Sprawozdanie

Metody Numeryczne 2022L
Elektromobilność 03.06.2022
Semestr II

Zespół 6:

Bernard Kościewicz
Michał Wołowicz
Hubert Żyliński

Temat projektu:

Zastąpienie wbudowanej do środowiska MatLab funkcji 'imbinarize' poprzez własnoręcznie napisaną funkcję „OtsuMtd", która jest potrzebna wbudowanej funkcji „imbanarize", dla wykonania algorytmu sterującego robotem. Między innymi dzięki niej mikrokontroler sterujący ramieniem, jest w stanie, rozpoznać kolory z otrzymanych obrazów z kamery.

Opis napisanej funkcji:

Napisana przez nas funkcja opiera się na bisekcji oraz metodzie Otsu, która pozwala ze zdjęcia 'grayscale' wyróżnić obiekt na tle. Jest to możliwe dzięki wyznaczeniu wartości progowej wśród pikseli na obrazie, która służy do segregowania pozostałych pikseli między dwoma klasami. W naszym przypadku są to Obiekt (np. odcisk palca, budynek), oraz Tło (np. biała kartka, niebo). Przynależność zaś danego piksela pobranej przez program grafiki do jednej z dwóch klas jest ustalane na podstawie jego intensywności światła.

Metoda Otsu, która rysuje wykres z różnicy międzyklasowej, minimalizując wariancję progowanych pikseli czarnych i białych, z tej samej klasy, zdefiniowana jest jako suma ważona wariancji dwóch klas:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Gdzie: $'\boldsymbol{\omega_0}'$ i $'\boldsymbol{\omega_1}'$ – Są wagami prawdopodobieństw dwóch klas oddzielonych progiem $'\boldsymbol{t}'$, a $'\boldsymbol{\sigma_0^2}'$ i $'\boldsymbol{\sigma_1^2}'$ są wariancjami tych dwóch klas.

Wykres powstaje przy wykorzystaniu 256-bitowego histogramu obrazu uzyskany przez wykorzystanie wbudowanej do środowiska MatLab funkcji 'imhist()'. Można przy tym iteracyjne obliczać prawdopodobieństwa klas i środków, co pozwala uzyskać wydajny algorytm:

1. Oblicz histogram i prawdopodobieństwo dla każdego poziomu intensywności
2. Ustaw początkowe $'\boldsymbol{\omega_i(0)}'$ oraz $'\boldsymbol{\mu_i(0)}'$
3. Przejdź przez wszystkie możliwe progi $'\boldsymbol{t = 1, 2, \ldots}'$ maksymalnej intensywności
    a. Zaktualizuj $'\boldsymbol{\omega_i}'$ i $'\boldsymbol{\mu_i}'$
    b. Oblicz $'\boldsymbol{\sigma_b^2}(t)'$
4. Pożądany próg odpowiada maksimum $'\boldsymbol{\sigma_b^2}(t)'$

Tą wartość maksymalną spośród uzyskanych $'\boldsymbol{\sigma_b^2}(t)'$, pozwala nam uzyskać metoda bisekcji, która jest zastosowana na pochodnej z powyższej funkcji, ponieważ otrzymujemy punkt, w którym ona się zeruje. Jest to ekstremum dla wykresu, czyli jego wartość maksymalna tudzież szukany próg (ang.: threshold: T)

```
function [t,em] = OtsuMtd(calcs)

% Potwierdzenie, że tablica A należy do co najmniej jednej z
podanych KLAS i zawiera wszystkie podane ATRYBUTY.

validateattributes(elems, {'numeric'},
{'real','nonsparse','vector','nonnegative','finite'}, mfilename,
'COUNTS');

calcs = double(calcs(:));
num_bins = numel(calcs);
p = calcs / sum(calcs);
omega = cumsum(p);
mu = cumsum(p .* (1:num_bins)');
mu_t = mu(end);

%Wzór do poszukiwania progu minimalizującego wariancję
wewnątrzklasową, zdefiniowaną jako suma ważona wariancji dwóch klas:

sigma_b_squared = (mu_t * omega - mu).^2 ./ (omega .* (1 - omega));

% Zlokalizuj maksymalną wartość sigma_b_squared.

% Dodaj wszystkie lokalizacje razem, ponieważ maksimum może
obejmować wiele pikseli.

% Jeśli maxval jest NaN, sigma_b_squared będzie cała NaN, i bisekcja
zwróci wartość 0.

drvt_sigm = diff(sigma_b_squared);

xtrem_max = BisectMtd(drvt_sigm);

isfinite_maxval = isfinite(xtrem_max);

if isfinite_maxval

idx = mean(find(sigma_b_squared == xtrem_max));

% Normalizacja progu do zakresu [0, 1].

t = (idx - 1) / (num_bins - 1);

else

t = 0.0;

end

% Obliczenie skuteczności działań

if nargout > 1
if isfinite_maxval
em = xtrem_max/(sum(p.*((1:num_bins).^2)') - mu_t^2);
```

```
else
em = 0;
end
end
end
```

