8,12,2020

Robot Vision

RV ex8 Shape factors and unsupervised object classification

Mateusz Warmuz

1. Implement k-nn (k nearest neighbours) algorithm using MATLAB environment (see APPENDIX A). Test your program on synthetic data.

Code:

```
amountOfpoints = 10;

class1intervalk1 = 0:0.01:2;
class1intervalk2 = 0:0.01:2;
class1intervalk3 = 0:0.01:2;

class2intervalk1 = 2:0.01:4;
class2intervalk2 = 2:0.01:4;
class2intervalk3 = 2:0.01:4;

k = 3;

for i = 1:amountOfpoints
   class1positionk1(i) = randsample(class1intervalk1, 1);
   class1positionk2(i) = randsample(class1intervalk2, 1);
   class1positionk3(i) = randsample(class1intervalk3, 1);

   class2positionk1(i) = randsample(class2intervalk1, 1);
   class2positionk2(i) = randsample(class2intervalk2, 1);
   class2positionk3(i) = randsample(class2intervalk3, 1);

end

% 2 groups with random positions
c1 = [class1positionk1(:), class1positionk2(:), class1positionk3(:)];
c2 = [class2positionk1(:), class2positionk2(:), class2positionk3(:)];

pointintervalk1 = 0:0.01:4;
pointintervalk2 = 0:0.01:4;
pointintervalk3 = 0:0.01:4;
pointk1 = randsample(pointintervalk1, 1);
pointk2 = randsample(pointintervalk2, 1);
pointk3 = randsample(pointintervalk3, 1);

point = [pointk1, pointk2, pointk3];



for i = 1:size(c1,1)
   distancec1(i) = sqrt((c1(i,1)-point(1,1))^2+(c1(i,2)-point(1,2))^2+(c1(i,3)-point(1,3))^2);
   distancec2(i) = sqrt((c2(i,1)-point(1,1))^2+(c2(i,2)-point(1,2))^2+(c2(i,3)-point(1,3))^2);
end



distance = horzcat(distancec1, distancec2);
sortdistance = sort(distance);
```

```
sortinterval = sortdistance(1:k);


d = [];
for i = 1:k
   [r,c] = find(distance == sortinterval(i));
   d = [d c];
end

%countig which group satisfy the condition
count1=0;
count2=0;

for i = 1:k
   if (d(i)<=amountOfpoints)
      count1 = count1 + 1;
   end
   if (d(i)>amountOfpoints)
      count2 = count2 + 1;
   end
end
```

2. Analyse the influence of k value on classification results. Propose an algorithm to proceed in the case when the number of objects belonging to different classes is equal among the k nearest neighbours.

Code:

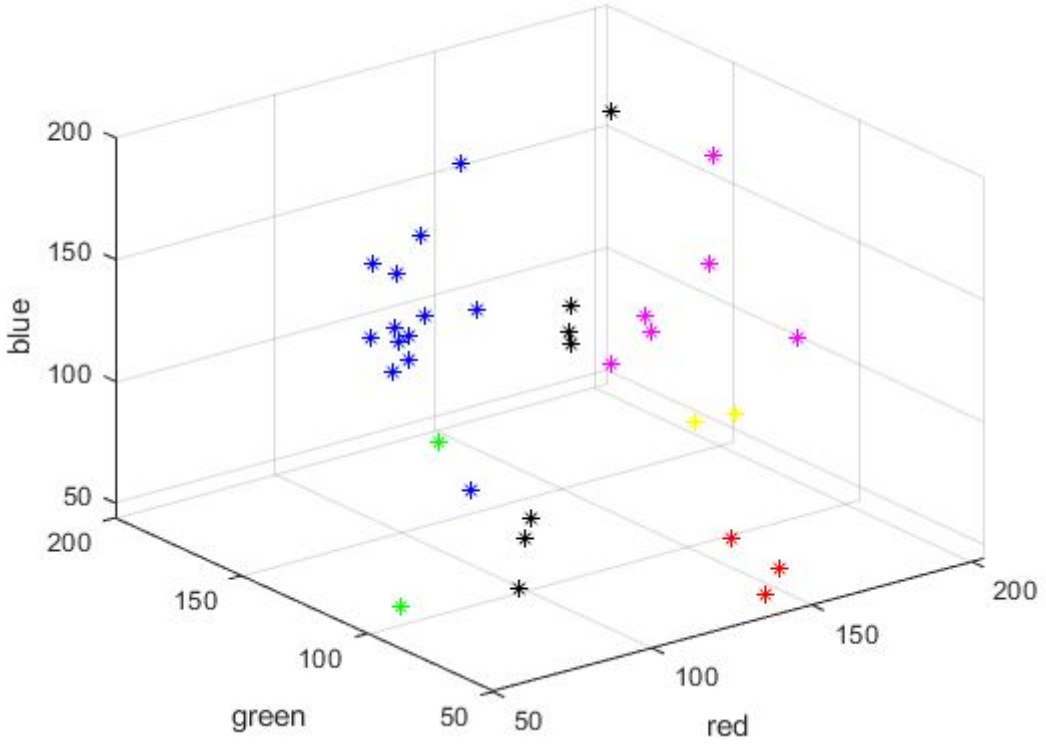
```
sum1 = 0;
sum2 = 0;

for i = 1:k
   if (count1 == count2 && d(i)<=amountOfpoints)
      sum1 = sum1 + distance(d(i));
   end
    if (count1 == count2 && d(i)>amountOfpoints)
       sum2 = sum2 + distance(d(i));
    end
end
group1 = 0;
group2 = 0;

if (sum1>sum2)
   group1 = 1;
end
if (sum2>sum1)
   group2 = 1;
end
```

