ROZUMIENIE OBRAZÓW I SYGNAŁÓW

DETEKCJA PROSTYCH OBIEKTÓW

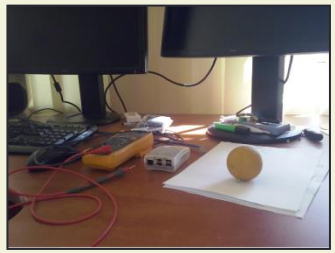
Krzysztof Ślot

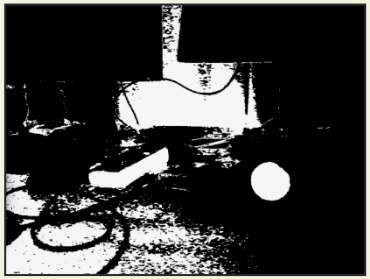
Instytut Informatyki Stosowanej

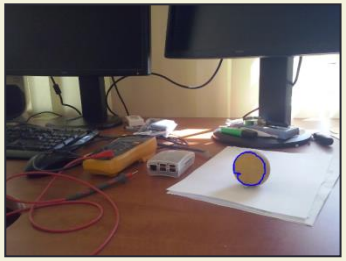
Blok Inteligentne Systemy Autonomiczne

Wprowadzenie

- Cel przetwarzania obrazów z perspektywy systemów inteligentnych (autonomicznych)
  - Rozpoznawanie treści obrazu (zawartość obrazu – jakie obiekty, jakie relacje między obiektami)
- Etapy przetwarzania
  - Pobranie obrazu
  - Przygotowanie do analizy
  - Analiza (detekcja, rozpoznanie)
  - Decyzja i akcja

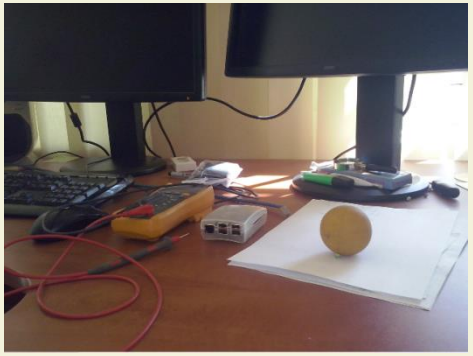


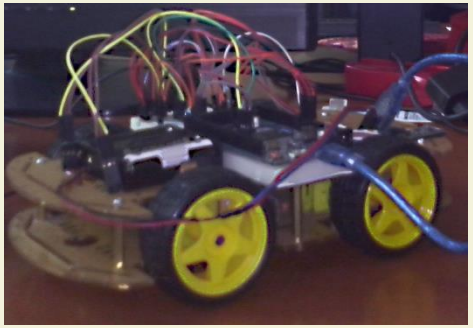




Wprowadzenie

- Scenariusze rozważane w kursie
  - Śledzenie „prostego” obiektu
    - „Prosty” – stały i regularny kształt, jednorodny kolor (znany /nieznany)
  - Śledzenie linii (wyznaczającej trasę)
    - Kolor linii znany / nieznany
    - Podłoże sztuczne / naturalne
  - Śledzenie obiektu rzeczywistego o stałej geometrii
    - Kształt obiektu się nie zmienia





Narzędzia

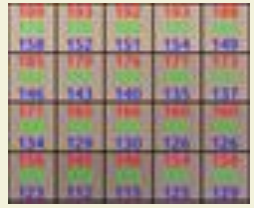
- Sprzęt
  - Raspberry Pi 3
  - Kamera do RPI (dedykowana / internetowa)
  - Pojazd kontrolowany przez Arduino
- Oprogramowanie
  - System operacyjny Raspbian
  - Środowisko Qt dla języka C++
  - Biblioteka OpenCV (wersja 3.x)
    - Moduły: `core`, `highgui` (GUI), `imgcodecs` (odczyt/zapis), `imgproc` (analiza), `features` (detekcja/rozpoznawanie),…

Obrazy i OpenCV

- Reprezentacja obrazów: macierz
  - Obiekt typu **cv::Mat** (tablica) o rozmiarze obrazu
  - Elementy tablicy

namespace („moduł”) cv :: - element

    - 1B 2B 4B/piksel (monochromatyczne): **CV_8U (char,uchar), CV_32F (float) CV_32S (int)**
    - 3B/piksel (kolorowe): **CV_8UC3 (cv::Vec3b)**
    - 4B/p (kolor + przezroczystość): CV_8U4
  - Typ - szablon klasy: wymaga specyfikacji typu elementów



