Computer Vision hw_7

By R01922124 許彥彬

1. Intro of this homework:

   This assignment is about to do “thining” to “Lena.bmp” with size 64x64. Thining includes three parts of steps, Yokoi connectivity, pair relationship and coneected shrink. I will discuss them in detail respectively. The neighbor is define as the figure shown below.

x_7	x_2	x_6
x_3	x_0	x_1
x_8	x_4	x_5

   Last homework I defined three parameters Q, R, and S as 1, 0 and 2. In this homework there are two more parameters, Q_2, P and they are 0 and 1 respectively.

```
#define Q 1
#define R 0
#define S 2

#define P 1
#define Q_2 0
```

   This work includes some functions as follows.

```
void pixel_set(Mat &, int, int ,int);
uchar pixel_get(Mat &, int ,int);
void binary_image(Mat *, int);
void resize(Mat *, int, int);
int F4(int , int , int , int );
int F4(int , int , int , int,int,int );
int H4(int,int, int, int);
void YokoiConnectivityNumber (Mat&, Mat &);
bool ConnectedShrink (const Mat& ,Mat& , Mat&  );
void PairRelationship ( const Mat&  , Mat&  );
void Thinning ( const Mat& , Mat& );
```

2. Resize

   I. Code:

```
void resize(Mat *p, int rows, int cols){
        Mat res(rows, cols, p->type());
        int row_scale = p->rows/rows;
        int col_scale = p->cols/cols;
        for(int i=0;i<rows;i++){
                for(int j=0;j<cols;j++){
                        pixel_set(res,i,j,pixel_get(*p,i*row_scale,j*col_scale));
                }
        }
        *p=res.clone();
}
```

3. Yoikoi connectivity:

I. Formula:

$$a_1 = h(x_o, x_1, x_6, x_2),\ \ a_2 = h(x_o, x_2, x_7, x_3),\ \ a_3 = h(x_o, x_3, x_8, x_4),$$

$$a_4 = h(x_o, x_4, x_5, x_1)$$

$$h(b,c,d,e) = \begin{cases} q, & if\ b = c\ and\ (d \neq b\ or\ e \neq b) \\ r, & if\ b = c\ and\ (d = b\ and\ e = b) \\ s, & if\ b \neq c \end{cases}$$

$$f(a_1, a_{2,} a_3, a_4) = \begin{cases} 5, & if\, a_1 = a_2 = a_3 = a_4 = r \\ n, & where\ n = \#\{a_k | a_k = q\}, otherwise \end{cases}$$

II. Code:

```
int H4(int b,int c, int d, int e) {
        if ( b==c && ( d!=b || e!=b ) )
                return Q;
        if ( b==c && d==b && e==b )
                return R;
        if ( b!=c )
                return S;
        return -1;
}


int F4(int a1, int a2, int a3, int a4) {
        if ( a1 == a2 && a1 == a3 && a1==a4 && a1==R ) return 5;
        int n=0;
        if ( a1 == Q ) n++;
        if ( a2 == Q ) n++;
        if ( a3 == Q ) n++;
        if ( a4 == Q ) n++;
        return n;
}

void YokoiConnectivityNumber(Mat &image , Mat &res){
        Mat t_image;
        image.copyTo(t_image);
        int mv[8][2] = { 0,1, -1,0, 0,-1, 1,0, 1,1,-1,1, -1,-1, 1,-1 };
        int x[9],a[4];
        int rows = t_image.rows;
        int cols = t_image.cols;
        int tr,tc;

        image.copyTo(res);

        for(int i=0;i<rows;i++){
        for(int j=0;j<cols;j++){
                pixel_set(res,i,j,255);
        }
         }

        for (int i=0;i<rows;i++){
        for (int j=0;j<cols;j++){
                x[0] = pixel_get(t_image,i,j)/255;
                if (x[0] == 0 ) continue;
                for (int k=0;k<8;k++){
                        tr = i+mv[k][0];
                        tc = j+mv[k][1];
                if (tr< 0||tr>=rows||tc<0||tc>=cols)
                        x[k+1] = 0;
                else
                        x[k+1] = pixel_get(t_image,tr,tc)/255;
        }
        a[0] = H4(x[0],x[1],x[6],x[2]);
        a[1] = H4(x[0],x[2],x[7],x[3]);
        a[2] = H4(x[0],x[3],x[8],x[4]);
        a[3] = H4(x[0],x[4],x[5],x[1]);
        pixel_set(res,i,j,F4(a[0],a[1],a[2],a[3]));
    }
  }
}
```

4. Pair relationship:

I. Formula:

$$h(a,1) = \begin{cases} 1, & if\ a = 1 \\ 0, & otherwise \end{cases}$$

$$y = \begin{cases} q, & if \sum_{n=1}^{4} h(x_n, 1) < 1\ or\ x_0 \neq 1 \\ p, & if \sum_{n=1}^{4} h(x_n, 1) \geq 1\ and\ x_0 \neq 1 \end{cases}$$

