9.12.2020

Robot Vision

RV ex9 Case studies

Mateusz Warmuz

9.12.2020

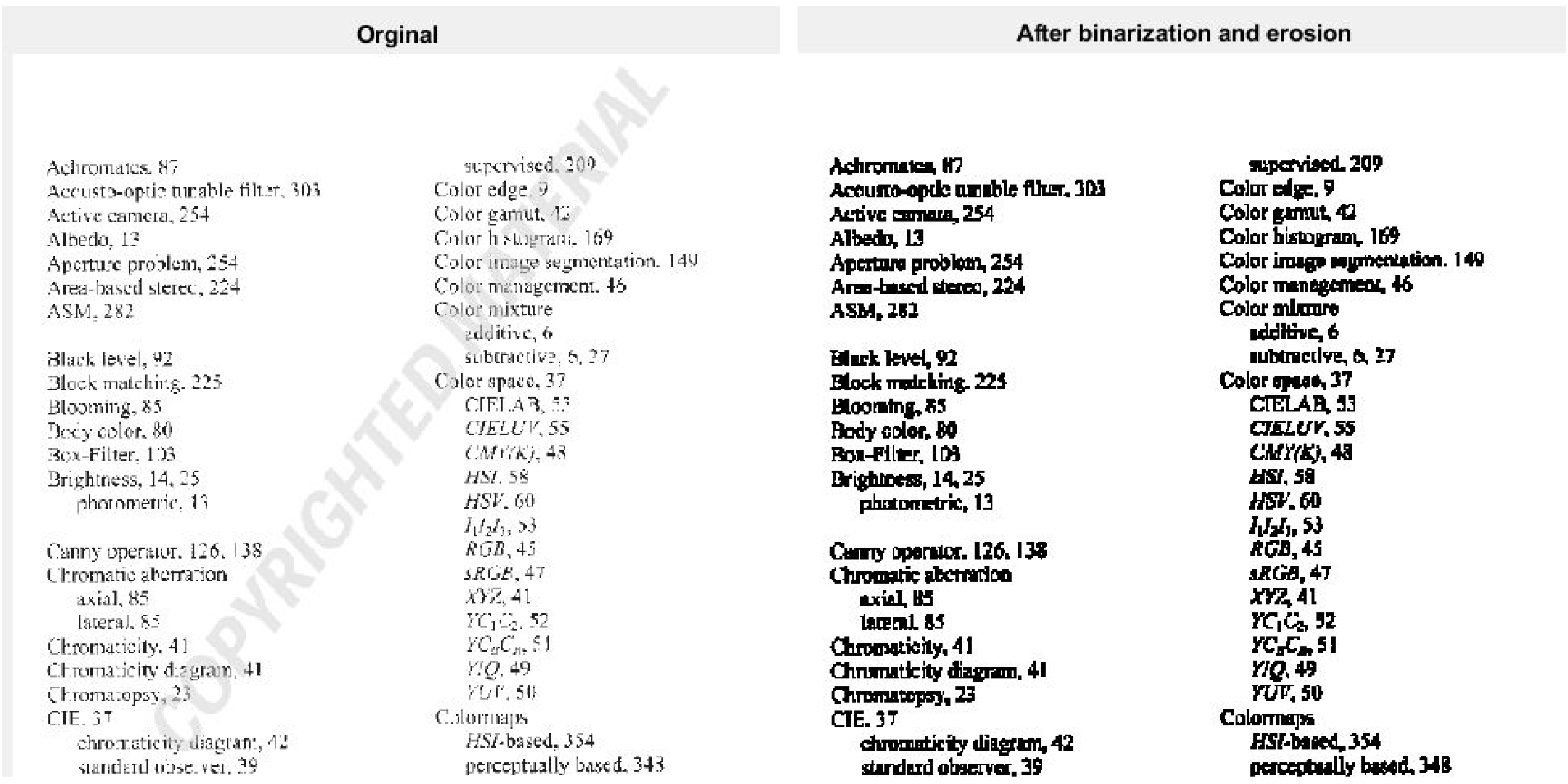
TASK 1. INSCRIPTION REMOVING

1. For the colour image TEXT.BMP propose an algorithm that will be used for removing the inscription from this image. The removing procedure should not damage the original text in the image.

