

Górka Bartosz      Kruszyna Mateusz  
127228                127252

## System rozpoznawania wybranych monet

Poznań 01.12.2017

## **Spis treści**

<b>1. Cel projektu . . . . .</b>	<b>3</b>
<b>2. Ograniczenia . . . . .</b>	<b>3</b>
<b>3. Historia pracy nad algorytmem . . . . .</b>	<b>3</b>
<b>4. Analiza algorytmu . . . . .</b>	<b>3</b>
4.1. Wyszukiwanie monet . . . . .	3
4.2. Wyszukiwanie banknotów . . . . .	5
4.3. Sposób rozpoznawania monet . . . . .	7
4.4. Sposób rozpoznawania banknotów . . . . .	8
<b>5. Wnioski . . . . .</b>	<b>9</b>
<b>6. Analizowane przypadki . . . . .</b>	<b>10</b>
6.1. Pomyślne rozpoznanie . . . . .	10
6.2. Niepomyślne rozpoznania . . . . .	11
<b>7. Bibliografia . . . . .</b>	<b>11</b>

## 1. Cel projektu

Systemy automatycznego wnioskowania, rozpoznawania obrazów i wzorców stają się coraz bardziej popularne i wykorzystywane w codziennym życiu. Projekt miał na celu przygotowanie wstępniego zarysu programu komputerowego będącego w stanie rozpoznawać wybrane monety stosowane w Polsce.

Ograniczając projekt poprzez wyeliminowanie sztucznej inteligencji, system miał osiągnąć jak najlepszy wynik wykorzystując samą graficzną obróbkę plików źródłowych.

## 2. Ograniczenia

Przy przygotowaniu realizacji zabronione było wykorzystywanie w pełni gotowych rozwiązań ułatwiających wyszukiwanie monet i banknotów na obrazach. Zabronione było wykorzystanie *klasyfikatora Haara*, który rozwiązałby większość problemów. Dodatkowo wszelkiego rodzaju *OCR* również był zakazany.

Jako ułatwienie przyjęto ograniczenie analizowanych monet i banknotów do unikatowych kolorystycznie tj. *0,05 PLN, 1 PLN, 2 PLN, 5 PLN, 10 PLN, 50 PLN, 100 PLN*.

## 3. Historia pracy nad algorytmem

Przygotowanie algorytmu do wykrywania obiektów na obrazie nie jest rzeczą trywialną. Wiele czynników może negatywnie oddziaływać na proces analizy. Są to między innymi różnice w naświetleniu obiektu, refleksy, czy też sama jakość zdjęcia. Prace rozpoczęto od wykrywania monet na obrazie, aby następnie poddać analizie i określić z jaką monetą mamy do czynienia.

