Doğukan Yiğit Polat

CS484-Homework 1

21401797

Dilation routine (see functions.hpp)

```
Mat my_dilate(Mat img, Mat se)
{
    Mat out = Mat(img.rows, img.cols, 0);
    out = out * 0;
    int H = out.rows;
    int W = out.cols;
    int Hs = se.rows;
    int Ws = se.cols;

    //assuming img is binary: 255 is 1.
    for(int i = 0; i < H-Hs; i++)
    {
        for(int j = 0; j < W-Ws; j++)
        {
            int val = 0;
            for(int x = 0; x < Hs; x++)
            {
                for(int y = 0; y < Ws; y++)
                {
                    if(se.data[x*Ws + y] && img.data[(i+x)*W + (j+y)])
                    {
                        val = 255;
                    }
                }
            }
            out.data[(i+Hs)*W + (j+Ws)] = val;
        }
    }

    return out;
}
```

Dilation is just like convolution but instead of doing elementwise multiplication, we apply LOGICAL AND operator and instead of summing the products up, we apply LOGICAL OR.

Erosion routine (see functions.hpp)

```
Mat my_erode(Mat img, Mat se)
{
    Mat out = Mat(img.rows, img.cols, 0);
    out = out * 0;
    int H = out.rows;
    int W = out.cols;
    int Hs = se.rows;
    int Ws = se.cols;

    //assuming img is binary: 255 is 1.
    for(int i = 0; i < H-Hs; i++)
    {
        for(int j = 0; j < W-Ws; j++)
        {
            int val = 255;
            for(int x = 0; x < Hs; x++)
            {
                for(int y = 0; y < Ws; y++)
                {
                    if(se.data[x*Ws + y] && !img.data[(i+x)*W + (j+y)])
                    {
                        val = 0;
                    }
                }
            }
            out.data[(i+Hs)*W + (j+Ws)] = val;
        }
    }

    return out;
}
```

Erosion is very similar to dilation we just invert all of the logical operations and swap 1 with 0.

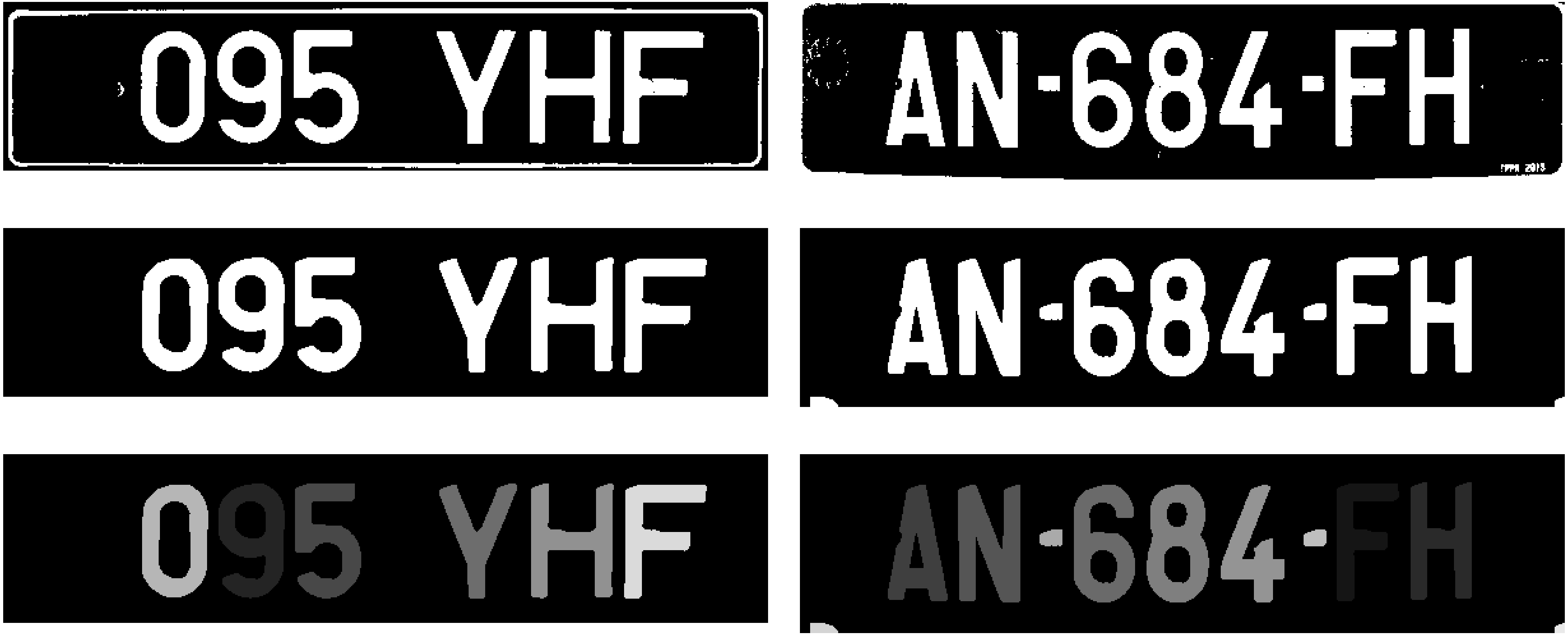
I my case 1 was 255.