Wartość maksymalną uzyskujemy przez użycie pomocnej funkcji „BisekcjaMtd" na pochodnej z zaimplementowanej już funkcji dla $'\boldsymbol{\sigma_b^2}(t)'$, ta stosuje metodę bisekcji do znalezienia punkt '0' (=pierwiastek z funkcji pochodnej) podanego jej wektora wartości wykresu. Odbywa się iteracja po wektorze: algorytm dzieli zakres iteracji na dwie równe części wyznaczając środek tego zakresu. Następnie sprawdza po której stronie wyznaczonego środka iloczyn wartości granicznych będzie mniejszy od 0. Wartości graniczne tej strony stają się nowymi granicami zakresu. W ten sposób funkcja jest w stanie wyznaczyć wartość maksymalną, w której różnica między 'Tłem' a 'Obiektem' będzie największa. Odpowiadająca tej wartości maksymalnej intensywność piksela jest szukanym przez nas Otsu threshold = T.

```
function [x_0] = BisectMtd(x_array, values_array)
if values_array(1) == 0
x_0 = x_array(1);
return
elseif values_array(end) == 0
x_0 = x_array(1);
return
end
a = length(x_array);
mid = ceil(a/2);
check_left = values_array(1)*values_array(mid);
check_right = values_array(end)*values_array(mid);
if check_left > 0 && check_right >0
x_0 = nan;
return
end
if a < 3
x_0 = x_array(a);
return
end
if values_array(mid) == 0 || values_array(mid-1) == 0 ||
values_array(mid+1) == 0
x_0 = x_array(mid);
return
else
if check_left < 0
x_array = x_array(1:mid);
values_array = values_array(1:mid);
elseif check_right <0
x_array = x_array(mid:a);
values_array = values_array(mid:a);
end
x_0 = BisectMtd(x_array,values_array);
end
end
```

Implementacja tych funkcji pomocniczych do reszty kodu funkcji ImbinarizeMtd', która jest stworzona przez podmienienie wbudowanego 'imbinarize' (podkreślenie na żółto). Pozwala ona na przekształcenie danej algorytmowi grafiki na obraz binarny (czarno-biały).

```
% Notatki

% -----

% W metodzie "globalnej" do obliczenia progu Otsu wykorzystuje się
histogram obrazu z 256 komór.

% Aby wygenerować obraz binarny z obrazu indeksowanego, należy
najpierw przekonwertować obraz na obraz intensywności za pomocą
narzędzia IND2GRAY. Aby utworzyć obraz intensywności w skali
szarości z obrazu RGB, należy najpierw użyć narzędzia RGB2GRAY do
przekształcenia obrazu na obraz intensywności w skali szarości.

% IMBINARIZE przewiduje, że obrazy zmiennoprzecinkowe będą
znormalizowane w zakresie [0,1].

function BW = ImbinarizeMtd(I, varargin)
varargin = matlab.images.internal.stringToChar(varargin);
[I,isNumericThreshold,options] = parseInputs(I,varargin{:});
if isNumericThreshold
T = options.T;
else
method = options.Method;
if strcmp(method,'global')
T = computeGlobalThreshold(I);
else
sensitivity = options.Sensitivity;
fgPolarity = options.ForegroundPolarity;
T = adaptthresh(I,sensitivity,'ForegroundPolarity',fgPolarity);
end
end
% Binaryzacja obrazu przy użyciu obliczonego progu
BW = binarize(I,T);
end
function BW = binarize(I,T)
classrange = getrangefromclass(I);
switch class(I)
case {'uint8','uint16','uint32'}
BW = I > T*classrange(2);
case {'int8','int16','int32'}
BW = I > classrange(1) + (classrange(2)-classrange(1))*T;
case {'single','double'}
BW = I > T;
end
end
function T = computeGlobalThreshold(I)
if isfloat(I)
I = im2uint8(I);
T = Otsu_mtd( imhist(I) );
else
T = Otsu_mtd( imhist(I) );
```