Code:

```
photo=imread('text.bmp');
grey = rgb2gray(photo);

% on threshold higher than 0.9 it apper unwanted text
bw = imbinarize(grey, 0.9);


se = strel('square',2);
D = imerode(bw,se);
subplot(2,1,1)
imshow(photo)
title('Orginal');
subplot(2,1,2)
imshow(D);
title('After binarization and erosion');
```

2. Implement the proposed algorithm in MATLAB and test it.



- Achromates, 87
- Accusto-optic tunable filter, 303
- Active camera, 254
- Albedo, 13
- Aperture problem, 254
- Area-based stereo, 224
- ASM, 282

- Black level, 92
- Block matching, 225
- Blooming, 85
- Body color, 80
- Box-Filter, 103
- Brightness, 14, 25
  - photometric, 13

- Canny operator, 126, 138
- Chromatic aberration
  - axial, 85
  - lateral, 85
- Chromaticity, 41
- Chromaticity diagram, 41
- Chromatopsy, 23
- CIE, 37
  - chromaticity diagram, 42
  - standard observer, 39
  - supervised, 209
- Color edge, 9
- Color gamut, 42
- Color histogram, 169
- Color image segmentation, 149
- Color management, 46
- Color mixture
  - additive, 6
  - subtractive, 6, 27
- Color space, 37
  - CIELAB, 53
  - *CIELUV*, 55
  - *CMY(K)*, 48
  - *HSI*, 58
  - *HSV*, 60
  - $I_1I_2I_3$, 53
  - *RGB*, 45
  - *sRGB*, 47
  - *XYZ*, 41
  - YC_1C_2, 52
  - YC_bC_r, 51
  - *YIQ*, 49
  - *YUV*, 50
- Colormaps
  - *HSI*-based, 354
  - perceptually based, 343

- Achromates, 87
- Accusto-optic tunable filter, 303
- Active camera, 254
- Albedo, 13
- Aperture problem, 254
- Area-based stereo, 224
- ASM, 282

- Black level, 92
- Block matching, 225
- Blooming, 85
- Body color, 80
- Box-Filter, 103
- Brightness, 14, 25
  - photometric, 13

- Canny operator, 126, 138
- Chromatic aberration
  - axial, 85
  - lateral, 85
- Chromaticity, 41
- Chromaticity diagram, 41
- Chromatopsy, 23
- CIE, 37
  - chromaticity diagram, 42
  - standard observer, 39
  - supervised, 209
- Color edge, 9
- Color gamut, 42
- Color histogram, 169
- Color image segmentation, 149
- Color management, 46
- Color mixture
  - additive, 6
  - subtractive, 6, 27
- Color space, 37
  - CIELAB, 53
  - *CIELUV*, 55
  - *CMY(K)*, 48
  - *HSI*, 58
  - *HSV*, 60
  - $I_1I_2I_3$, 53
  - *RGB*, 45
  - *sRGB*, 47
  - *XYZ*, 41
  - YC_1C_2, 52
  - YC_bC_r, 51
  - *YIQ*, 49
  - *YUV*, 50
- Colormaps
  - *HSI*-based, 354
  - perceptually based, 348

The idea is very simple. We binarize above the original image to as high as possible threshold (in my case 0.9) by this operation we get filtered text without unwanted text. To improve quality of text i use morphological operation

TASK 4. THE INSPECTION OF CANS

1. Propose an algorithm for a vision system that is used for detection of opened cans on the production line. You can use both grey-scale and colour images.

I use the euler number of a binary image. The idea is that we have three matrices

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

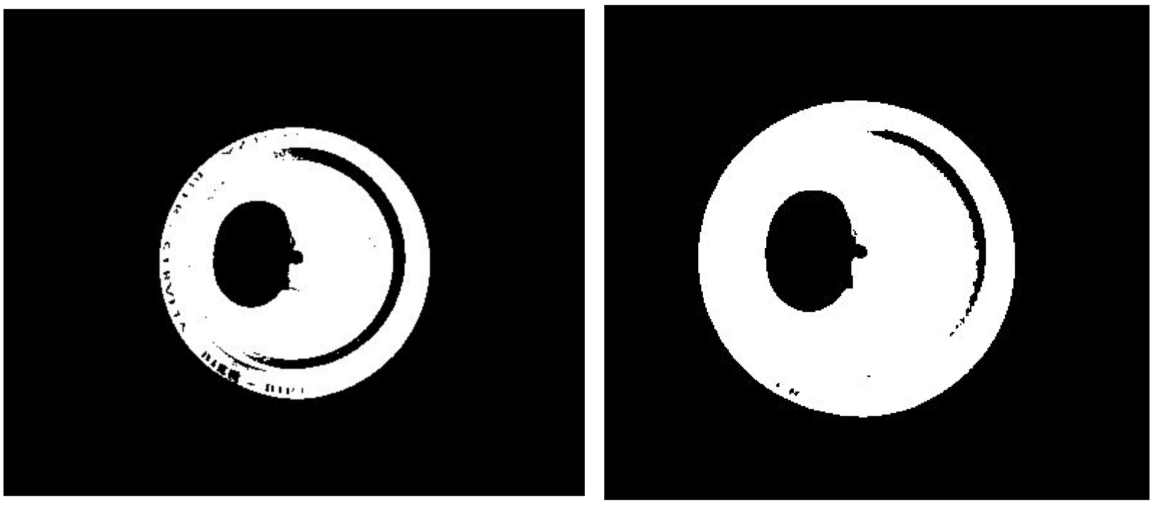
These above matrices represent the edge of the picture. Comparing the image to the matrices and calculating how many times it appears we have sufficient information to calculate euler formula. This can be used to calculate how many holes are in the image.
$euler = m1 - m2 + m3$.

