Final Project: Background Subtraction on Color Image

Digital Image Processing, Autumn 2017

by Pargorn Puttapirat (3117999011)

School of Electronic and Information Engineering

Xi'an Jiaotong University

Table of Contents

Introduction

Background subtraction technique can be used to separate the changed foreground from the background in a stationary frame. The situations may include processing image with known or controllable background and security cameras.

Method

Since our input image is full color image in RGB model. To simplify the calculation, we transform RGB model to grayscale image with the following transformation matrix.

$$\begin{bmatrix} R \\ G \\ B \\ 1 \end{bmatrix} = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 \\ 1/3 & 1/3 & 1/3 & 0 \\ 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \\ 1 \end{bmatrix}$$

After the input image is transformed. R, G, or B can be used as grayscale-input image f. Sample result is shown here.

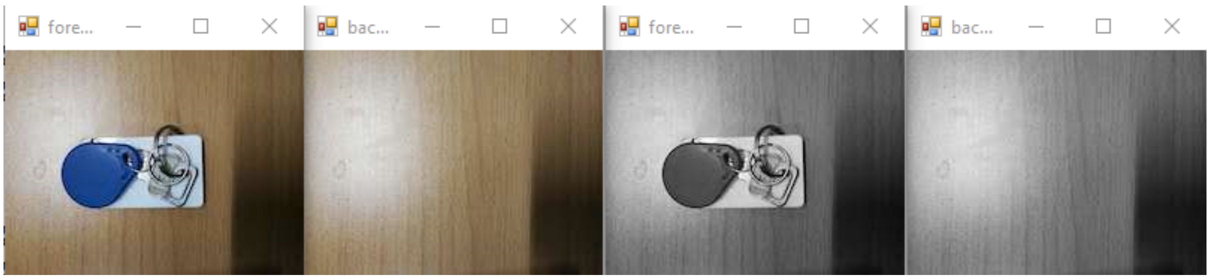


Figure 1 Original object image and background in RGB and grayscale

The concept of this technique is shown in equation 1. The object or foreground can be obtained by finding the differences between the foreground image, f, and the background image, b. If the different is more than some set threshold, thr, we can consider that the specific pixel as an object.

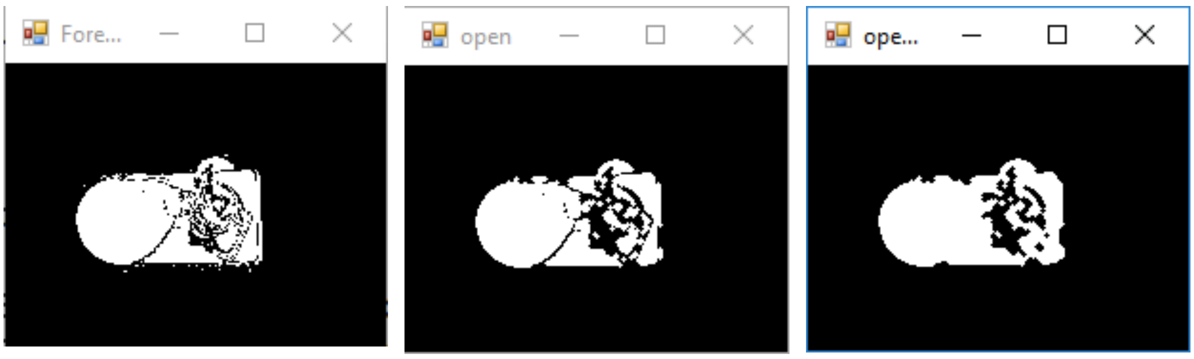
$$g(x,y) = \begin{cases} 1, & |f(x,y) - b(x,y)| > thr \\ 0, & others \end{cases} \quad (1)$$

Image $g(x,y)$ is a binary image that contains coordinates of foreground (white) and background (black). $g(x,y)$ can be used to apply to the color RGB input image to obtain only foreground. However, image g might contain some noises, therefore we need to remove the noises. In this project, morphological operation will be used. In this case, we can consider using open operation with our binary image. Open operation can be done by erosion then dilation. The expressions for erosion and dilation are as follow.

