Computer Vision hw_4

By R01922124 許彥彬

In this part the OpenCV-2.4.2 I/O function was included.

A class “Kernel” is used in this homework to denote a kernel, which used to do erosion and dilation. We use octagon kernel for all except “hit and miss”.

```
class Kernel{
        public:
                int cols,rows;
                int **ele;
                Kernel(int x, int y){
                        rows=x;
                        cols=y;
                        ele = new int *[rows];
                        for(int i=0;i<rows;i++){
                                ele[i]=new int[cols];
                        }
                        for(int i=0;i<rows;i++){
                                for(int j=0;j<cols;j++){
                                        ele[i][j]=0;
                                }
                        }
                }
                void set_ele(int **k){
                        for(int i=0;i<rows;i++){
                                for(int j=0;j<cols;j++){
                                        ele[i][j]=k[i][j];
                                }
                        }
                }
};
```

1. Dilation:
   i. First, use functions that wrote in previous homework to transform lena.bmp in to binary image. Search the whole binary image, if a pixel is true, do “di_ele” function to the pixel in order to do the dilation process.
   ii. Code:

```
Mat * dilation(Mat *p, Kernel *k){
        Mat *result=new Mat(p->rows,p->cols,0);
        for(int i=0;i<p->rows;i++){
                for(int j=0;j<p->cols;j++){
                        pixel_set(result,i,j,0);
                }
        }
        for(int i=0;i<p->rows;i++){
                for(int j=0;j<p->cols;j++){
                        if(pixel_get(p,i,j)==255){
                                di_ele(result,k,i,j);
                        }
                }
        }
        return result;
}

void di_ele(Mat *p, Kernel *k, int x, int y){
        int p_x=x-(k->rows/2);
        int p_y=y-(k->cols/2);
        for(int i=0;i<k->rows;i++){
                for(int j=0;j<k->cols;j++){
                        if(p_x+i>=0 && p_x+i<p->rows && p_y+j>=0 && p_y+j<p->cols && k->ele[i][j]==1){
                                pixel_set(p,p_x+i,p_y+j,255);
                        }
                }
        }
}
```

iii. Result:

2. Erosion

i. First, use functions that wrote in previous homework to transform lena.bmp in to binary image. Search the whole binary image, if a pixel is true, do “ero_ele” function to the pixel in order to do the erosion process. If those pixels around the pixel are just like the kernel, return true, else false.

ii. Code:

```
Mat * erosion(Mat *p, Kernel *k){
	Mat *result=new Mat(p->rows,p->cols,0);
	for(int i=0;i<p->rows;i++){
		for(int j=0;j<p->cols;j++){
			pixel_set(result,i,j,0);
		}
	}
	for(int i=0;i<p->rows;i++){
		for(int j=0;j<p->cols;j++){
			pixel_set(result,i,j,ero_ele(p,k,i,j));
		}
	}
	return result;
}

int ero_ele(Mat *p, Kernel *k, int x, int y){
	int p_x=x-(k->rows/2);
	int p_y=y-(k->cols/2);
	for(int i=0;i<k->rows;i++){
		for(int j=0;j<k->cols;j++){
			if(k->ele[i][j]==1){
				if(p_x+i<0 || p_x+i>=p->rows || p_y+j<0 || p_y+j>=p->cols){ //out of bound
					//continue;
					return 0;
				}
				if(pixel_get(p,p_x+i,p_y+j)!=255){
					return 0;
				}
			}
		}
	}
	return 255;
}
```

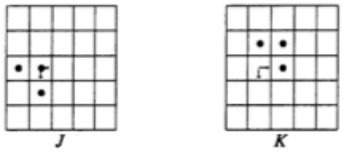
iii. Result:

3. Opening
   i. We first do the erosion to lena.bmp, and then do the dilation.
   ii. Result:

4. Closing
   i. We first do the dilation to lena.bmp, and then do the erosion.
   ii. Result:

5. Hit and miss:

   i. Kernel:



   ii. Let A be input picture lena.bmp the formula is:

$$A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K)$$

   iii. Code:

```
Mat * hit_and_miss(Mat *p){
	Kernel j(3,3);
	Kernel k(3,3);
	int j_arr[3][3]={0,0,0,1,1,0,0,1,0};
	int k_arr[3][3]={0,1,1,0,0,1,0,0,0};
	int **arr_j=new int *[3];
	int **arr_k=new int *[3];
	for(int i=0;i<3;i++){
		arr_j[i]=new int[3];
		arr_k[i]=new int[3];
	}
	for(int i=0;i<3;i++){
		for(int j=0;j<3;j++){
			arr_j[i][j]=j_arr[i][j];
			arr_k[i][j]=k_arr[i][j];
		}
	}
	j.set_ele(arr_j);
	k.set_ele(arr_k);
	Mat *A=new Mat(p->rows,p->cols,0);
	Mat *A_c=new Mat(p->rows,p->cols,0);
	Mat *result=new Mat(p->rows,p->cols,0);
	for(int i=0;i<A_c->rows;i++){
		for(int j=0;j<A_c->cols;j++){
			pixel_set(A,i,j,pixel_get(p,i,j));
			pixel_set(A_c,i,j,255-pixel_get(p,i,j));
		}
	}
	A=erosion(A,&j);
	A_c=erosion(A_c,&k);
	for(int i=0;i<p->rows;i++){
		for(int j=0;j<p->cols;j++){
			if(pixel_get(A,i,j)==255 && pixel_get(A_c,i,j)==255){
				pixel_set(result,i,j,255);
			}
			else{
				pixel_set(result,i,j,0);
			}
		}
	}
	return result;
}
```

iv. Result:

6. Appendix
   i. build_all.sh
      command: “sh build_all.sh” will automatically compile the code
   ii. R01922124_HW4.cpp
      source code
   iii. lena.bmp, di_lena.bmp, ero_lena.bmp, close_ lena.bmp, open_lena.bmp, hit_and_miss_lena.bmp
      results for this homework
   iv. R01922124_HW4.pdf