2021/11/01 陳嘉政 r10922171 HW4

1.dilation

(1) 程式碼

此為 dilation 的主要程式碼，用 for loop 去掃描整張影像，只要遇到值為 1 的，以此點推算出 kernel 矩陣的大小，然後與 kernel 疊加，存回原圖，即可達到 dilation 效果。

```
Mat dilation(Mat input, Mat kernel)
{
    Mat dilation_image = Mat::zeros(input.rows, input.cols, CV_8UC1);
    for (int i = (kernelSize - 1) / 2; i < input.rows - (kernelSize - 1) / 2; i++) {
        for (int j = (kernelSize - 1) / 2; j < input.cols - (kernelSize - 1) / 2; j++) {
            if (input.at<uint8_t>(i, j) == 1) {
                Mat dilation_temp = dilation_image(cv::Rect(j - (kernelSize - 1) / 2, i - (kernelSize - 1) / 2, kernelSize, kernelSize));
                add(dilation_temp, kernel, dilation_temp);
            }
        }
    }
    for (int i = 0; i < dilation_image.rows; i++) {
        for (int j = 0; j < dilation_image.cols; j++) {
            if (dilation_image.at<uint8_t>(i, j) != 0)
                dilation_image.at<uint8_t>(i, j) = 255;
        }
    }
    return dilation_image;
}
```

(2) 結果

2.erosion

(1)程式碼

此為 erosion 的主要程式碼，用 for loop 跑整張影像，將每個點推算出 kernel 大小存在 temp，將 temp 與 kernel 相乘且結果會等於 kernel，將結果與 kernel 比對，如果相同表示 kernel 中的每點皆存在於 temp，將範圍中的每點設為 0，中心點設為 1，因為中心點即為原點。

```
Mat erosion(Mat input, Mat kernel)
{
    Mat erosion_image = Mat::zeros(input.rows, input.cols, CV_8UC1);
    for (int i = (kernelSize - 1) / 2; i < input.rows - (kernelSize - 1) / 2; i++) {
        for (int j = (kernelSize - 1) / 2; j < input.cols - (kernelSize - 1) / 2; j++) {
            Mat erosion_temp = input(cv::Rect(j - (kernelSize - 1) / 2, i - (kernelSize - 1) / 2, kernelSize, kernelSize));
            Mat erosion_result = erosion_temp.mul(kernel);
            Mat diff = erosion_result != kernel;
            bool equal = cv::countNonZero(diff) == 0;
            if (equal) {
                for (int a = i - (kernelSize - 1) / 2; a < kernelSize; a++) {
                    for (int b = j - (kernelSize - 1) / 2; b < kernelSize; b++) {
                        erosion_image.at<uint8_t>(a, b) = 0;
                    }
                }
                erosion_image.at<uint8_t>(i, j) = 255;
            }
        }
    }
    return erosion_image;
}
```

(2)結果

3.opening

(1)程式碼

先 erosion 再 dilation。

```
/*opening*/
Mat opening_image = dilation(binary(erosion_image), kernel);
imshow("opening", opening_image);
waitKey(0);
```

(2)結果

4.closing

(1)程式碼

先 dilation 再 erosion。

```
/*closing*/
Mat closing_image = erosion(binary(dilation_image), kernel);
imshow("closing", closing_image);
waitKey(0);
```

(2)結果

5.hit-and-miss

(1)程式碼

1.先取二值圖的補圖

2.原圖被 kernel j erosion

3.補圖被 kernel k erosion

4.將兩個結果取交集得到答案

```
/*hit-and-miss*/
Mat complement = Mat::zeros(thresh_image.rows, thresh_image.cols, CV_8UC1);
for (int i = 0; i < thresh_image.rows; i++) {
    for (int j = 0; j < thresh_image.cols; j++) {
        if (thresh_image.at<uint8_t>(i, j) == 1)
            complement.at<uint8_t>(i, j) = 0;
        else
            complement.at<uint8_t>(i, j) = 1;
    }
}

Mat erosion_by_J = erosion(thresh_image, J_kernel);
Mat erosion_by_K = erosion(complement, K_kernel);
Mat hit_and_miss = Mat::zeros(thresh_image.rows, thresh_image.cols, CV_8UC1);
for (int i = 0; i < hit_and_miss.rows; i++) {
    for (int j = 0; j < hit_and_miss.cols; j++) {
        if (erosion_by_J.at<uint8_t>(i, j) && erosion_by_K.at<uint8_t>(i, j))
            hit_and_miss.at<uint8_t>(i, j) = 255;
        else
            hit_and_miss.at<uint8_t>(i, j) = 0;
    }
}
imshow("hit and miss", hit_and_miss);
waitKey(0);
```

(2)結果