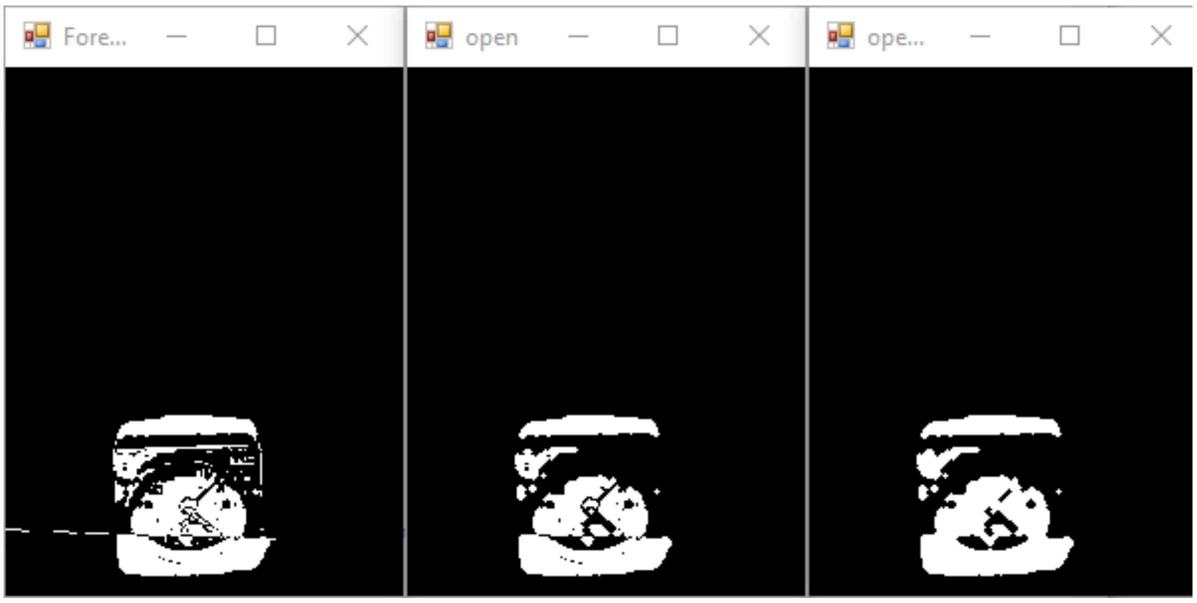
Erosion $\quad A(-)B = \{z|(B)z \subset A\}$

Dilation $\quad A(+)B = \{z|(B)z \cap A \neq \emptyset\}$

To further improve the result, we consider using close operation to connect the disconnected part of the foreground in the output image. The followings show to result of mask without morphological operation, mask with open, and mask with open then close operation. Noted that all dilation and erosion mask size is 3-by-3 pixels.



Original mask Open Open and close



Original mask Open Open and close

Figure 2 Original background subtraction mask and the masks that has been operated with open and open and close operation

The last step is the apply our foreground-background mask to the original image. To do this, we must apply the mask to each channel, R, G, and B, individually. We can do this in the program (See appendix 1.16).