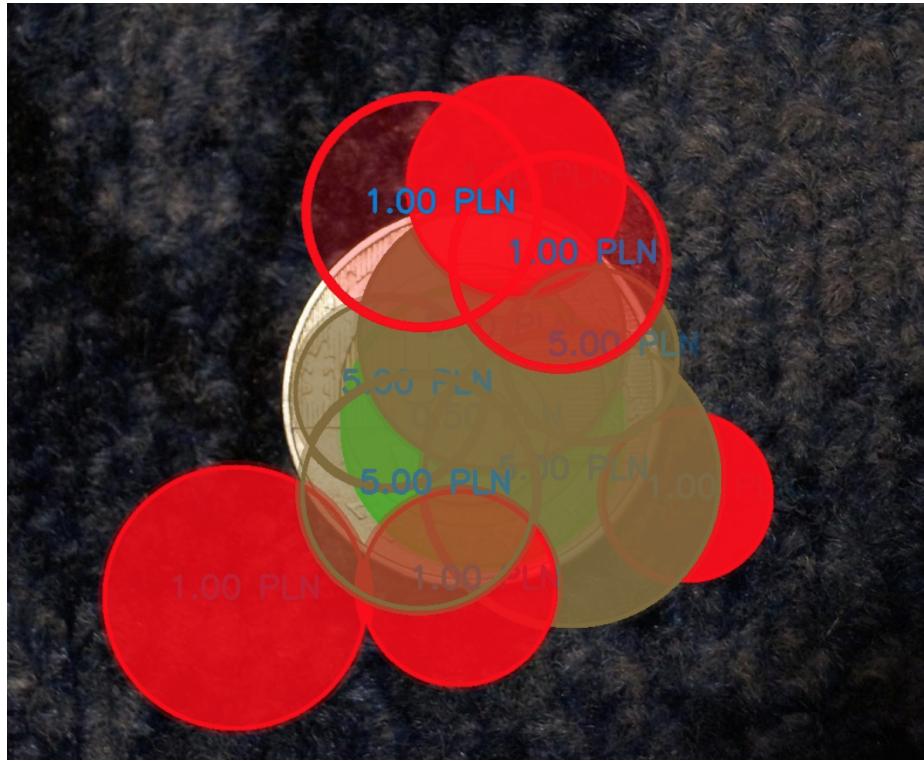
Na sam początek zaproponowano, aby odróżniac monety po kolorach, co w pierwotnej wersji okazało się bardzo słabe (a dokładniej implementacja tego pomysłu zawiodła). Przeglądając dalej dostępne funkcje w dokumentacji oraz prace naukowe o przetwarzaniu obrazów, chcieliśmy skupić się na różnych fakturach monet. Korzystając z różnej struktury monety (co szczególnie widoczne jest po zastosowaniu algorytmu detekcji krawędzi - *Sobel*) pragnęliśmy zastosować wiedzę o *momentach Hu* oraz *filtre Gabora* do analizy tekstuury. Niestety obydwa wyżej wymienione sposoby wymagają jasno określonego wzorca, który nie jest możliwy do osiągnięcia w naszych zastosowaniach. W przypadku monet, nawet sztucznie wygenerowane przekształcenia, nie były wystarczająco wierne w stosunku do oryginałów. Problem spowodowany był precyzją w wycinaniu obiektu z obrazu, gdyż różnica kilku pikseli znaczaco wpływała na wynik porównania.

Po analizie niepowodzenia z implementacją tak rozbudowanego porównania, postanowiliśmy zmienić sam sposób podejścia do wykrytego obiektu. Zamiast porównywać fakturę, powróciliśmy do sprawdzenia kolorów, ale z innym nastawieniem (na różnice między kolorami, a nie ich wartości w wybranej przestrzeni barw). Pozwoliło to na wyprowadzenie zależności dotyczących klasy obiektu.

## 4. Analiza algorytmu

### 4.1. Wyszukiwanie monet

Wyszukiwanie monet odbywa się dzięki użyciu funkcji *findContours* z biblioteki OpenCV. Wykorzystanie tej funkcji pozwala na wykrywanie kształtów, w tym okręgów, które są dla nas najbardziej istotne, poprzez funkcję *minEnclosingCircle*. Niestety sama funkcja zwraca bardzo wiele obiektów, które trzeba poddać analizie w celu eliminacji błędów. Sytuację bez tego kroku można zaobserwować poniżej.



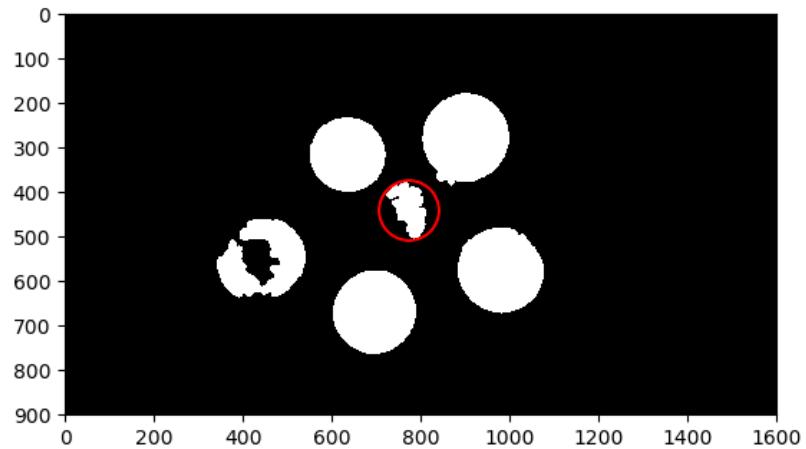
Rysunek 1. Zbyt wiele wykrytych okrągów

Aby tego uniknąć, w algorytmie zastosowano różne opcje wykrywania obrazu i jego parametrów. W zależności od średniej wartości kanału  $V$  w przestrzeni barw  $HSV$  algorytm wybiera odpowiedni preprocesing obrazu.