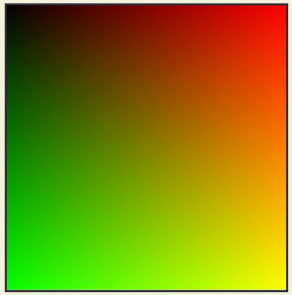
```
cv::Mat im1(640,480,CV_8U);        // obraz monochromatyczny, uchar
cv::Mat im2(1024,1024,CV_32F);     // monochromatyczny - float
cv::Mat im3(100,100,CV_8UC3);      // kolorowy - 3 kanały
```

Obrazy i OpenCV

- Manipulacja elementami obrazu
  - Zapis odczyt pojedynczych elementów: metody wymagają specyfikacji typu elementów (szablon)
  - Typy wpisywanych danych
    - proste (np. int gdy macierz ma jeden kanał)
    - wektory (wiele kanałów) `cv::Vec{2,3,4,6}{b,w,s,i,f,d}`

```
cv::Mat o(256,256,CV_8U);
  o.at<unsigned char>(10,10) = 200;          // wpisanie wartości
cv::Mat o2(200,100,CV_8UC3);
  o2.at<unsigned char>(10,10) = cv::Vec3b(10,20,30); // wpisanie wektora
```

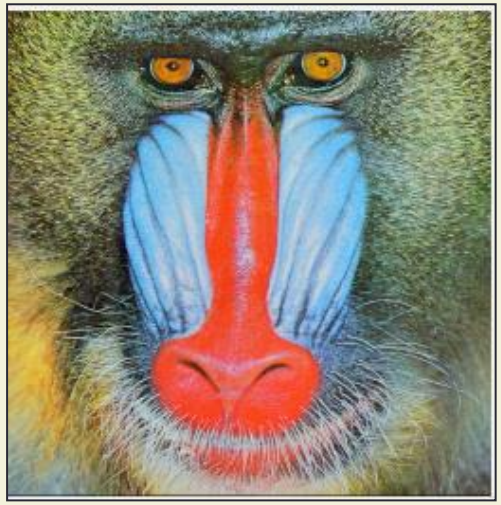
```
cv::Mat o(256,256,CV_8UC3);
  for(int i=0; i<o.rows; i++)        // zapełnianie macierzy
    for(int j=0; j<o.cols; j++)
      o.at<cv::Vec3b>(i,j)=cv::Vec3b(0,i,j);
```



Obrazy i OpenCV

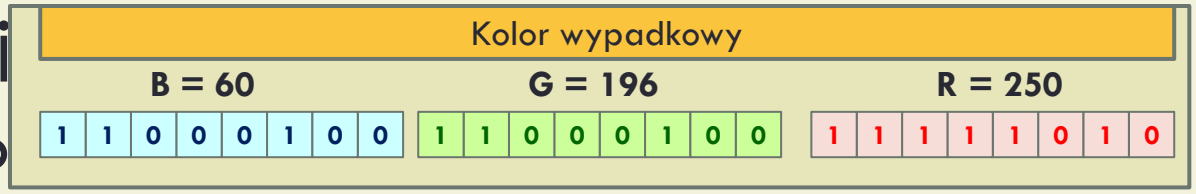
- Wczytywanie obrazów do programu
  - Źródło: plik / kamera
- Wczytywanie z pliku: `imread(..)` (moduł `Imgcodecs`)
  - Różne formaty obrazów (różne kodeki) – png, jpg, …
  - Automatyczna alokacja pamięci dla obrazu
  - Domyślna (kolor - BGR) lub narzucona reprezentacja pikseli
- Wyświetlenie obrazu: `imshow(..)`
  - moduł `highgui`

```
cv::Mat obraz; // tworzenie ‚pustego' obiektu
cv::imread(obraz,"../baboon.jpg);
cv::imshow(„Obraz",obraz);
```



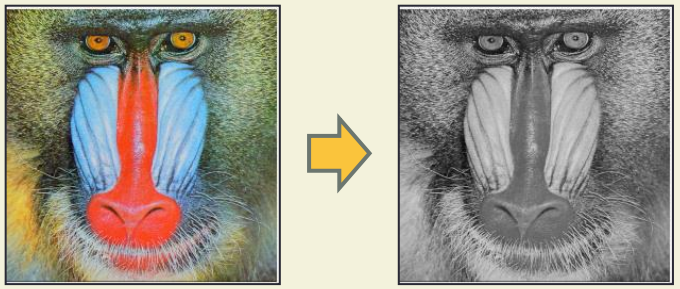
Obrazy i OpenCV

- Reprezentacj



  - 3Bpp, 4Bpp
    - Podstawa: reprezentacja **BGR** (BigEndian - RGB),
    - BGRA (ARGB), inne: przestrzenie kolorów HSV, Lab, …