If k is even and the program cannot decide where to allocate the point to the group. I propose to compare the sum of distance one group to the other. Based on this the condition checks where the sum is the smallest.

3. Test your program on real data—use images from LEGO BRICK directory. Propose a method for the classification of blocks of a specific shape, size, colour and blocks connecting these features (e.g. identification of pink blocks with dimensions of 2x2)



In my feature plane I use 3 coordinates (red, green, blue) to recognize the color of the object.

Code:

```
photo1=imread('LEGO_61.jpg');


bw = rgb2gray(photo1);
bw = 1 - imbinarize(bw);

group = cell(1,6);

% yellow = [];
%
% for i = 1:2
%     pound = bwselect(bw);
%
%     props = regionprops(pound, 'Centroid');
%     x = props.Centroid(1);
%     y = props.Centroid(2);
%
%     r = photo1(floor(y),floor(x),1);
```

```
%     g = photo1(floor(y),floor(x),2);
%     b = photo1(floor(y),floor(x),3);
%     color = [r, g, b];
%
%
%     yellow = [yellow; color];
%
% end
group{1,1} = yellow;

%
% green = [];
% for i = 1:3
%     pound = bwselect(bw);
%
%     props = regionprops(pound, 'Centroid');
%     x = props.Centroid(1);
%     y = props.Centroid(2);
%
%     r = photo1(floor(y),floor(x),1);
%     g = photo1(floor(y),floor(x),2);
%     b = photo1(floor(y),floor(x),3);
%     color = [r, g, b];
%
%
%     green = [green; color];
%
% end

group{1,2} = green;

% blue = [];
% for i = 1:15
%     pound = bwselect(bw);
%
%     props = regionprops(pound, 'Centroid');
%     x = props.Centroid(1);
%     y = props.Centroid(2);
%
%     r = photo1(floor(y),floor(x),1);
%     g = photo1(floor(y),floor(x),2);
%     b = photo1(floor(y),floor(x),3);
%     color = [r, g, b];
%
%
%     blue = [blue; color];
%
% end

group{1,3} = blue;

% gray = [];
```

```
% for i = 1:7
%     pound = bwselect(bw);
%
%     props = regionprops(pound, 'Centroid');
%     x = props.Centroid(1);
%     y = props.Centroid(2);
%
%     r = photo1(floor(y),floor(x),1);
%     g = photo1(floor(y),floor(x),2);
%     b = photo1(floor(y),floor(x),3);
%     color = [r, g, b];
%
%
%     gray = [gray; color];
%
% end

group{1,4} = gray;
%
% pink = [];
% for i = 1:6
%     pound = bwselect(bw);
%
%     props = regionprops(pound, 'Centroid');
%     x = props.Centroid(1);
%     y = props.Centroid(2);
%
%     r = photo1(floor(y),floor(x),1);
%     g = photo1(floor(y),floor(x),2);
%     b = photo1(floor(y),floor(x),3);
%     color = [r, g, b];
%
%
%     pink = [pink; color];
%
% end

group{1,5} = pink;

% red = [];
% for i = 1:3
%     pound = bwselect(bw);
%
%     props = regionprops(pound, 'Centroid');
%     x = props.Centroid(1);
%     y = props.Centroid(2);
%
%     r = photo1(floor(y),floor(x),1);
%     g = photo1(floor(y),floor(x),2);
%     b = photo1(floor(y),floor(x),3);
%     color = [r, g, b];
%
```

```
%
%     red = [red; color];
%
% end

group{1,6} = red;


% view(3);
% hold on
% grid on
% for i = 1:15
%     scatter3(group{i}(:,1), group{i}(:,2), group{i}(:,3), '*');
%     set(gcf,'color','w');
% end
% hold off
% xlabel('red');
% ylabel('green');
% zlabel('blue');


view(3);
hold on
grid on
scatter3(group{1}(:,1), group{1}(:,2), group{1}(:,3), '*y');
set(gcf,'color','w');

hold off

hold on
grid on
scatter3(group{2}(:,1), group{2}(:,2), group{2}(:,3), '*g');
set(gcf,'color','w');

hold off

hold on
grid on
scatter3(group{3}(:,1), group{3}(:,2), group{3}(:,3), '*b');
set(gcf,'color','w');

hold off

hold on
grid on
scatter3(group{4}(:,1), group{4}(:,2), group{4}(:,3), '*k');
set(gcf,'color','w');

hold off
hold on
grid on
scatter3(group{5}(:,1), group{5}(:,2), group{5}(:,3), '*m');
```

```
set(gcf,'color','w');
hold on
grid on
scatter3(group{6}(:,1), group{6}(:,2), group{6}(:,3), '*r');
set(gcf,'color','w');
hold off

hold off
xlabel('red');
ylabel('green');
zlabel('blue');
```