II. Code:

```
void PairRelationship ( const Mat& src , Mat& dest ){
	Mat t_image;
	src.copyTo(t_image);
	int mv[8][2] = { 0,1, -1,0, 0,-1, 1,0, 1,1,-1,1, -1,-1, 1,-1 };
	int x[9],a[4];
	int rows = t_image.rows;
	int cols = t_image.cols;
	int tr,tc;

	src.copyTo(dest);
	for ( int i=0 ; i<rows ; i++ ) {
		for ( int j=0 ; j<cols ; j++ ) {
			pixel_set(dest,i,j,255);
		}
	}

	for ( int i=0 ; i<rows ; i++ ) {
		for ( int j=0 ; j<cols ; j++ ) {
			x[0] = pixel_get(t_image,i,j);
			if ( x[0] == 255 )
				continue;
			if ( x[0] == 1 ) {
				int k;
				for ( k=0 ; k<4 ; k++ ) {
					tr = i+mv[k][0];
					tc = j+mv[k][1];
					if ( tr>= 0 && tr<rows && tc>=0 && tc<cols && pixel_get(t_image,tr,tc) == 1)
				break;
				}
				if ( k<4 ){
					pixel_set(dest,i,j,P);
					continue;
				}
			}
			pixel_set(dest,i,j,Q_2);
		}
	}
}
```

Note that there is already a Q in Yokoi connectivity, so I define q in here as Q_2 instead.

5. Connected Shrink

I. Formula:

$$h(b,c,d,e) = \begin{cases} 1, & if\ b = c\ and\ (d \neq b\ or\ e \neq b) \\ 0, & otherwise \end{cases}$$

$$f(a_1, a_{2,} a_3, a_4, x) = \begin{cases} g, & if\ exactly\ one\ of\ (a_1, a_{2,} a_3, a_4) = 1 \\ x, & otherwise \end{cases}$$

Note that the h function is very similar to Yokoi connectivity. So I will use the same function and do a little modify. If the return number from

H4 is 1 assign 1, else assign 0.

II. Code:

```
int F4(int a1, int a2, int a3, int a4, int g, int x){
	if(a1+a2+a3+a4==1)
		return g;
	else
		return x;
}

bool ConnectedShrink (const Mat& src ,Mat& data, Mat& dest){
	Mat t_image;
	src.copyTo(t_image);
	bool flag = false;
	int mv[8][2] = { 0,1, -1,0, 0,-1, 1,0, 1,1,-1,1, -1,-1, 1,-1 };
	int x[9],a[4];
	int rows = t_image.rows;
	int cols = t_image.cols;
	int tr,tc;

	for ( int i=0 ; i<rows ; i++ ) {
		for ( int j=0 ; j<cols ; j++ ) {
			if ( pixel_get(data,i,j) != P )
				continue;
			x[0]=255;
			for ( int k=0 ; k<8 ; k++ ) {
			tr = i+mv[k][0];
			tc = j+mv[k][1];
			if ( tr< 0 || tr>=rows || tc<0 || tc>=cols )
				x[k+1] = 0;
			else
				x[k+1] = pixel_get(t_image,tr,tc);
			}
			a[0] = H4(x[0],x[1],x[6],x[2])==1?1:0;
			a[1] = H4(x[0],x[2],x[7],x[3])==1?1:0;
			a[2] = H4(x[0],x[3],x[8],x[4])==1?1:0;
			a[3] = H4(x[0],x[4],x[5],x[1])==1?1:0;
			int t = F4(a[0],a[1],a[2],a[3],0,255);
			if ( t == 0 )
				flag = true;
			 pixel_set(t_image,i,j,t);

		}
	}
	t_image.copyTo(dest);
	return flag;
}
```

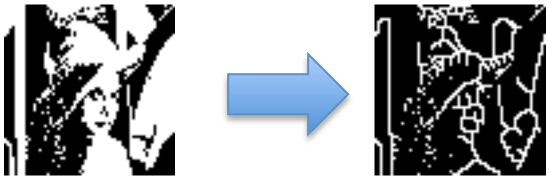
6. Thinning Operator

I. Idea: Involve the previous three operators.

II. Code:

```
void Thinning(const Mat& src , Mat& dest){
	Mat data;
	Mat t_image = src.clone();
	bool flag = true;
	binary_image(&t_image,128);
	while ( flag ){
		YokoiConnectivityNumber(t_image,data);
		PairRelationship(data,data);
		flag = ConnectedShrink(t_image,data,dest);
		t_image = dest.clone();
	}
}
```

7. Result:



8. Appendix

   I. build_all.sh

      “sh build_all.sh” will automatically compile the code in terminal.

   II. R01922124_HW7.cpp

      source code

   III. lena.bmp

      original lena image

   IV. binary_lena.bmp, thinning_lena.bmp

      results for this homework

   V. R01922124_HW7.pdf

      report