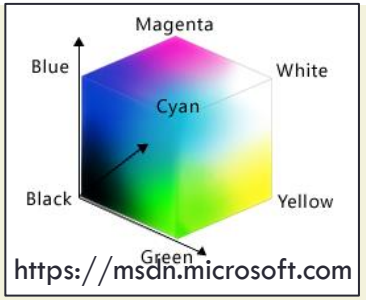
- Konwersje `cvtColor(..)`
  - Kolorowy – monochromatyczny (obraz mono to **podstawa** dla większości analiz!)



```
cv::Mat im = imread(„…”), gs, hsv;          // wczytanie obrazu
cv::cvtColor(im,gs,cv::COLOR_BGR2GRAY);     // konwersja do obrazu mono
cv::cvtColor(im,gs,cv::COLOR_BGR2HSV);      // konwersja do przestrzenie HSV
```

Przestrzenie kolorów

- Opis piksela kolorowego: wektor
  - Przestrzeń RGB: replikacja właściwości oka
  - HSV: barwa (H), nasycenie (S), jasność (V)



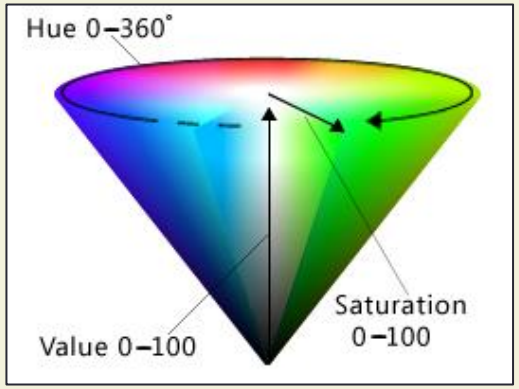
H — wzór eksperymentalny

S

$$S_{i,j} = \frac{\max(R_{i,j}, G_{i,j}, B_{i,j}) - \min(R_{i,j}, G_{i,j}, B_{i,j})}{\max(R_{i,j}, G_{i,j}, B_{i,j})}$$

V

$$V_{i,j} = \max(R_{i,j}, G_{i,j}, B_{i,j})$$



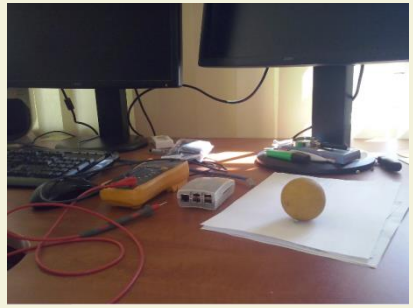
Analiza obrazów z OpenCV

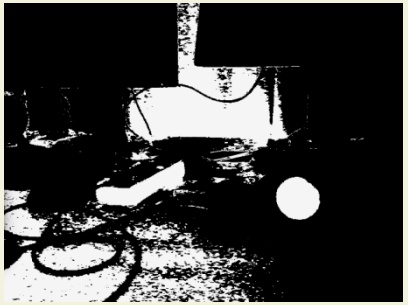
- Gotowe implementacje większości znanych procedur przetwarzania obrazów
  - Program to sekwencja wywołań odpowiednich funkcji
  - Konieczne twórcze rozwijanie tego schematu …
- OpenCV – biblioteka obiektowa (ale …)
  - Typowa postać funkcji: `void fun(src, dst, par1, …)`
  - Automatyczna alokacja zmiennej wynikowej
  - „Smart pointers”: `a=b` vs. `a=b.clone()`

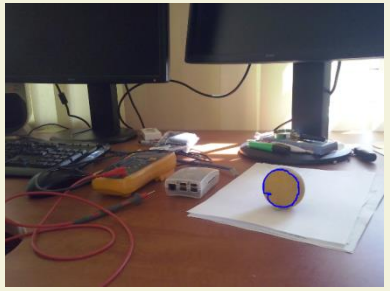
```
cv::Mat out;
cv::resize(in, out, Size(100,100));          // in: źródło ( wcześniej
cv::cvtColor(in, out,cv::COLOR_BGR2GRAY);    //zdefiniowane, out: wynik
cv::Canny(in, out, 50, 100);
```

