資工所碩一 R07922164 林群崴

Mathematical Morphology - Gray Scaled Morphology

1. Dilation

```
template<typename T, size_t M, size_t N>
Mat Dilation(Mat img, T(&kernel)[M][N], int widthLimit, int heightLimit) {
	//創建新圖準備處理
	Mat result(heightLimit, widthLimit, CV_8U, Scalar(0));

	for (int height = 0; height < heightLimit; height++) {
		for (int width = 0; width < widthLimit; width++) {

			if (img.at<uchar>(height, width) > 0) {
				uchar max = 0;

				for (int i = 0; i < M; i += 1) {
					if (kernel[i][0] + height >= 0 && kernel[i][0] + height < heightLimit
						&& kernel[i][1] + width >= 0 && kernel[i][1] + width < widthLimit)
						max = img.at<uchar>(kernel[i][0] + height, kernel[i][1] + width) > max ?
img.at<uchar>(kernel[i][0] + height, kernel[i][1] + width) : max;
				}

				for (int i = 0; i < M; i += 1) {
					if (kernel[i][0] + height >= 0 && kernel[i][0] + height < heightLimit
```

```
                    && kernel[i][1] + width >= 0 && kernel[i][1] + width < widthLimit)
                    result.at<uchar>(kernel[i][0] + height, kernel[i][1] + width) = max;
            }

        }
    }
}

    return result;
}
```

在Main function中：

```
Mat dilation = Dilation(img, kernel, widthLimit, heightLimit);
```

解釋：利用octonal 3-5-5-5-3 kernel with value = 0，偵測其最大值max，在不超出整張圖的範圍內，將kernel所遮罩的點的identity替換成max值

2. Erosion

```
template<typename T, size_t M, size_t N>
Mat Erosion(Mat img, T(&kernel)[M][N], int widthLimit, int heightLimit) {
    //創建新圖準備處理
    Mat result(heightLimit, widthLimit, CV_8U, Scalar(0));

    for (int height = 0; height < heightLimit; height++) {
        for (int width = 0; width < widthLimit; width++) {
```

```
                bool exist;

                if (img.at<uchar>(height, width) > 0) {
                    exist = true;
                    uchar min = 255;

                    for (int i = 0; i < M; i += 1) {
                        if (kernel[i][0] + height < 0 || kernel[i][0] + height >= heightLimit
                            || kernel[i][1] + width < 0 || kernel[i][1] + width >= widthLimit
                            || img.at<uchar>(kernel[i][0] + height, kernel[i][1] + width) == 0) {
                            exist = false;
                            break;
                        }
                        min = img.at<uchar>(kernel[i][0] + height, kernel[i][1] + width) < min ?
img.at<uchar>(kernel[i][0] + height, kernel[i][1] + width) : min;
                    }

                    for (int i = 0; i < M; i += 1) {
                        if (exist)
                            result.at<uchar>(kernel[i][0] + height, kernel[i][1] + width) = min;
                    }
                }
            }
        }

        return result;
}
```

在Main function中：

```
Mat erosion = Erosion(img, kernel, widthLimit, heightLimit);
```

解釋：利用octonal 3-5-5-5-3 kernel with value = 0，偵測其最小值min，若圓心所對到的位置無法容納下整個kernel，則exist為false，反之則為true，並進行接下來的像素處理，在不超出整張圖的範圍內，將kernel所遮罩的點的identity替換成min值

3. Opening

```
template<typename T, size_t M, size_t N>
Mat Opening(Mat img, T(&kernel)[M][N], int widthLimit, int heightLimit) {
     return Dilation(Erosion(img, kernel, widthLimit, heightLimit), kernel, widthLimit, heightLimit);
}
```

在 Main function 中：

```
Mat opening = Opening(img, kernel, widthLimit, heightLimit);
```

解釋：依照定義，相當於先做 Erosion 後再做 Dilation

4. Closing

```
template<typename T, size_t M, size_t N>
```

```
Mat Closing(Mat img, T(&kernel)[M][N], int widthLimit, int heightLimit) {
     return Erosion(Dilation(img, kernel, widthLimit, heightLimit), kernel, widthLimit, heightLimit);
}
```

在 Main function 中：

```
Mat closing = Closing(img, kernel, widthLimit, heightLimit);
```

解釋：依照定義，相當於先做 Dilation 後再做 Erosion