Structuring-Element-Assignment

August 20, 2019

1 Implementation of Morphological Operations

We had discussed how to use dilation and erosion operations in the previous section. In this section, we will see what is going on under the hood. The most important concept that you need to understand is the Structuring element. We will discuss about the structuring element and how it is used for performing these morphological operations.

```
In [ ]: #include <iostream>
```

```
In [ ]: #include "../resource/lib/public/includeLibraries.h"
        #include "../resource/lib/public/matplotlibcpp.h"
        #include "../resource/lib/public/displayImages.h"
        #include <opencv2/opencv.hpp>
        #include <opencv2/core.hpp>
        #include <time.h>
```

```
In [ ]: using namespace std;
```

```
In [ ]: using namespace cv;
```

```
In [ ]: using namespace matplotlibcpp;
```

1.1 Implement Method 2

1. Scan through the image and superimpose the kernel on the neighborhood of each pixel.
2. Perform an AND operation of the neighborhood with the kernel.
3. Replace the pixel value with the `maximum` value in the neighborhood given by the kernel.

This means that you check every pixel and its neighborhood with respect to the kernel and change the pixel to white if any of the pixel in this neighborhood is white. OpenCV implements an optimized version of this method. This will work even if the image is not a binary image.

2 Create a Demo Image

2.1 Create an empty matrix

```
In [ ]: Mat demoImage = Mat::zeros(Size(10,10),CV_8U);
In [ ]: cout << demoImage;
```

```
In [ ]: plt::figure();
        plt::imshow(demoImage);
        auto pltImg = displayImage(demoImage);
```

```
In [ ]: pltImg
```

2.2 Lets add some white blobs

We have added the blobs at different places so that all boundary cases are covered in this example.

```
In [ ]: demoImage.at<uchar>(0,1) = 1;
```

```
In [ ]: demoImage.at<uchar>(9,0) = 1;
```

```
In [ ]: demoImage.at<uchar>(8,9) = 1;
```

```
In [ ]: demoImage.at<uchar>(2,2) = 1;
```

```
In [ ]: demoImage(Range(5,8),Range(5,8)).setTo(1);
```

```
In [ ]: cout << demoImage;
```

```
In [ ]: plt::figure();
        plt::imshow(demoImage*255);
        pltImg = displayImage(demoImage*255);
        pltImg
```

This becomes our demo Image for illustration purpose

2.3 Create an Ellipse Structuring Element

Let us create a 3x3 ellipse structuring element.

```
In [ ]: Mat element = getStructuringElement(MORPH_CROSS, Size(3,3));
```

```
In [ ]: cout << element;
```

```
In [ ]: plt::figure();
        plt::imshow(demoImage*255);
        pltImg = displayImage(demoImage*255);
        pltImg
```

```
In [ ]: int ksize = element.size().height;
```

```
In [ ]: int height, width;
        height = demoImage.size().height;
        width  = demoImage.size().width;
```

2.4 First check expected output using cv::dilate

```
In [ ]: Mat dilatedEllipseKernel;
        dilate(demoImage, dilatedEllipseKernel, element);
        cout << dilatedEllipseKernel;
```

```
In [ ]: plt::figure();
        plt::imshow(dilatedEllipseKernel*255);
        pltImg = displayImage(dilatedEllipseKernel*255);
        pltImg
```

2.5 Write Code for Dilation from scratch

Create a VideoWriter object and write the result obtained at the end of each iteration to the object. Save the video to `dilationScratch.avi` and display it using markdown below:

`dilationScratch.avi` will come here

<video width="320" height="240" controls>   <source src="dilationScratch.avi" type="video/mp4"> </video>

Note

1. Use FPS as 10
2. Before writing the frame, resize it to 50x50
3. Convert the resized frame to BGR
4. Release the object

```
In [ ]: int border = ksize/2;
        Mat paddedDemoImage = Mat::zeros(Size(height + border*2, width + border*2),CV_8UC1);
        copyMakeBorder(demoImage,paddedDemoImage,border,border,border,border,BORDER_CONSTANT,0

        Mat paddedDilatedImage = paddedDemoImage.clone();
        Mat mask;
        Mat resizedFrame;

        double minVal, maxVal;

        // Create a VideoWriter object
        ///
        /// YOUR CODE HERE
        ///

        for (int h_i = border; h_i < height + border; h_i++){
            for (int w_i = border; w_i < width + border; w_i++){
                ///
                /// YOUR CODE HERE
                ///
                // Resize output to 50x50 before writing it to the video
                ///
                /// YOUR CODE HERE
                ///
                // Convert resizedFrame to BGR before writing
                ///
                /// YOUR CODE HERE
                ///
            }
        }

        // Release the VideoWriter object
        ///
        /// YOUR CODE HERE
        ///
```

```
In [ ]: // Display final image (cropped)
        ///
        /// YOUR CODE HERE
        ///
        pltImg
```

3 Implement Erosion

3.1 First check expected output using cv::erode

```
In [ ]: Mat ErodedEllipseKernel;
        erode(demoImage, ErodedEllipseKernel, element);
        cout << ErodedEllipseKernel;
```

```
In [ ]: plt::figure();
        plt::imshow(ErodedEllipseKernel*255);
        pltImg = displayImage(ErodedEllipseKernel*255);
        pltImg
```

3.2 Write code for Erosion from scratch

Create a VideoWriter object and write the result obtained at the end of each iteration to the object. Save the video to `erosionScratch.avi` and display it using markdown below:

`erosionScratch.avi` will come here

`<video width="320" height="240" controls>   <source src="erosionScratch.avi" type="video/mp4"> </video>`

Note

1. Use FPS as 10
2. Before writing the frame, resize it to 50x50
3. Convert the resized frame to BGR
4. Release the object

```
In [ ]: border = ksize/2;
        paddedDemoImage = Mat::zeros(Size(height + border*2, width + border*2),CV_8UC1);
        copyMakeBorder(demoImage,paddedDemoImage,border,border,border,border,BORDER_CONSTANT,0
        Mat paddedErodedImage = paddedDemoImage.clone();

        // Create a VideoWriter object
        ///
        /// YOUR CODE HERE
        ///

        for (int h_i = border; h_i < height + border; h_i++){
            for (int w_i = border; w_i < width + border; w_i++){
                ///
                /// YOUR CODE HERE
                ///
                // Resize output to 50x50 before writing it to the video
                ///
                /// YOUR CODE HERE
                ///
                // Convert resizedFrame to BGR before writing
                ///
                /// YOUR CODE HERE
                ///
            }
        }

        // Release the VideoWriter object
        ///
        /// YOUR CODE HERE
        ///
```

```
In [ ]: // Display final image (cropped)
        ///
        /// YOUR CODE HERE
        ///
        pltImg
```