Cel kursu: detekcja obiektów

- Obiekty ‚proste': kula, jednolity kolor
- Możliwy scenariusz
  - Wydziel obiekty o zbliżonym kolorze
  - Problem: może być ich kilka
- Rozbudowany scenariusz
  - Dodatkowo, określ kształt
- Zadania
  - **Segmentacja** obrazu względem koloru
  - Analiza kształtu obiektu
- Metodyka: segmentacja i rozpoznawanie kształtu

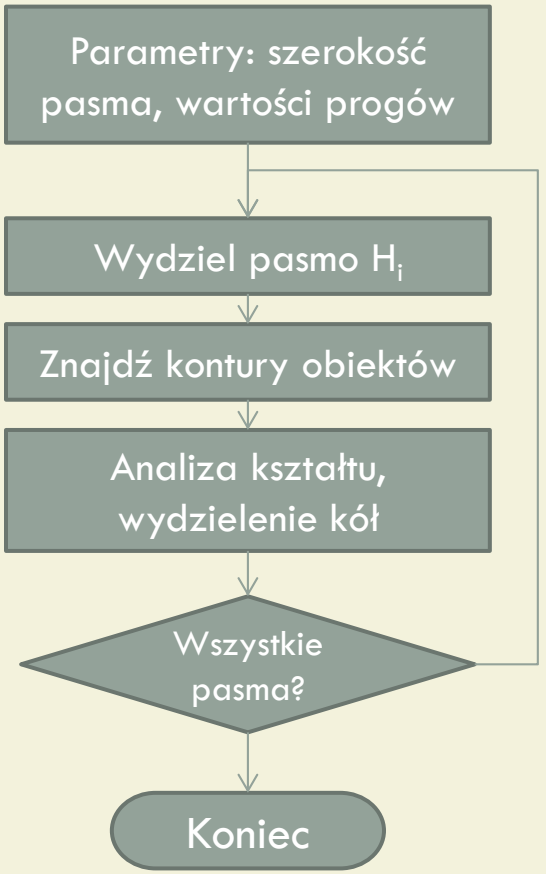






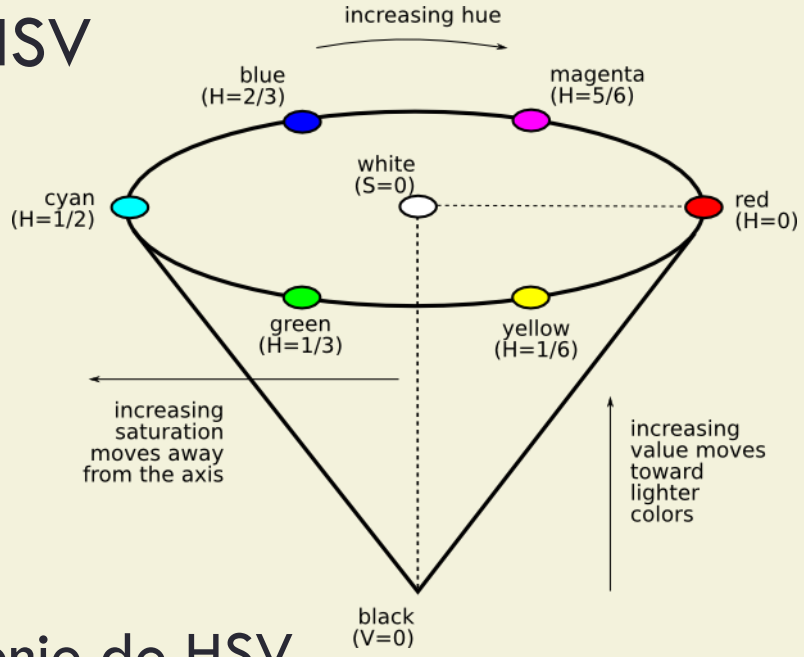
Procedura analizy: detekcja kul

- Cel: detekcja kul o jednorodnym kolorze
- Segmentacja obrazu w funkcji koloru
  - Przedziały $[H_l .. H_u]$
  - Zakres: kompromis (zmienność – selektywność)
- Detekcja konturów obiektów kolejnych pasm
  - Dane: obraz binarny (obiekty i tło)
  - Kontur – punkty brzegowe: ekstrakcja
  - Istotne tylko kontury zewnętrzne obiektów
- Analiza kształtu (cel: znalezienie okręgów)
  - Dane: zbiór punktów (krzywa)
  - Cel: porównanie podobieństwa do okręgu
  - Modele okręgu



Segmentacja względem koloru

- Najwygodniejsza przestrzeń: HSV
- Segmentacja
  - Podział obrazu na kategorie o spójnych właściwościach
  - Kryterium: kolor