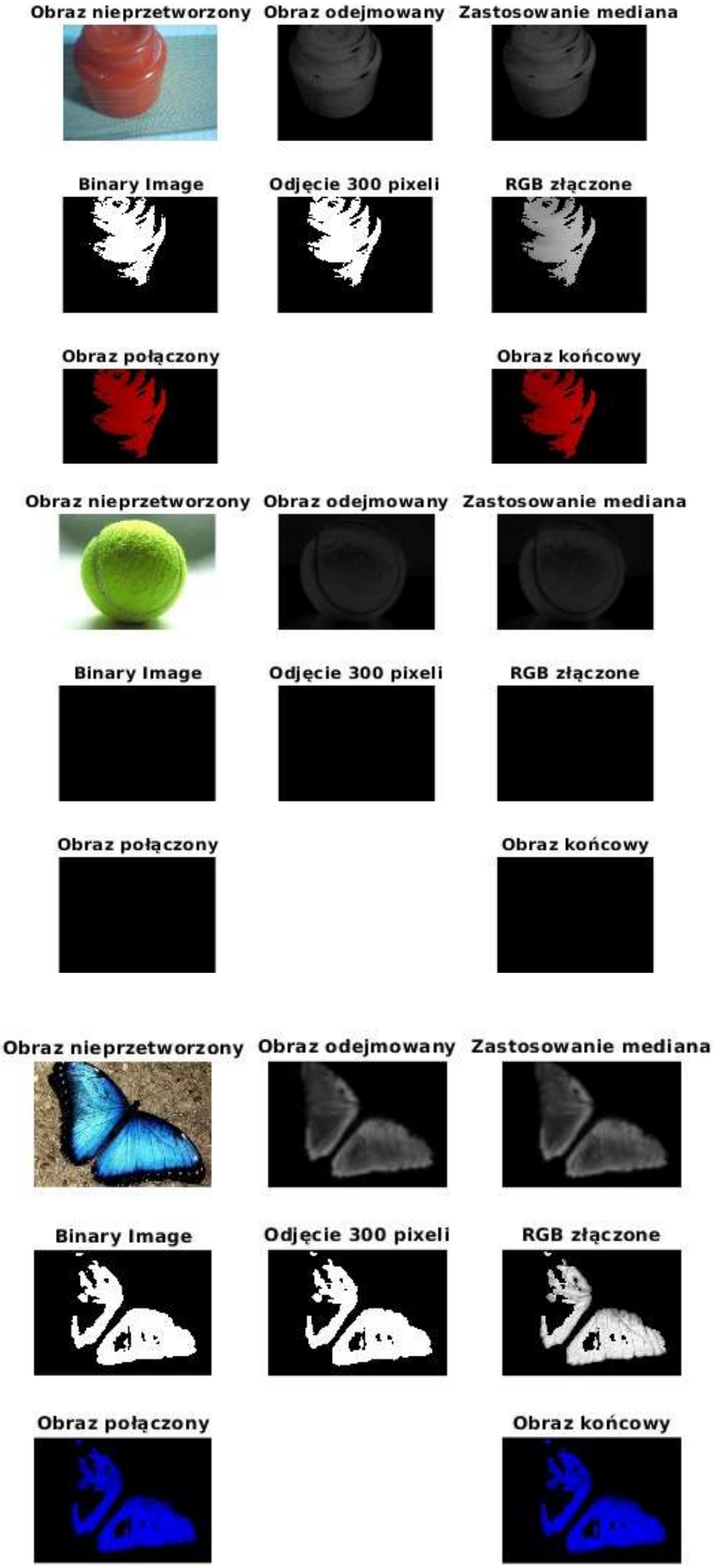
```
end
end

% Analiza wprowadzanych danych
function [I,isNumericThreshold,options] = parseInputs(I, varargin)
narginchk(1,6);

% validate image
validateImage(I);
isNumericThreshold = ~isempty(varargin) && ~ischar(varargin{1});
if isNumericThreshold
options.T = validateT(varargin{1},size(I));
if numel(varargin)>1
error(message('MATLAB:TooManyInputs'))
end
else
if isempty(varargin)
options.Method = 'global';
return;
end
options.Method =
validatestring(varargin{1},{'global','adaptive'},mfilename,'Method',
2);
if strcmp(options.Method,'global')
if numel(varargin)>1
error(message('MATLAB:TooManyInputs'))
end
else
options.Sensitivity = 0.5;
options.ForegroundPolarity = 'bright';
numPVArgs = numel(varargin)-1;
if mod(numPVArgs,2)~=0
error(message('images:validate:invalidNameValue'));
end
ParamNames = {'Sensitivity','ForegroundPolarity'};
ValidateFcn = {@validateSensitivity,@validateForegroundPolarity};
for p = 2 : 2 : numel(varargin)-1
Name = varargin{p};
Value = varargin{p+1};
idx = strncmpi(Name, ParamNames, numel(Name));
if ~any(idx)
error(message('images:validate:unknownParamName', Name));
elseif numel(find(idx))>1
error(message('images:validate:ambiguousParamName', Name));
end
validate = ValidateFcn{idx};
options.(ParamNames{idx}) = validate(Value);
end
end
end
end
function validateImage(I)
```

```
supportedClasses =
{'uint8','uint16','uint32','int8','int16','int32','single','double'}
;
supportedAttribs = {'real','nonsparse','3d'};
validateattributes(I,supportedClasses,supportedAttribs,mfilename,'I'
);
end
function T = validateT(T,sizeI)
validateattributes(T,{'numeric'},{'real','nonsparse','3d'},mfilename
,'Threshold',2);
if ~( isscalar(T) || isequal(size(T),sizeI) )
error(message('images:imbinarize:badSizedThreshold'))
end
end
function s = validateSensitivity(s)
validateattributes(s,{'numeric'},{'real','nonsparse','scalar','nonne
gative','<=',1},mfilename,'Sensitivity');
end
function fgp = validateForegroundPolarity(fgp)
fgp =
validatestring(fgp,{'bright','dark'},mfilename,'ForegroundPolarity')
;
end
```