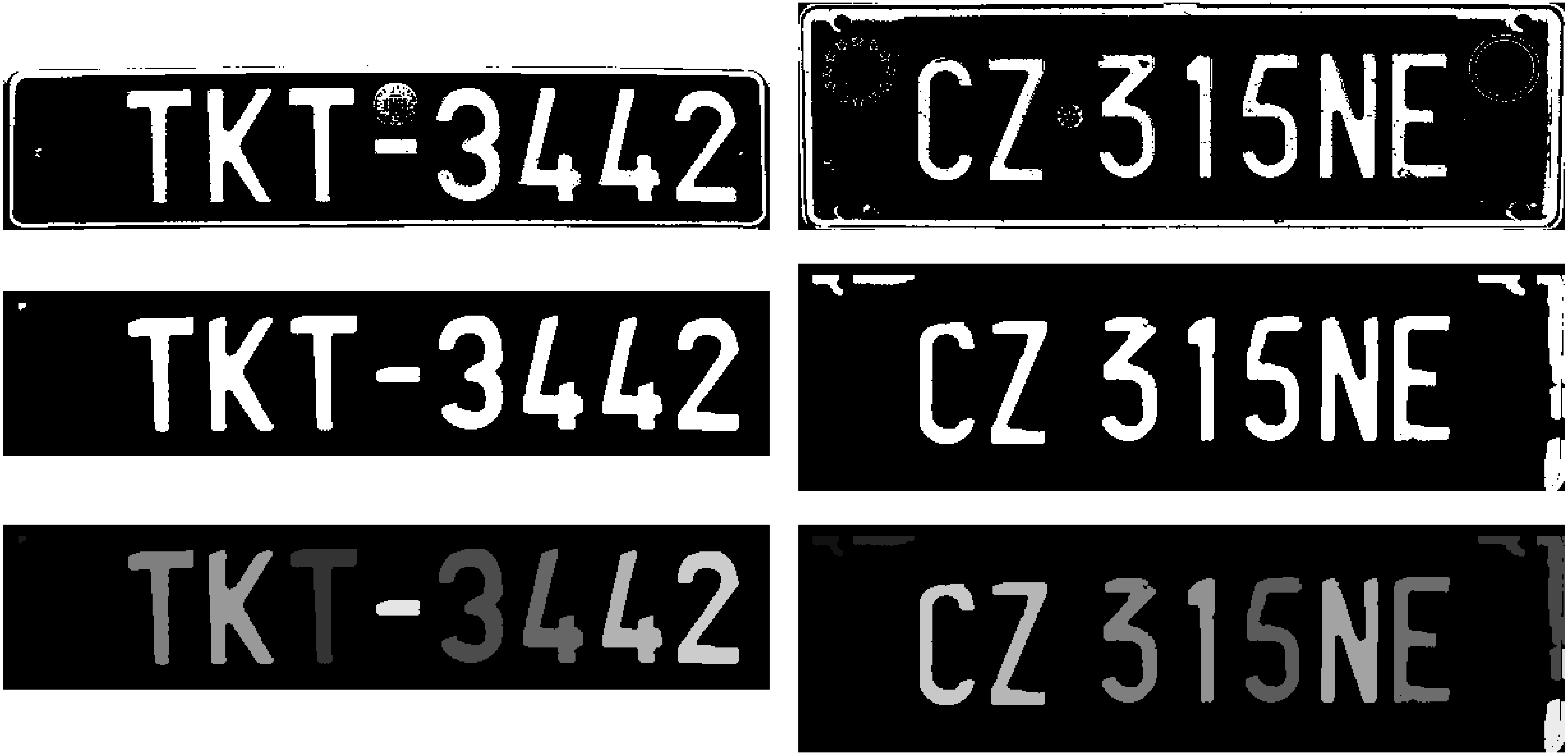
Plate Number Detection Recipe
(See plates.cpp for the full implementation)