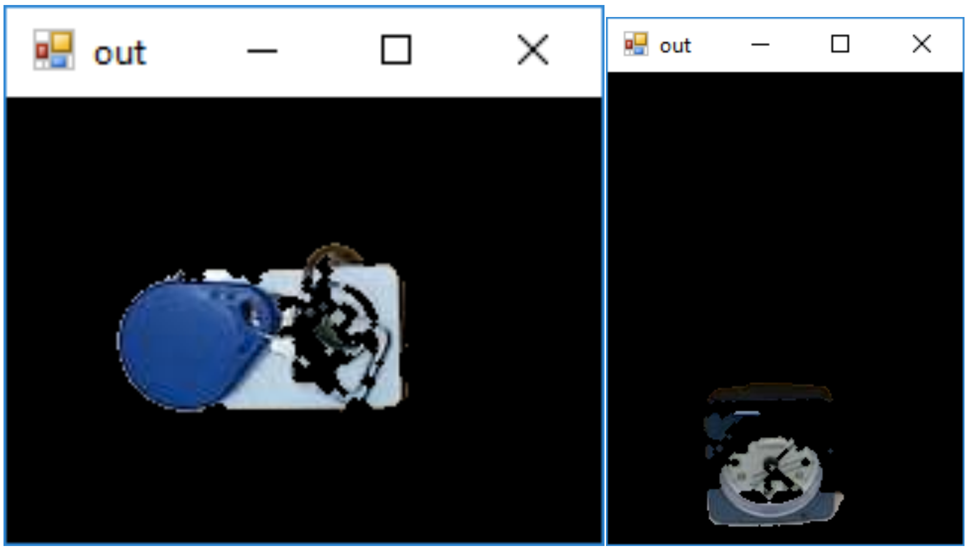
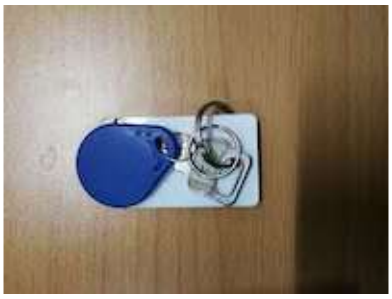
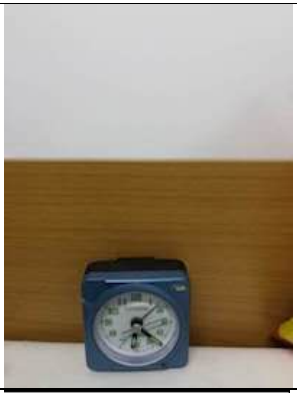
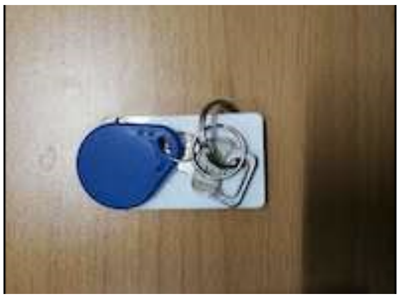
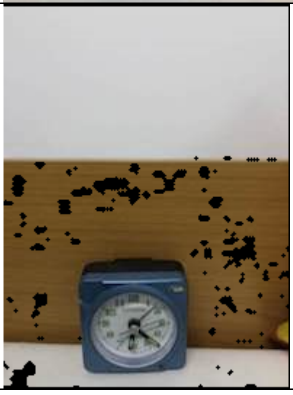


Figure 3 Sample final result of the proposed method

Experiment Results

When we use the proposed method, threshold can be set to different values. In this project, we will try different thresholds including 0, 10, 20, and 30 with two sample images sized 200x150 pixels. The results are as follow.

Threshold	Image 1	Image 2
Input Image		
0		

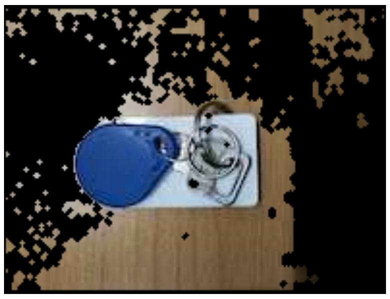
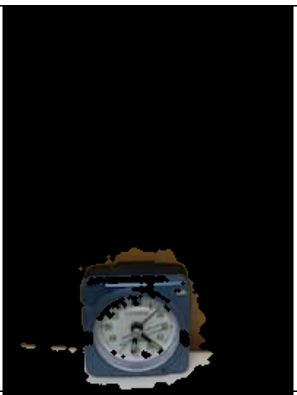
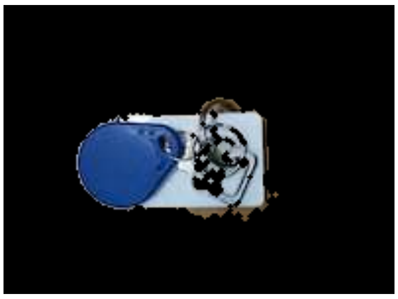
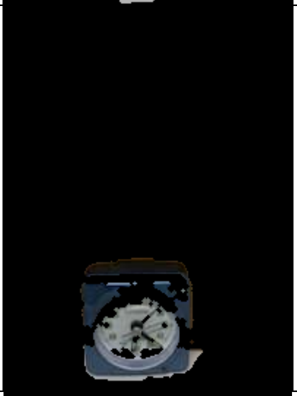
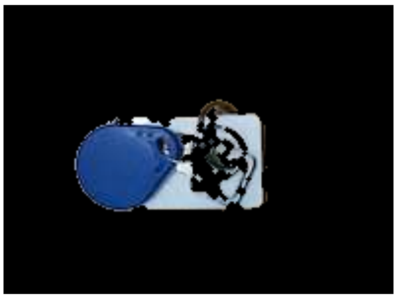
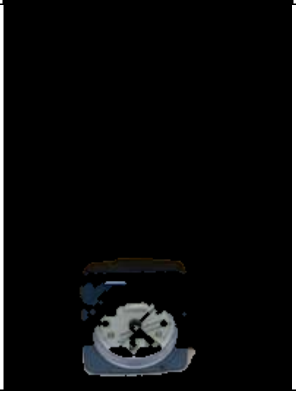
10		
20		
30		