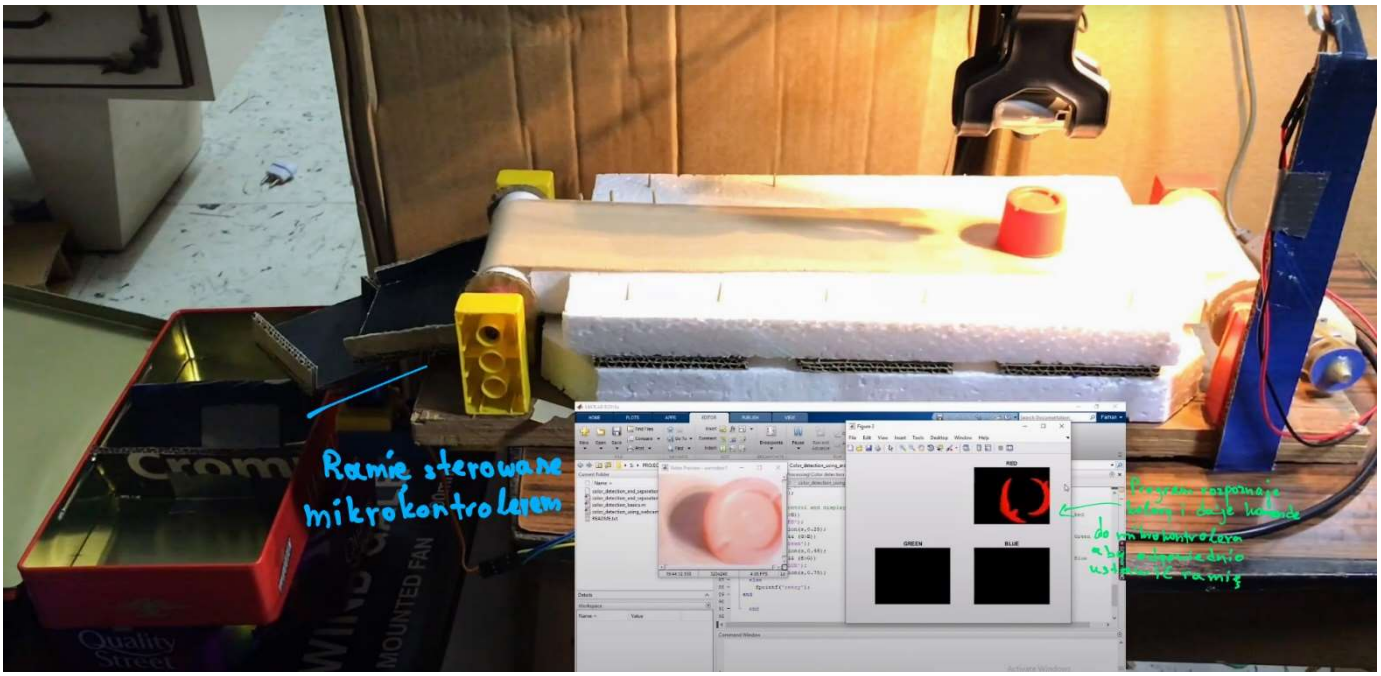
Przedstawiony poniżej rezultat uruchomienia całego algorytmu (z implementacją naszej funkcji) pokrywa się z wynikiem uzyskanym przez wywołanie oryginalnego algorytmu z wbudowaną do środowiska MatLab funkcją ‘imbinarize()’. Potwierdza to poprawność działania napisanej przez nas funkcji.

```
% Uzyskanie od użytkownika danych w postaci obrazu.
 m='red3.jpg';          % Obrazek jest zapisywany w zmiennej.
                        % Sprawdź, czy obrazek istnieje w bieżącym
katalogu
 im= imread(m);
 subplot(3,3,1);        % Do wyświetlania obrazu używana jest macierz
2x2, @ POS(1,1).
 imshow(m);
 title('Obraz nieprzetworzony');
% Wykrywanie koloru czerwonego
  r=im(:,:,1); g=im(:,:,2); b=im(:,:,3);
  diff_red=imsubtract(r,rgb2gray(im));   % Wyodrębnianie czerwonych
obiektów z obrazu w odcieniach szarości
  subplot(3,3,2);
  imshow(diff_red);

  title('Obraz odejmowany');
  diff_r=medfilt2(diff_red,[3 3]);       % Zastosowanie mediana
  subplot(3,3,3);
  imshow(diff_r);

  title('Zastosowanie mediana');
  bw_r=ImbinarizeMtd(diff_r,0.2);    % Konwersja obrazu w skali szarości na
obraz czarno-biały z wartością progową 0,2.
  subplot(3,3,4);
  imshow(bw_r);

  title('Binary Image');
  area_r=bwareaopen(bw_r,300);           % Aby usunąć obiekty o rozmiarze
mniejszym niż 300 pikseli
  subplot(3,3,5);
  imshow(area_r);

  title('Odjęcie 300 pixeli');
  R=sum(area_r(:));                          % Wykorzystanie jako funkcji
segregacji
  rm=immultiply(area_r,r);  gm=g.*0;  bm=b.*0; % Mnoży kolor czerwony
wykrytego obiektu z kolor czerwonym w celu wizualizacji.
  subplot(3,3,6);
  imshow(rm);

  title('RGB złączone');
  image_r=cat(3,rm,gm,bm);                   %łączy wszystkie obrazy RGB
  subplot(3,3,7);
  imshow(image_r);

  title('Obraz połączony');
  subplot(3,3,9);
  imshow(image_r);

  title('Obraz końcowy');
```