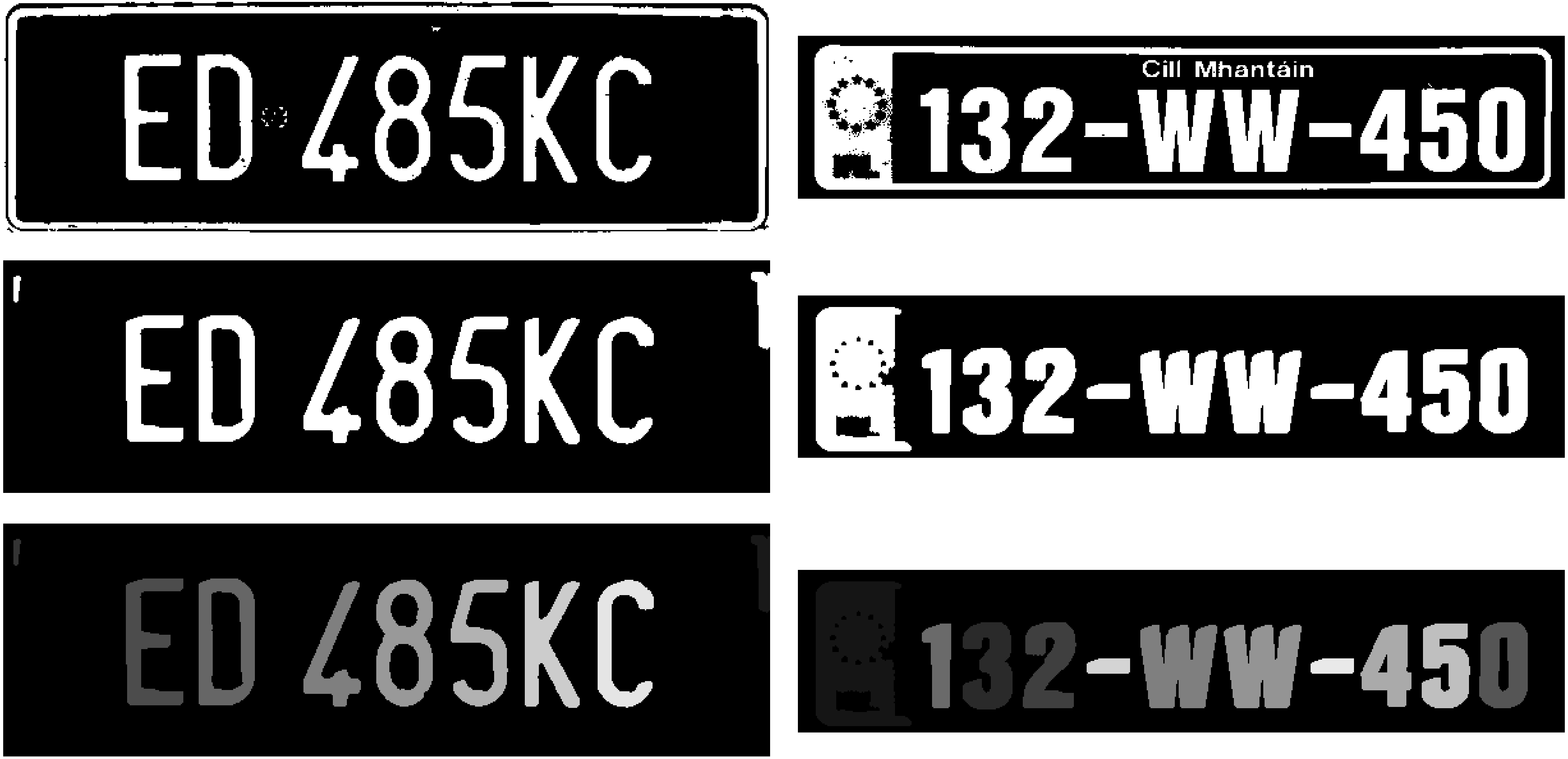
- thresholded = (gray_img < 52);
- eroded = my_diskerode(thresholded, 4);
- morph = my_diskdilate(eroded, 4);
- morph = my_diskerode(morph, 5);
- morph = my_diskdilate(morph, 5);
- morph = my_diskerode(morph, 6);
- morph = my_diskdilate(morph, 7);
- morph = my_diskerode(morph, 8);
- morph = my_diskdilate(morph, 10);
- masked = eroded & morph;
- morph = my_diskdilate(masked, 4);
- masked = thresholded & morph;
- morph = my_diskerode(masked, 3);
- masked = thresholded & morph;
- morph = my_diskdilate(masked, 5);
- masked = masked & morph;
- masked = my_diskdilate(masked, 5);
- masked = masked & thresholded;
- masked = my_diskerode(masked, 3);
- masked = my_diskdilate(masked, 10);
- masked = masked & thresholded;
- masked = my_diskdilate(masked, 10);
- masked = my_diskerode(masked, 6);
- masked = masked & thresholded;
- masked = my_diskdilate(masked, 3);
- cc = connected_components(masked);

