have x_width != y_width,extent =1
40.         if(( (STATS(i).BoundingBox(3)~=STATS(i).BoundingBox(4)) &&
    (STATS(i).Extent>=0.9)) ||
    (STATS(i).MajorAxisLength>4*STATS(i).MinorAxisLength))
41.             is_line = is_line + 1;
42.             lineSTAT = STATS(i);
43.             imshow(STATS(i).ConvexImage)
44.             centroid = STATS(i).Centroid;
45.             plot(centroid(1),centroid(2),'w+');
46.             text(centroid(1),centroid(2),num2str(i),'Color','y');
47.         end
48.     end
49.
50.     vertical=0;
```

```

51.     a = tand(lineSTAT.Orientation);
52.     if (lineSTAT.Orientation>=88)
53.         vertical=1
54.     end
55.     a=a*(-1);
56.     b = lineSTAT.Centroid(2) - (a*lineSTAT.Centroid(1));
57.
58.     x=0:1:length(BW)-1;
59.
60.     y=a*x+b;
61.     plot(x,y)
62.
63.     if (isstruct(lineSTAT))
64.         for i = 1 : length(STATS)
65.             centroid = STATS(i).Centroid;
66.             if(STATS(i).Circularity>=0.76 && STATS(i).Circularity<=
1.1 )
67.                 if(vertical)
68.                     if(STATS(i).Centroid(1)<(lineSTAT.Centroid(1)))
69.                         circles_1= circles_1 +1;
70.                     else
71.                         circles_2 = circles_2 +1;
72.                     end
73.                 else
74.                     if(STATS(i).Centroid(2)<(a*STATS(i).Centroid(1)+b))
75.                         circles_1= circles_1 +1;
76.                     else
77.                         circles_2 = circles_2 +1;
78.                     end
79.                 end
80.                 plot(centroid(1),centroid(2),'w+');
81.                 text(centroid(1),centroid(2),num2str(i),'Color','y');
82.
83.             elseif((STATS(i).BoundingBox(3)==STATS(i).BoundingBox(4)) &&
(STATS(i).Extent > 0.76 && STATS(i).Extent < .795))
84.
85.                 plot(centroid(1),centroid(2),'wO');
86.                 text(centroid(1),centroid(2),num2str(i),'Color','y');
87.
88.
89.             elseif((STATS(i).BoundingBox(3)~=STATS(i).BoundingBox(4)) &&
(STATS(i).Extent > 0.76 && STATS(i).Extent < .795))
90.
91.                 plot(centroid(1),centroid(2),'w*');
92.                 text(centroid(1),centroid(2),num2str(i),'Color','y');
93.             end
94.
95.         end

```



```
96.         end
97.
98.
99.         title("Część górna: " + circles_1 + " część dolna: " +
circles_2);
```