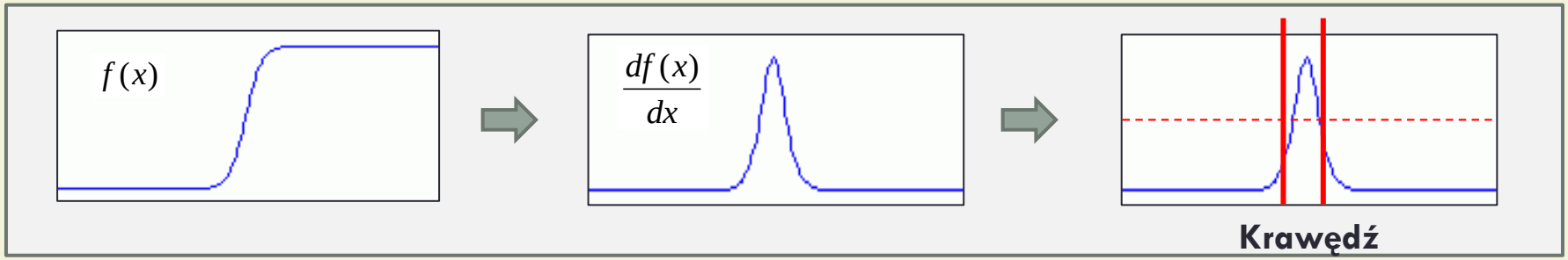


- Algorytm postępowania
  - Wczytanie obrazu – przekształcenie do HSV
  - Wydzielenie zakresu barw

```
cv::Mat img = cv::imread(„…”);
cv::Mat hsv, Seg;
cv::cvtColor(img,hsc,cv::COLOR_BGR2HSV);
cv::inRange(hsv, cv::Scalar(l,100, 100), cv::Scalar(u,255,255), Seg);
```

Detekcja krawędzi i konturów

- Krawędzie: wyraziste gradienty jasności / barwy
  - Kluczowe znaczenie dla analizy obrazu: granice obiektów, wygląd obiektu
  - Detekcja: poszukiwanie lokalnych maksimów zmienności
    - Pochodna (zmienność) + progowanie modułu pochodnej



- Numeryczna aproksymacja pochodnej

$$\frac{df(x)}{dx} \approx \frac{f(x+h) - f(x-h)}{h} \quad \Rightarrow \quad h = 1 \rightarrow f(x+1) - f(x-1)$$

Detekcja krawędzi i konturów

- Liniowe przekształcenia funkcji
  - Splot funkcji f i H
  - Obejmuje wyznaczanie pochodnej

$$g(k) = \sum_{i=-n}^{n} f_{k-i} * H_i \qquad f = [f_a \dots f_b] \qquad H = [H_{-n} \dots H_n]$$

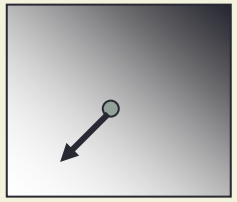
$$g(k) = f * H$$

- Przekształcanie funkcji w celu uwypuklenia / redukcji lokalnych właściwości: filtracja
  - Detekcja krawędzi: uwypuklenie zmienności
  - Wygładzanie: redukcja zmienności

$$H = [-1,0,1]$$

$$H = [0.5,0,0.5]$$

- Dziedzina 2D
  - Pochodna ⇨ gradient



$$H_V = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

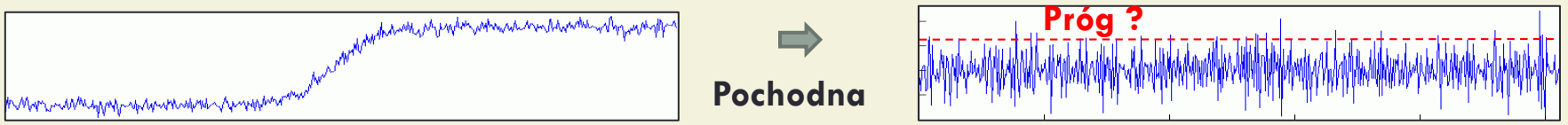
$$H_H = [-1,0,1]$$

$$\nabla f(x,y) = \left[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right]$$