Plate Number Detection Difficulties

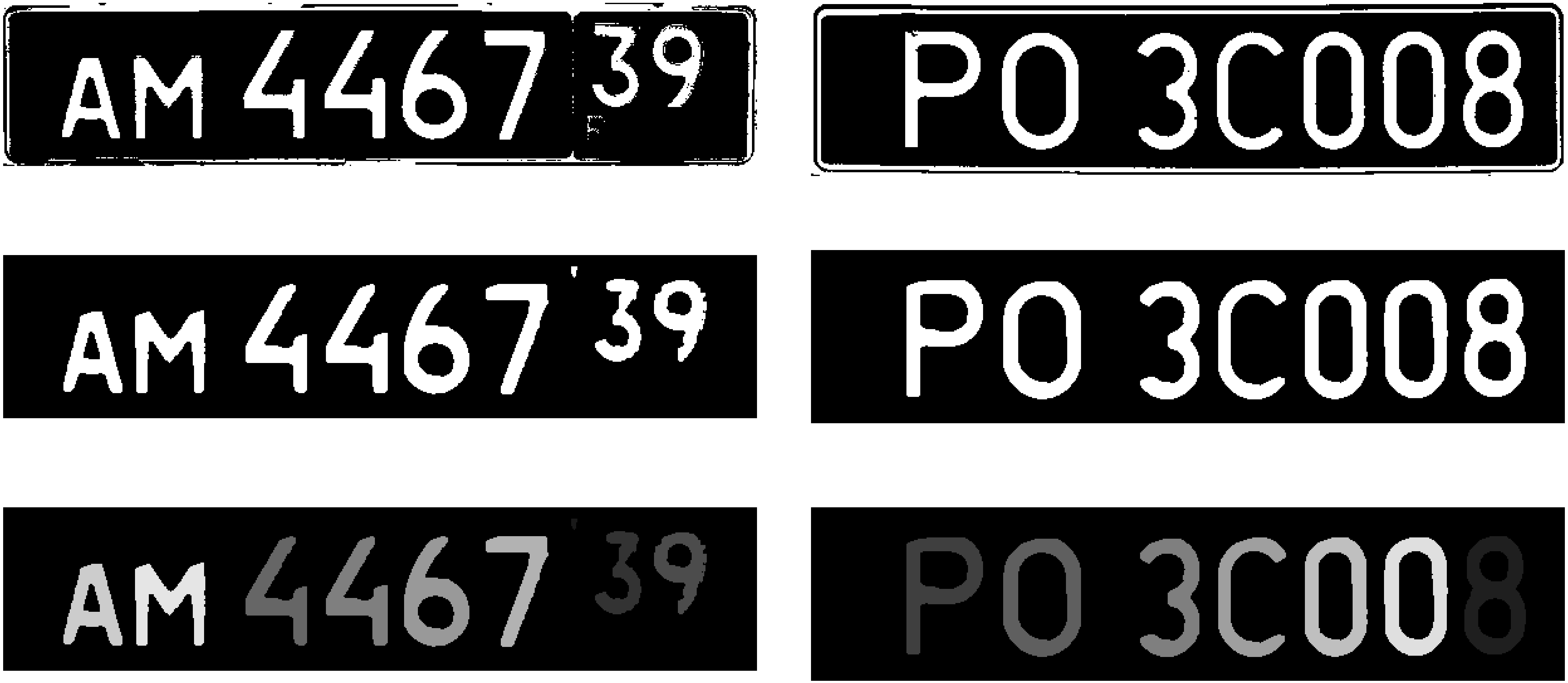
- The characteristics of the region that we want to extract is not the same for all plates.
- A good threshold for one plate destroys all of the useful information in some other plate.
- Font sizes and shapes were different so a good structuring element for one plate damaged or performed poorly on some other plate.
- We can eliminate false connected components by area thresholding. We may ignore areas smaller than -let’s say- one third of the mean component area in the image, but I have not done that.