Figure 4 Result with different threshold setting

Discussion

From the experiment results, we can see that the appropriate threshold value is different for different image. This could be affected by different lighting conditions and color of the object and the background. If the color of the object and the background is similar or the same, then this method would work poorly.

Since the subtraction process in this project is based on grayscale image. Unexpected subtraction of intensity might occur. For example, the pixel which has red (255,0,0) color and blue (0,0,255) color would have the same intensity in grayscale image ((255+0+0)/3 = 85). If this kind of situation occur, the proposed method would be ineffective to separate the foreground from the background.

Suggested Improvements

1. The mask size of dilation and erosion could be adjusted for better results and adaptive to input image size.
2. Instead of transforming RGB image to grayscale and operate the background subtraction, we can apply the operation to each color channel individually and may get a better result.

Appendix

Source Code

1.1.1. Function for erosion

```
byte[,] erosion(byte[,]f)
{
int w = f.GetLength(0);
int h = f.GetLength(1);

byte[,] g = new byte[w,h];

for (int y=1;y<h-1;y++)
  for (int x=1;x<w-1;x++)
     if (f[x,y]==255 && f[x-1,y]==255 && f[x+1,y]==255 &&
f[x,y-1]==255 && f[x,y+1]==255) g[x,y]=255;

return g;
}
```

1.1.2. Function for dilation

```
byte[,] dilation(byte[,]f)
{
int w = f.GetLength(0);
int h = f.GetLength(1);

byte[,] g = new byte[w,h];

for (int y=1;y<h-1;y++)
  for (int x=1;x<w-1;x++)
     if (f[x,y]==255 || f[x-1,y]==255 || f[x+1,y]==255 ||
f[x,y-1]==255 || f[x,y+1]==255) g[x,y]=255;

return g;
}
```

1.1.3. Function for open and close (based on erosion and dilation)

```
byte[,] open(byte[,]f)
{
return dilation(erosion(f));
}

byte[,] close(byte[,]f)
{
return erosion(dilation(f));
}
```

1.1.4. Function for transforming RGB image to grayscale image

```
byte[,] rgb2gray(ARGB[,]f)
{
int w = f.GetLength(0);
int h = f.GetLength(1);

byte[,] g = new byte[w,h];

for(int x=0; x<w; x++)
    for(int y=0; y<h; y++){
      g[x,y] = (byte)((f[x,y].R+f[x,y].G+f[x,y].B)/3);
    }

return g;
}
```

1.1.5. Function for background subtraction

```
byte[,] bgSubtract(byte[,]f1,byte[,]f2,int thr)
{
int w = f1.GetLength(0);
int h = f1.GetLength(1);

byte[,] g = new byte[w,h];

for(int x=0; x<w; x++)
    for(int y=0; y<h; y++){
    int intensity_diff = (int)Abs(f1[x,y]-f2[x,y]);
    g[x,y] = 0;
    if(intensity_diff > thr) g[x,y] = 255;
    }
return g;
}
```

1.1.6. Function for applying foreground-background mask to RGB Image

```
ARGB[,] applyFGBGToRGB(byte[,]fgbg,ARGB[,]f)
```

```
{
int w = f.GetLength(0);
int h = f.GetLength(1);

ARGB[,] g = new ARGB[w,h];

for(int x=0; x<w; x++)
    for(int y=0; y<h; y++){
      g[x,y].R = (byte)(f[x,y].R*(fgbg[x,y]/255));
      g[x,y].G = (byte)(f[x,y].G*(fgbg[x,y]/255));
      g[x,y].B = (byte)(f[x,y].B*(fgbg[x,y]/255));
    }

return g;
}
```

Name: PUTTAPIRAT PARGORN

Student ID: 3117999011

Email: pargorn@live.com