Rezultat użycia naszej funkcji ‘ImbinarizeMtd’ znajduje się na grafice powyżej:



Natomiast implementacja naszej funkcji do całego algorytmu sterującego robotem, pozwala na uzyskanie następującego rezultatu, które z przyczyn ograniczeń czasowych nie możemy przedstawić fizycznie:



Wnioski

Projekt nam przybliżył rzeczywistość implementacji funkcji z Metod Numerycznych w algorytm pracy robota. Przez zrozumienie zasady działania Metody Otsu, mogliśmy określić na jakim jej etapie może być wprowadzona bisekcja. Określenie wartości progowej dla intensywności światła, dzięki której program rozróżnia Obiekt od Tła daje nam lepsze zrozumienie działania sztucznej inteligencji.

Przy testowaniu działania naszego kodu, wynikły pewne ograniczenia metody Otsu:

Badana grafika musi posiadać duży kontrast (wyraźną różnicę odcieni/kolorów) pomiędzy obiektem a tłem, aby otrzymane wyniki były zadowalające. Przykład tego ograniczenia można zaobserwować na wyników otrzymanych dla grafiki zielonej piłki tenisowej na białym tle.

Z naszych założeń projektowych nie dokończyliśmy kodu zastępczego dla „bwareaopen", która jednak po przebadaniu opisu działania może być podobnie wsparta bisekcją.