Dla ciemnych obrazów  $V_{avg} < 100$  wprowadza się dużą korekcję gamma 15 oraz skalowanie intensywności obrazu w zależności od percentylów. W tym miejscu otrzymujemy obraz binarny (czarne tło oraz białe obiekty). Następnie droga wykrywania kół rozchodzi się na dwie ścieżki. Jedna przetwarza obraz za pomocą operacji morfologicznych, a druga używa funkcji *Canny* oraz operacji morfologicznych. Wynik tych dwóch ścieżek jest łączony (operator  $\&$ ) i przekazywany do funkcji *findContours*.

Dla jasnych obrazów  $V_{avg} >= 100$  wprowadza się przeciwną korekcję gamma 0.7. Dalej algorytm korzysta z funkcji *adaptiveThreshold* z parametrem *cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C* oraz operacji morfologicznych. Wynik przekazywany jest do funkcji *findContours*.

Wykrywanie kół odbywa się poprzez funkcję *minEnclosingCircle* i sprawdzenie czy pole wykrytej figury nie jest zbyt małe, albo zbyt duże, oraz czy jest w przybliżeniu równe polu koła  $\Pi * r^2$ . Pozwala to na pominięcie kształtów, które zostały po wstępnym przetworzeniu obrazu, tak jak na obrazkach poniżej.



Rysunek 2. Wykryty kontur, ale nie spełniający wymagań koła



Rysunek 3. Końcowy rezultat, bez zbędnego konturu

#### 4.2. Wyszukiwanie banknotów

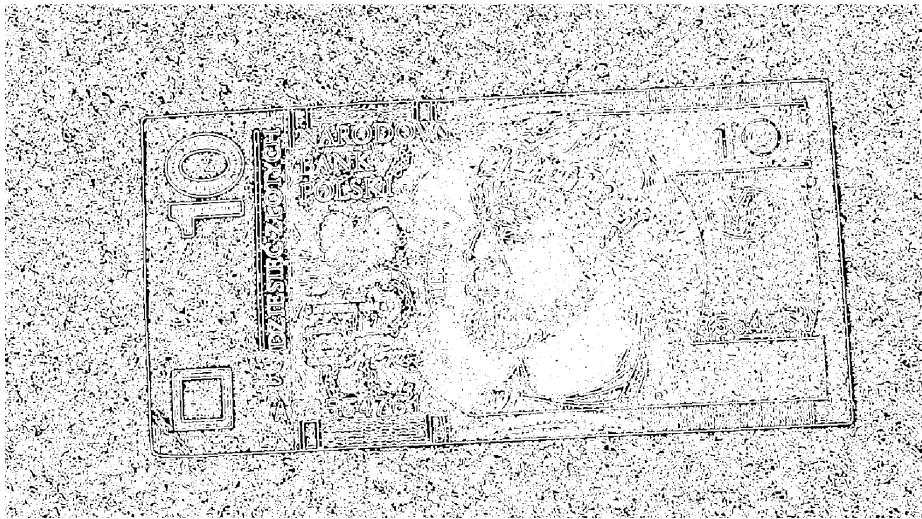
Poszukiwanie konturów, które mogą być potencjalnymi prostokątami zawierającymi banknoty, wymaga sprawdzenia wielu warunków.

Obraz podawany jest detekcji krawędzi *Canny*, z wartościami progowania  $\theta_1$ ,  $\theta_2$  oraz wielkością maski  $5 \times 5$  pikseli. Rezultat działania poddany jest dylatacji, dzięki której wyszczególniamy nasze krawędzi.

Banknot *10 PLN* w kolejnych krokach algorytmu.