2. Implement this algorithm in MATLAB and test it. The program should be as fast as it is possible.

```
can=imread('opened_12.bmp');
bw = imbinarize(can, 0.2);

count1 = 0;
count2 = 0;
count3 = 0;
m1 = [0 0; 0 1];
m2 = [0 1; 1 1];
m3 = [1 0; 0 1];

for i = 1:(size(bw,1)-1)
   for j = 1:(size(bw,2)-1)
      if (isequal(bw(i:(i+1), j:(j+1)), m1))
         count1 = count1 + 1;
      end
      if (isequal(bw(i:(i+1), j:(j+1)), m2))
         count2 = count2 + 1;
      end
      if (isequal(bw(i:(i+1), j:(j+1)), m3))
         count3 = count3 + 1;
      end
   end
end
euler =abs(count1 - count2 + count3);
```

3. Test your program for 3 images from directory CANS.

Comparing all opened cans to closed. I can deduce that for open cans the euler number is relatively small (this number also depends for image quality), but for closed cans the euler number is sufficiently higher.

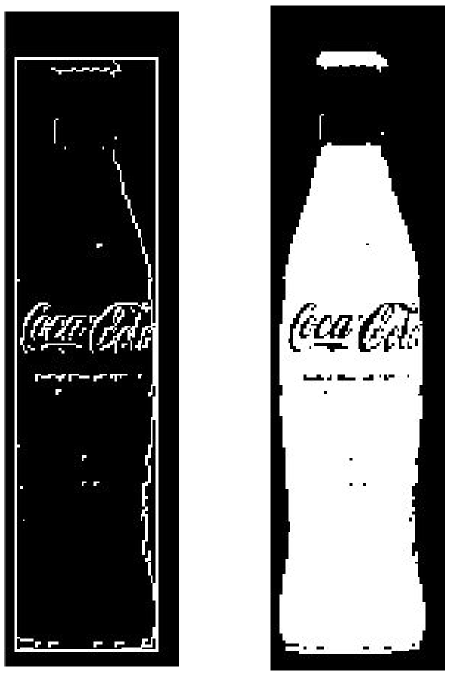


In the above images we can see two different position and quality images of can. We can notice that the shadow on second images is significantly small which causes a small euler number. Likewise we can see on the first image some noise of letters on can which also can negatively influence on euler numbers.

TASK 5. THE INSPECTION OF BOTTLES

1. Propose an algorithm for a vision system that can be used for inspection of bottles. You have to check if the bottles are full and closed—only bottles fulfilling that two conditions can pass the inspection.

I propose an algorithm which calculates the height of an object to the height of liquid in the bottle.