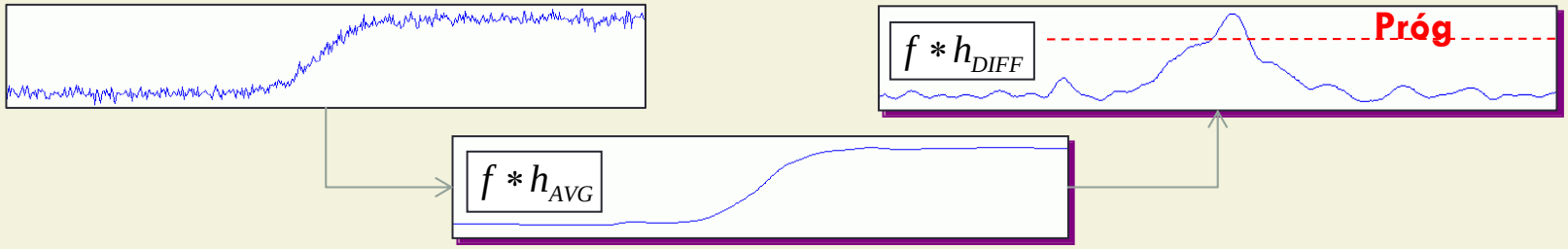
$$\| \nabla f \| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Detekcja krawędzi i konturów

- Obrazy rzeczywiste
  - Typowy przebieg linii obrazu (błędy akwizycji, zakłócenia)



- Konieczna eliminacja/redukcja lokalnej zmienności
  - Powiązanie wygładzania (uśredniania) z różniczkowaniem



- Operacje liniowe – realizacja obydwu operacji w jednym etapie

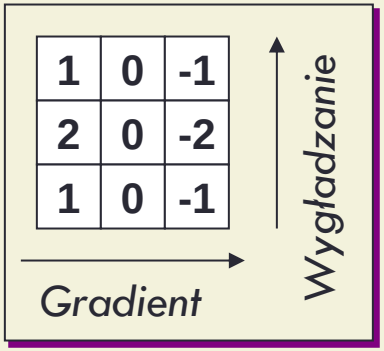
Detekcja krawędzi i konturów

- Detekcja krawędzi: maski Sobela cv::Sobel
  - Dwuwymiarowy operator H: wygładzanie + krawędzie

Uśrednienie $\mathbf{L} = [1,2,1]$ $\quad$ $\mathbf{H} = [-1,0,1]$ **Pochodna**

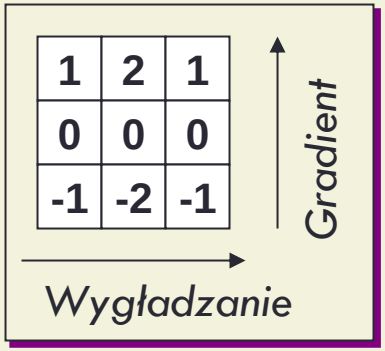
Maska pozioma

$S_H = H\ L^T$



Maska pionowa

$S_V = L\ H^T$



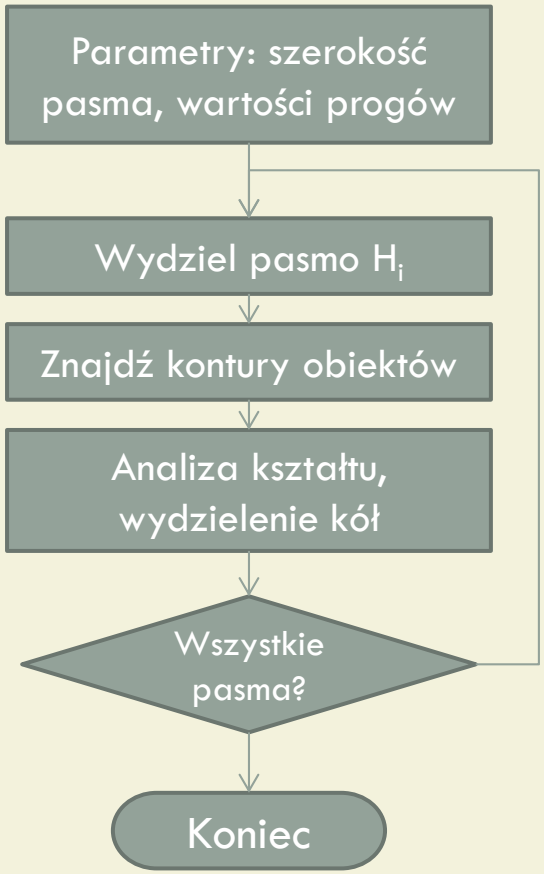
- Detekcja krawędzi: metoda Canny'ego
  - Rozwinięcie filtracji Sobela
  - Dodany mechanizm śledzenia konturów