Rysunek 4. Banknot 10 PLN (1)



Rysunek 5. Banknot 10 PLN (2)

Jak widać, ostatnie zdjęcie ułatwia wyszukanie konturów banknotu, gdyż stały się one szczególnie widoczne. Analizując poniższe zdjęcie, możemy zauważać, że bez dodatkowych zabezpieczeń w postaci sprawdzania kątów między krawędziami oraz dolnego limitu pola powierzchni, znaleziono by zbyt dużo obiektów. Po wprowadzeniu wymienionych warunków, otrzymujemy pożądany wynik.



Rysunek 6. Wykryte 10 PLN

#### 4.3. Sposób rozpoznawania monet

Wykorzystana została właściwość badanych obiektów, jakim u nas jest różnica w barwach wnętrza monety oraz jej obrzeża.

Dzięki takiemu postępowaniu możemy wydedukować, z którą z czterech kombinacji mamy do czynienia. Sukcesy w rozpoznawaniu monet zostają niestety ograniczone przez wykorzystaną funkcję. Wymaga ona podania minimalnego oraz maksymalnego promienia, które bez wcześniejszej ingerencji ze strony operatora mogą wyeliminować z obliczeń zdjęcia zawierające zbyt małe, bądź też zbyt duże obiekty.



Rysunek 7. Sposób wykrywania monet

Na początku wycinane jest wnętrze obiektu i poddawane analizie kolorów. Następnie ta sama czynność dotyczy przygotowanego pierścienia.

Każdy z pikseli innych od czarnego ( $R: 0, G: 0, B: 0$ ), który zostały dodane przez nas jako tło maski, jest analizowany. Obliczamy sumę wartości bezwzględnych różnic pomiędzy poszczególnymi częściami koloru tj.  $r - g$ ,  $r - b$ ,  $g - b$ . Suma ta jest dodawana do tablicy wartości częściowych, która na samym końcu zostaje uśredniona dzięki zastosowaniu *numpy.average*.

W drodze eksperymentu z przygotowanymi zdjęciami, wybrano następujące zakresy:

$$decyzja = \begin{cases} \text{zignorowanie} & \text{gdy średnia} < 20 \\ \text{srebro} & \text{gdy średnia} < 120 \\ \text{złoto} & \text{gdy średnia} \geq 120 \end{cases}$$

Kluczowym etapem jest poprawne przygotowanie elementu do analizy. Nieustety również taka implementacja jest wyjątkowo wrażliwa na prześwietlone obiekty, które człowiek rozpozna bez wielkiego problemu, a system rozpoznawania nie.

#### 4.4. Sposób rozpoznawania banknotów

Próby rozpoznania banknotów rozpoczęto podobnie jak w przypadku monet. Początkowe podejście z liczeniem średniej dla całego banknotu zostało szybko zarzucone, aby liczyć średnią tylko dla wybranego wycinka. Wykorzystano korzystną właściwość występowania na banknocie władcy Polski w różnym kolorze.



Rysunek 8. Rozpoznawanie banknotów (10 PLN)

W drodze eksperytmu z przygotowanymi zdjęciami, wybrano następujące zakresy dla wnioskowania odnośnie banknotu:

$$decyzja = \begin{cases} 10 \text{ PLN} & \text{gdy średnia} > 80 \\ 100 \text{ PLN} & \text{gdy średnia} > 40 \\ 50 \text{ PLN} & \text{gdy średnia} \leq 40 \end{cases}$$