5. Finding a given object on images of real scenes.

My data of image recognition.

Lego_31

Without any hesitation the program recognizes that the color of the block above is blue. Due to that data the image is reached by blue blocks.

Lego_13

In this image program was around 40% sure that the color of the block is yellow with k = 5, due to my learning data of yellow colors being poor. It contains only two images of this color block. If i change the parameter k program is 67% sure that the color of the block is yellow.

Lego_42

In this case the program was 60% sure that the image above is red.

Conclusions:
To increase the quality of the color recognition program, It has to contain more data of colors/hue of lego bricks, so called learning data.

Code:

```
% loading new image
% photo1=imread('LEGO_31.jpg');
%
%
% bw = rgb2gray(photo1);
% bw = 1 - imbinarize(bw);
%
% new = [];
%
% pound = bwselect(bw);
%
% props = regionprops(pound, 'Centroid');
% x = props.Centroid(1);
% y = props.Centroid(2);
%
% r = photo1(floor(y),floor(x),1);
% g = photo1(floor(y),floor(x),2);
% b = photo1(floor(y),floor(x),3);
% color = [r, g, b];



%calculating k-nn

k = 5;

for i = 1:size(yellow,1)
   distance(i,1) =
abs(double(yellow(i,1))-double(color(1,1)))+abs(double(yellow(i,2))-double(color(1,2)))+abs(double(yel
low(i,3))-double(color(1,3)));
end

for i = 1:size(green,1)
   distance(i,2) =
abs(double(green(i,1))-double(color(1,1)))+abs(double(green(i,2))-double(color(1,2)))+abs(double(gre
en(i,3))-double(color(1,3)));
end

for i = 1:size(blue,1)
   distance(i,3) =
abs(double(blue(i,1))-double(color(1,1)))+abs(double(blue(i,2))-double(color(1,2)))+abs(double(blue(i,
3))-double(color(1,3)));
end

for i = 1:size(gray,1)
   distance(i,4) =
abs(double(gray(i,1))-double(color(1,1)))+abs(double(gray(i,2))-double(color(1,2)))+abs(double(gray(i,
3))-double(color(1,3)));
end
```

```
for i = 1:size(pink,1)
   distance(i,5) =
abs(double(pink(i,1))-double(color(1,1)))+abs(double(pink(i,2))-double(color(1,2)))+abs(double(pink(i,3
))-double(color(1,3)));
end

for i = 1:size(red,1)
   distance(i,6) =
abs(double(red(i,1))-double(color(1,1)))+abs(double(red(i,2))-double(color(1,2)))+abs(double(red(i,3))-
double(color(1,3)));
end


disctance_all = vertcat(distance(:,1), distance(:,2), distance(:,3), distance(:,4), distance(:,5),
distance(:,6));

distance_withoutzeros = nonzeros(disctance_all);

sorted = sort(distance_withoutzeros);
sortedinterval = sorted(1:k);


d = [];
for i = 1:k
   [r,c] = find(distance == sortedinterval(i));
   d = [d; c];
end
```