095 YHF
AN-684-FH
095 YHF
AN-684-FH
095 YHF
AN-684-FH


TKT-3442
CZ 315NE
TKT-3442
CZ 315NE
TKT-3442
CZ 315NE


ED 485KC
Cill Mhantáin
132-WW-450
ED 485KC
132-WW-450
ED 485KC
132-WW-450


CHAW352
DN 96650
CHAW352
DN 96650
CHAW352
DN 96650



АМ 4467 39
РО 3С008
АМ 4467 39
РО 3С008
АМ 4467 39
РО 3С008


JMR 297
34 EU 2170
JMR 297
34 EU 2170
JMR 297
34 EU 2170

PETS 2000 Recipe
(See pets.cpp for the full implementation)

- x = x > 71;
- y = my_diskerode(x, 2);
- y = my_diskdilate(y, 2);
- y = my_diskdilate(y, 7);
- y = my_diskerode(y, 5);
- y = my_diskdilate(y, 5);
- y = my_diskerode(y, 6);
- y = my_diskdilate(y, 4);
- y = my_diskerode(y, 3);
- y = my_diskdilate(y, 6);
- y = my_diskerode(y, 7);
- y = my_diskclose(y, 8);
- y = my_diskdilate(y, 5);
- y = my_diskopen(y, 2);
- y = my_diskopen(y, 2);
- y = my_diskopen(y, 3);
- y = my_diskerode(y, 7);
- y = my_diskdilate(y, 9);
- y = my_diskerode(y, 7);
- y = my_diskerode(y, 3);
- y = my_diskdilate(y, 7);
- y = my_diskerode(y, 2);
- y = my_diskdilate(y, 4);
- z = connected_components(y);

PETS 2001 Recipe
(See pets.cpp for the full implementation)

- x = x > 64;
- y = my_diskerode(x, 2);
- y = my_diskdilate(y, 3);
- y = my_diskerode(y, 2);
- y = my_diskdilate(y, 3);
- y = my_diskerode(y, 3);
- y = my_diskdilate(y, 3);
- y = my_diskerode(y, 3);
- y = my_diskdilate(y, 3);
- y = my_diskerode(y, 3);
- y = my_diskdilate(y, 2);
- y = my_diskerode(y, 3);
- y = my_diskdilate(y, 4);
- y = my_diskerode(y, 2);
- y = my_diskdilate(y, 2);
- y = my_diskdilate(y, 2);
- y = my_diskerode(y, 3);
- y = my_diskdilate(y, 2);
- y = my_diskerode(y, 2);
- y = my_diskdilate(y, 3);
- z = connected_components(y);

PETS Difficulties

- Objects’ distances were tremendously different among scenes. This caused huge problems in distinguishing what was noise and what was not; because we usually eliminate noise by eroding the image and choosing a correct size for a structuring element was not feasible.
- Object types were not similar. This variety made it impossible to use object specific structuring elements that enable pattern matching.
- Intensity of noisy regions were different among scenes, this made finding a good threshold value harder.
- In PETS2000 set, all objects successfully marked with no noise.
- PETS2001 results have very poor accuracy.