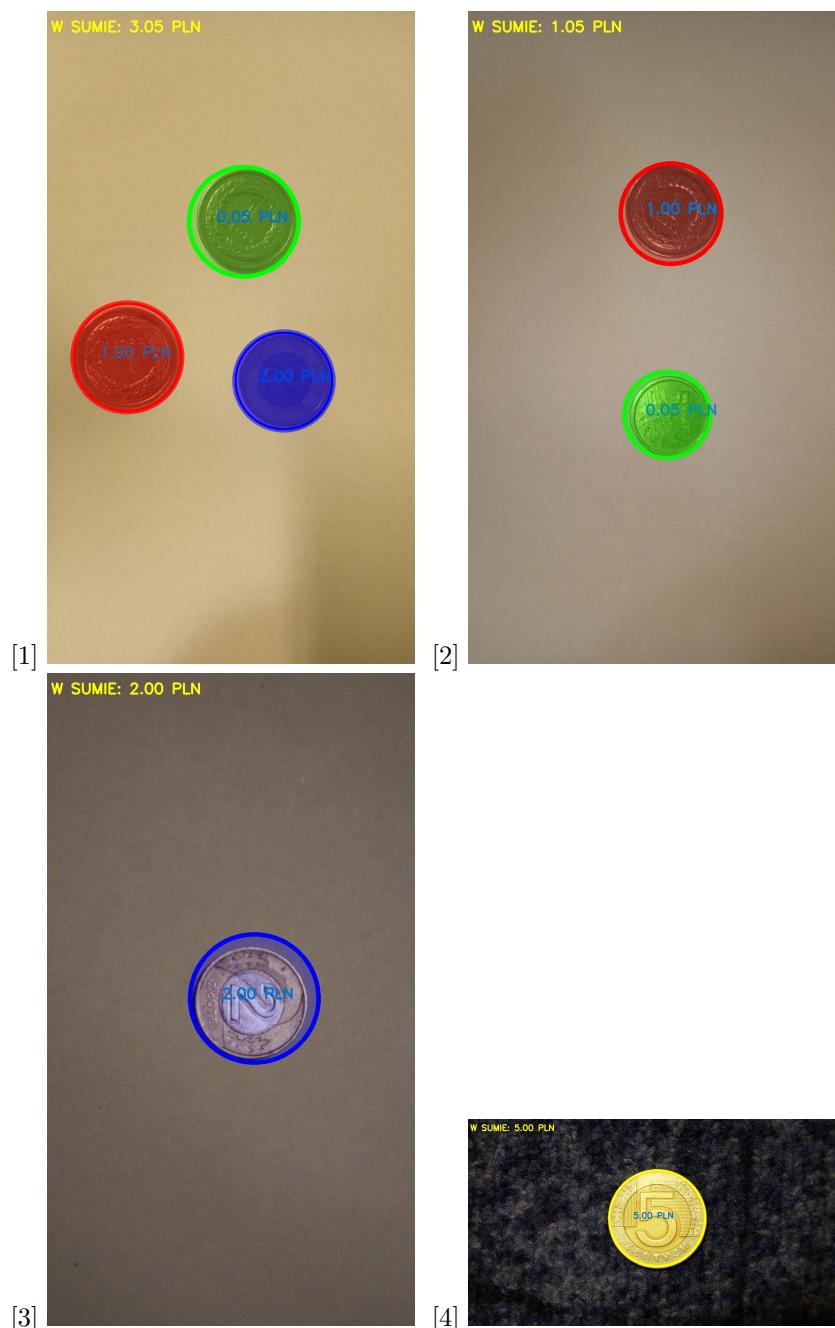
## 5. Wnioski

Analizując zachowanie się algorytmu dla wszystkich przygotowanych zdjęć, można zauważać jego niedoskonałości wymagające poprawy. Pierwszą rzeczą jest sposób wykrywania potencjalnych monet czy banknotów (ogólnie lepsze wykrywanie krawędzi tego czego chcemy). Następnie można by poprawić detekcję monet, ale jest to trudne ponieważ kolory zdjęcia są przerożne (nie mówiąc już o na przykład kolorowym światele z lampki).