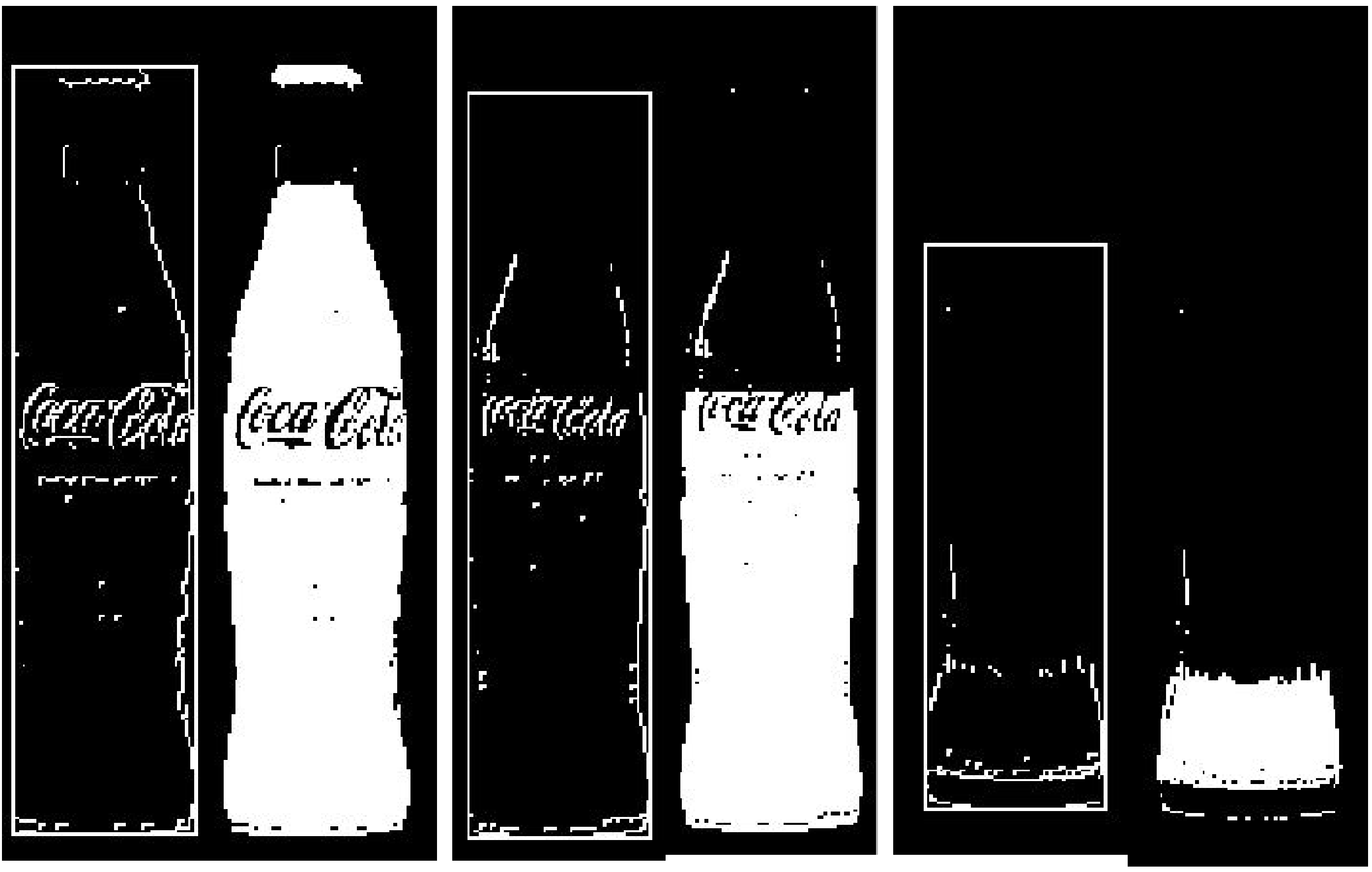
Based on the feret box I was able to get enough approximate the size of the object. Having the vertical size and area of coke I can approximately calculate the height of the liquid using formula of area computation of rectangle.

2. Implement your algorithm in MATLAB and test it. The program should be as fast as it is possible.

```
photo=imread('bottles_21.jpg');
bottle_gray = rgb2gray(photo);

bw = 1 - imbinarize(bottle_gray);

imshow(bw);

se = strel('square',2);
D = imerode(bw,se);

pound = bw;
% pound = bwselect(bw);
area = bwarea(pound);


%% Calculating size of bottle

contour = (pound-D);

x_max = 0;
x_min = size(contour,1);

y_max = 0;
y_min = size(contour,2);


for row = 1:size(contour,1)
    for col = 1:size(contour,2)
        if(contour(row,col) == 1)
            if(row > y_max)
                y_max = row;

            end
            if(row < y_min)
                y_min = row;

            end
            if(col > x_max)
                x_max = col;
            end
            if(col < x_min)
                x_min = col;
            end
        end
    end
```

```
end
line = contour;

line(y_min, x_min:x_max) = 1;
line(y_max, x_min:x_max) = 1;
line(y_min:y_max, x_min) = 1;
line(y_min:y_max, x_max) = 1;

out = [x_max-x_min, y_max-y_min];

subplot(2,2,1)
imshow(line);
subplot(2,2,2)
imshow(pound);
%% calculation

heightOfCoke = area/out(1);
procent = (heightOfCoke/out(2))*100;
```

3. Test your program for 3 images from directory BOTTLES.



For all full bottles the ratio of height was around 70%. For not enough coke, the ratio is getting smaller. Based on the above images the percent ratio respectively was 73.9%, 55%, 18,75%. We can see that this method is quite good for good quality of image. If we have some shadow or disturbances we can observe that the algorithm can mislead some part of the object, what we can observe in the last case.