1.Algorithm & principal code fragment:

#include <iostream>

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <cv.h>

using namespace std;

using namespace cv;

Mat picInitialSrc(Mat pic, Mat src, int row, int col)

{

int i,j;

for(i=0; i<col; i++)

{

for(j=0; j<row; j++)

{

pic.at<unsigned char>(i,j) = src.at<unsigned char>(i,j);

}

}

return pic;

}

int cal(int a[4]) // q=1, r=2, s=3

{

if(a[0] != a[1])

return 3;

else if(a[2] == a[0] && a[3] == a[0])

return 2;

else

return 1;

}

int\* hFunction(Mat yokoi64, int i, int j)

{

int \*f4 = new int[4];//return 4 neborhood

int a[4] = {0};//temp

for(int k=0; k<4; k++)

{

a[k] = 0;

}

a[0] = yokoi64.at<unsigned char>(i,j);

if(j<=62)

a[1] = yokoi64.at<unsigned char>(i,j+1);

if(i>=1 && j<=62)

a[2] = yokoi64.at<unsigned char>(i-1,j+1);

if(i>=1)

a[3] = yokoi64.at<unsigned char>(i-1,j);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[0] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(i>=1)

a[1] = yokoi64.at<unsigned char>(i-1,j);

if(i>=1 && j>=1)

a[2] = yokoi64.at<unsigned char>(i-1,j-1);

if(j>=1)

a[3] = yokoi64.at<unsigned char>(i,j-1);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[1] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(j>=1)

a[1] = yokoi64.at<unsigned char>(i,j-1);

if(i<=62 && j>=1)

a[2] = yokoi64.at<unsigned char>(i+1,j-1);

if(i<=62)

a[3] = yokoi64.at<unsigned char>(i+1,j);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[2] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(i<=62)

a[1] = yokoi64.at<unsigned char>(i+1,j);

if(i<=62 && j<=62)

a[2] = yokoi64.at<unsigned char>(i+1,j+1);

if(j<=62)

a[3] = yokoi64.at<unsigned char>(i,j+1);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[3] = cal(a);

return f4;

}

void showSavePicture(Mat picture,string windowName,string saveName)

{

namedWindow( windowName, WINDOW\_AUTOSIZE );

imshow( windowName, picture );

imwrite( saveName , picture );

}

int main(int argc,char\*\* argv )

{

Mat src;

src = imread("lena.bmp",CV\_LOAD\_IMAGE\_GRAYSCALE);

int i,j,k;

int row = src.rows;

int col = src.cols;

int\* h4num;

int interior[64][64] = {0};

int mark[64][64] = {0};

int dele[64] = {0};

int ans[64][64] = {0};

//Mat symbolic(512,512,CV\_8U);

//symbolic = picInitialSrc(symbolic, src, 512, 512);

Mat symbolic(64, 64, CV\_8U);

for(i=0; i<512; i++)//calaulate symbolic matrix

{

for(j=0; j<512; j++)

{

if(i%8==0 && j%8==0)

{

if(src.at<unsigned char>(i,j) >= 128)

symbolic.at<unsigned char>(i/8,j/8) = 255;

else

symbolic.at<unsigned char>(i/8,j/8) = 0;

}

}

}

for(int y=0; y<64; y++)

{

for(int x=0; x<64; x++)

{

cout << symbolic.at<unsigned char>(y,x);

}

cout << endl;

}

int a=0;

while(a<10)

{

for(i=0; i<64; i++)//interior 1:white 2:wh#include <iostream>

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <cv.h>

using namespace std;

using namespace cv;

Mat picInitialSrc(Mat pic, Mat src, int row, int col)

{

int i,j;

for(i=0; i<col; i++)

{

for(j=0; j<row; j++)

{

pic.at<unsigned char>(i,j) = src.at<unsigned char>(i,j);

}

}

return pic;

}

int cal(int a[4]) // q=1, r=2, s=3

{

if(a[0] != a[1])

return 3;

else if(a[2] == a[0] && a[3] == a[0])

return 2;

else

return 1;

}

int\* hFunction(Mat yokoi64, int i, int j)

{

int \*f4 = new int[4];//return 4 neborhood

int a[4] = {0};//temp

for(int k=0; k<4; k++)

{

a[k] = 0;