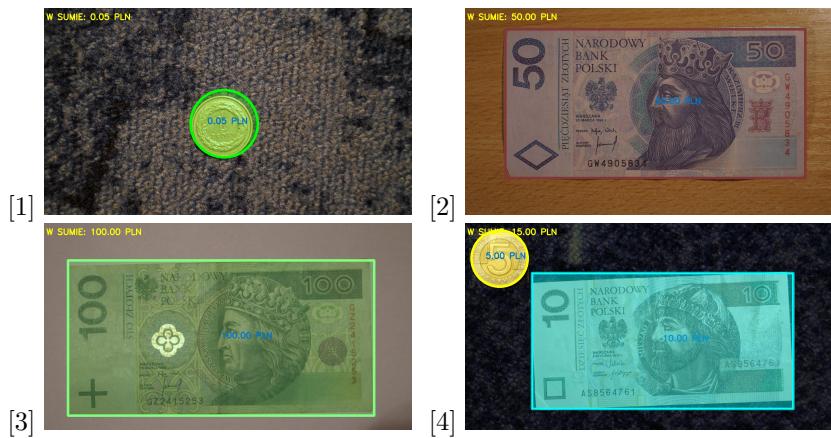
Prowadząc eksperymenty z ustawieniami programu, przygotowanymi zdjęciami testowymi czy zastosowanymi rozwiązaniami, można zauważać potrzebę użycia sztucznej inteligencji w projekcie. Rozpoznawanie obiektów dla człowieka wydaje się rzeczą naturalną, niestety dla komputera nie jest to trywialny problem. Próba podejścia do zagadnienia bazując wyłącznie na kolorach nie spełnia naszych oczekiwania, gdyż wiemy jak łatwo spreparować zdjęcia dla których algorytm odniesie sromotną porażkę.

## 6. Analizowane przypadki

### 6.1. Pomyślne rozpoznanie

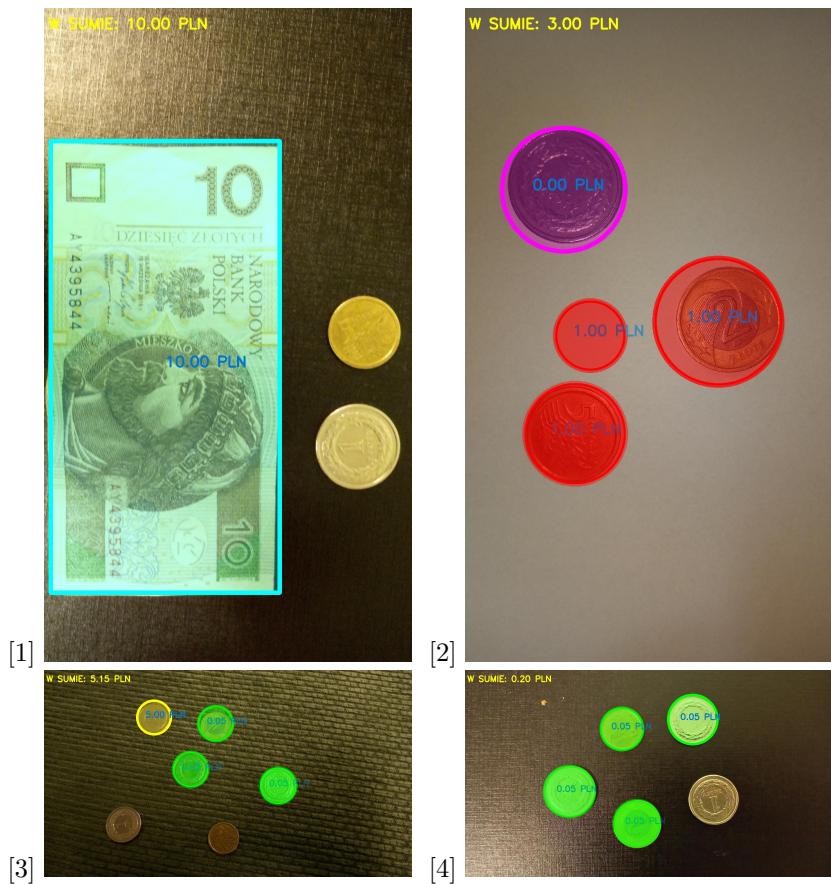


Rysunek 9. Pomyślnie rozpoznane obrazki (1)



Rysunek 10. Pomyślnie rozpoznane obrazki (2)

## 6.2. Niepomyślne rozpoznania



Rysunek 11. Błędnie rozpoznane obrazki

## 7. Bibliografia

Przygotując projekt, bazowaliśmy na wiedzy dostępnej w Internecie. Szczególnie przydatne okazały się:

- OpenCV documentation 3.0
- Skimage 0.14-dev documentation

— Stack Overflow  
— Uncle Google