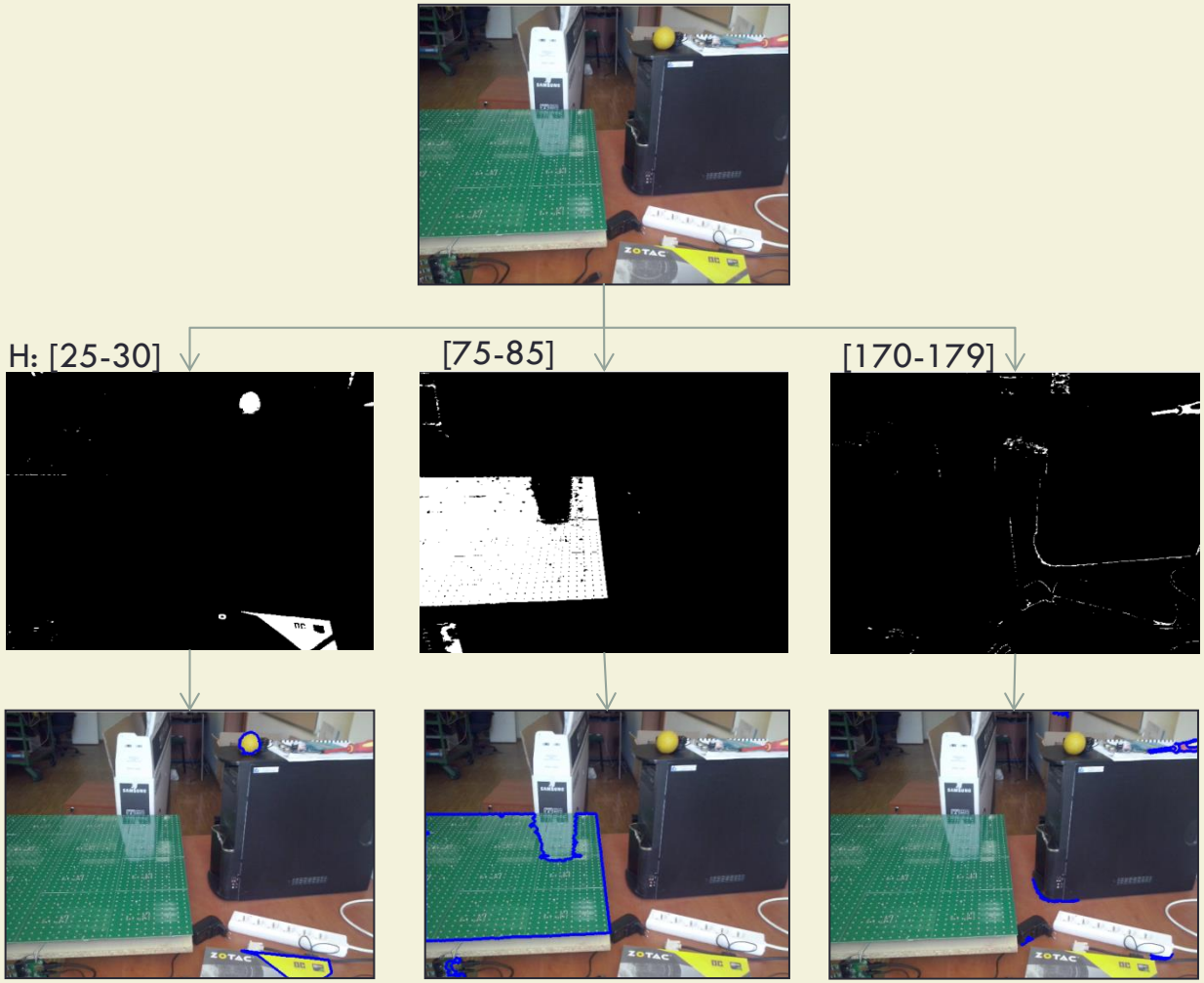
Detekcja konturów

- Kontury cv::findContours
  - Argument to obraz binarny (np. wydzielony w wyniku operacji lub zawierający wynik detekcji krawędzi)
  - Wynik to lista konturów, z których każdy zawiera listę należących do niego punktów)

- Istotne opcje cv::findContours
  - Możliwe uzyskanie konturu zewnętrznego obiektu lub wszystkich (wewnętrznych z informacją o relacji wzajemnego zawierania)
  - Możliwe kodowanie konturu lub brak kodowania (wynik to wszystkie punkty konturu)