}

a[0] = yokoi64.at<unsigned char>(i,j);

if(j<=62)

a[1] = yokoi64.at<unsigned char>(i,j+1);

if(i>=1 && j<=62)

a[2] = yokoi64.at<unsigned char>(i-1,j+1);

if(i>=1)

a[3] = yokoi64.at<unsigned char>(i-1,j);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[0] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(i>=1)

a[1] = yokoi64.at<unsigned char>(i-1,j);

if(i>=1 && j>=1)

a[2] = yokoi64.at<unsigned char>(i-1,j-1);

if(j>=1)

a[3] = yokoi64.at<unsigned char>(i,j-1);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[1] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(j>=1)

a[1] = yokoi64.at<unsigned char>(i,j-1);

if(i<=62 && j>=1)

a[2] = yokoi64.at<unsigned char>(i+1,j-1);

if(i<=62)

a[3] = yokoi64.at<unsigned char>(i+1,j);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[2] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(i<=62)

a[1] = yokoi64.at<unsigned char>(i+1,j);

if(i<=62 && j<=62)

a[2] = yokoi64.at<unsigned char>(i+1,j+1);

if(j<=62)

a[3] = yokoi64.at<unsigned char>(i,j+1);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[3] = cal(a);

return f4;

}

void showSavePicture(Mat picture,string windowName,string saveName)

{

namedWindow( windowName, WINDOW\_AUTOSIZE );

imshow( windowName, picture );

imwrite( saveName , picture );

}

int main(int argc,char\*\* argv )

{

Mat src;

src = imread("lena.bmp",CV\_LOAD\_IMAGE\_GRAYSCALE);

int i,j,k;

int row = src.rows;

int col = src.cols;

int\* h4num;

int interior[64][64] = {0};

int mark[64][64] = {0};

int dele[64] = {0};

int ans[64][64] = {0};

//Mat symbolic(512,512,CV\_8U);

//symbolic = picInitialSrc(symbolic, src, 512, 512);

Mat symbolic(64, 64, CV\_8U);

for(i=0; i<512; i++)//calaulate symbolic matrix

{

for(j=0; j<512; j++)

{

if(i%8==0 && j%8==0)

{

if(src.at<unsigned char>(i,j) >= 128)

symbolic.at<unsigned char>(i/8,j/8) = 255;

else

symbolic.at<unsigned char>(i/8,j/8) = 0;

}

}

}

for(int y=0; y<64; y++)

{

for(int x=0; x<64; x++)

{

cout << symbolic.at<unsigned char>(y,x);

}

cout << endl;

}

int a=0;

while(a<10)

{

for(i=0; i<64; i++)//interior 1:white 2:white && 8connected

{

for(j=0; j<64; j++)

{

if(symbolic.at<unsigned char>(i,j) == 255)

{

if(symbolic.at<unsigned char>(i+1, j) == 255 && symbolic.at<unsigned char>(i, j+1) == 255 && symbolic.at<unsigned char>(i-1, j) == 255 && symbolic.at<unsigned char>(i, j-1) ==255)

{

interior[i][j] = 2;

}

else

{

interior[i][j] = 1;

}

}

}

}

for(i=0; i<64; i++)//mark 1:mark point

{

for(j=0; j<64; j++)

{

if(interior[i][j] == 2)

{

//mark[i-1][j-1] = 1;

mark[i-1][j] = 1;

//mark[i-1][j+1] = 1;

mark[i][j-1] = 1;

mark[i][j] = 1;

mark[i][j+1] = 1;

//mark[i+1][j-1] = 1;

mark[i+1][j] = 1;

//mark[i+1][j+1] = 1;

}

}

}

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

if(interior[i][j] == 2)

{

mark[i][j] = 2;

}

cout << mark[i][j];

}

cout << endl;

}

cout << "======================" << endl;

/\*int timeForQ = 0;

int timeForR = 0;

if(symbolic.at<unsigned char>(i,j) == 255)

{

h4num = hFunction(symbolic,i,j);

for(int i = 0 ; i< 4; i++)

cout << "h4num:" << h4num[i] << endl;

for(k=0; k<4; k++)

{

if(h4num[k] == 1)

{

timeForQ++;

}

else if(h4num[k] == 2)

timeForR++;

}

}\*/

int timeForQ = 0;

int timeForR = 0;

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

h4num = hFunction(symbolic,i,j);

for(k=0; k<4; k++)

{

if(h4num[k] == 1)

timeForQ++;

else if(h4num[k] == 2)

timeForR++;

h4num[k] = 0;

}

if(timeForQ == 1) // cal deletable

{

dele[j] = 1;

}

if(timeForR == 4)

{

ans[i][j] = 5;

}

else

ans[i][j] = timeForQ;

timeForQ = 0;

timeForR = 0;