Procedura analizy: detekcja kul





Analiza kształtu

- Rozpoznanie – porównanie z modelem - ocena
  - Model: analityczny lub zbudowany na bazie przykładów
  - Porównanie: procedura generująca ilościowy wynik oceniający dopasowanie
  - Wynik podlega ocenie – próg podobieństwa
- Możliwe modele kształtu obiektu
  - Specyficzne – bazujące na wiedzy o kształcie figury
    - Analityczny model okręgu
  - Ogólne – statystyczne z parametrami
    - ‚Momenty' obiektu: zwykłe, centralne, unormowane
  - Ogólne – bazujące na cechach lokalnych

Modele analityczne

- Analityczne definicje obiektu
  - Okrąg – zbiór punktów jednakowo odległych od środka

$$\forall_{P_i \in K_j} : (x_i - a)^2 + (y_i - b)^2 = r^2$$

  - Okrąg – krzywa parametryczna

$$\forall_{P_i \in K_j} : \begin{cases} x_i = r \sin(i/r) \\ y_i = r \cos(i/r) \end{cases}$$

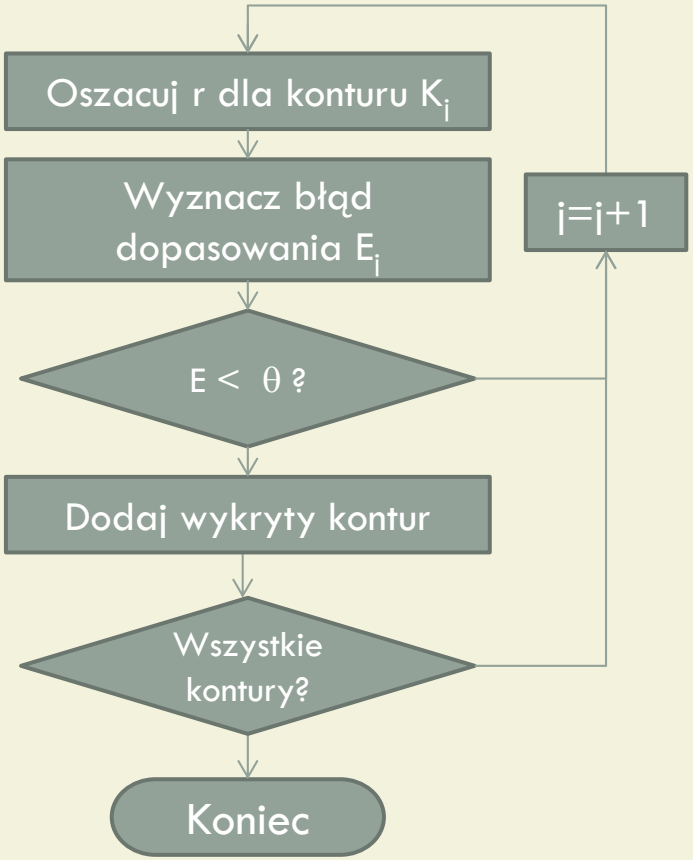
- Ocena zgodności konturu z modelem #1
  - Kryterium: średni błąd odstępstw punktów konturu od modelu (kontur ‚wycentrowany': $a=b=0$)
  - Konieczne oszacowanie ‚optymalnego' promienia okręgu

$$E = \frac{1}{n} \sum_{i=0}^{n-1} (x_i^2 + y_i^2 - r^2)^2$$

$$\frac{\partial E}{\partial r} = 0 \rightarrow r^2 = \frac{1}{n} \sum_{i=0}^{n-1} (x_i^2 + y_i^2)$$

Modele analityczne

- Procedura detekcji
  - Próg detekcji: wartość błędu uznana za akceptowalną
  - Wynik: lista ostateczna lub lista kandydatów do dalszego sprawdzenia
  - Dalsze sprawdzenie: użycie innego kryterium (np. #2)



Modele statystyczne

- Statystyczne właściwości figury (konturu)
  - Momenty zwykłe ($I(x,y)$ – jasność w punkcie (x,y))
  - Momenty centralne (względem wartości średniej)

$$m_{pq} = \sum_{i=0}^{n-1} I(x,y) x^p y^q$$

$$\mu_{pq} = \sum_{i=0}^{n-1} I(x,y)(x-\bar{x})^p (y-\bar{y})^q$$

- Możliwe statystyczne kryterium detekcji
  - Momenty centralne μ_{20} i μ_{02} powinny być podobne
  - Brak liniowej zależności współrzędnych okręgu: $\mu_{11}=0$

$$J = \frac{(\mu_{20}/\mu_{02} - 1)^2}{1 - \mu_{11}^2}$$