}

/\*for(k=0; k<64; k++)

{

cout << dele[k];

}\*/

//cout << endl;

for(k=0; k<64; k++)

{

if(dele[k] == 1)

{

if(mark[i][k] == 1)//8-connected

{

symbolic.at<unsigned char>(i,k) = 0;

}

}

}

for(int w=0; w<64; w++)

dele[w] = 0;

}

a++;

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

interior[i][j] = 0;

mark[i][j] = 0;

dele[j] = 0;

}

}

}

//dila = picInitialSrc(dila,src,row,col);

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

if(symbolic.at<unsigned char>(i,j) != 0 )

{

cout << ans[i][j];

}

else

cout << " ";

}

cout << endl;

}

showSavePicture(symbolic,"symbolic","symbolic.bmp");

waitKey(0);

return 0;

}ite && 8connected

{

for(j=0; j<64; j++)

{

if(symbolic.at<unsigned char>(i,j) == 255)

{

if(symbolic.at<unsigned char>(i#include <iostream>

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <cv.h>

using namespace std;

using namespace cv;

Mat picInitialSrc(Mat pic, Mat src, int row, int col)

{

int i,j;

for(i=0; i<col; i++)

{

for(j=0; j<row; j++)

{

pic.at<unsigned char>(i,j) = src.at<unsigned char>(i,j);

}

}

return pic;

}

int cal(int a[4]) // q=1, r=2, s=3

{

if(a[0] != a[1])

return 3;

else if(a[2] == a[0] && a[3] == a[0])

return 2;

else

return 1;

}

int\* hFunction(Mat yokoi64, int i, int j)

{

int \*f4 = new int[4];//return 4 neborhood

int a[4] = {0};//temp

for(int k=0; k<4; k++)

{

a[k] = 0;

}

a[0] = yokoi64.at<unsigned char>(i,j);

if(j<=62)

a[1] = yokoi64.at<unsigned char>(i,j+1);

if(i>=1 && j<=62)

a[2] = yokoi64.at<unsigned char>(i-1,j+1);

if(i>=1)

a[3] = yokoi64.at<unsigned char>(i-1,j);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[0] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(i>=1)

a[1] = yokoi64.at<unsigned char>(i-1,j);

if(i>=1 && j>=1)

a[2] = yokoi64.at<unsigned char>(i-1,j-1);

if(j>=1)

a[3] = yokoi64.at<unsigned char>(i,j-1);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[1] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(j>=1)

a[1] = yokoi64.at<unsigned char>(i,j-1);

if(i<=62 && j>=1)

a[2] = yokoi64.at<unsigned char>(i+1,j-1);

if(i<=62)

a[3] = yokoi64.at<unsigned char>(i+1,j);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[2] = cal(a);

for(int k=1; k<4; k++)

{

a[k] = 0;

}

if(i<=62)

a[1] = yokoi64.at<unsigned char>(i+1,j);

if(i<=62 && j<=62)

a[2] = yokoi64.at<unsigned char>(i+1,j+1);

if(j<=62)

a[3] = yokoi64.at<unsigned char>(i,j+1);

//cout << "0:" << a[0] << "1:" << a[1] << "2:" << a[2] << "3:" << a[3] << endl;

f4[3] = cal(a);

return f4;

}

void showSavePicture(Mat picture,string windowName,string saveName)

{

namedWindow( windowName, WINDOW\_AUTOSIZE );

imshow( windowName, picture );

imwrite( saveName , picture );

}

int main(int argc,char\*\* argv )

{

Mat src;

src = imread("lena.bmp",CV\_LOAD\_IMAGE\_GRAYSCALE);

int i,j,k;

int row = src.rows;

int col = src.cols;

int\* h4num;

int interior[64][64] = {0};

int mark[64][64] = {0};

int dele[64] = {0};

int ans[64][64] = {0};

//Mat symbolic(512,512,CV\_8U);

//symbolic = picInitialSrc(symbolic, src, 512, 512);

Mat symbolic(64, 64, CV\_8U);

for(i=0; i<512; i++)//calaulate symbolic matrix

{

for(j=0; j<512; j++)

{

if(i%8==0 && j%8==0)

{

if(src.at<unsigned char>(i,j) >= 128)

symbolic.at<unsigned char>(i/8,j/8) = 255;

else

symbolic.at<unsigned char>(i/8,j/8) = 0;

}

}

}

for(int y=0; y<64; y++)

{

for(int x=0; x<64; x++)

{

cout << symbolic.at<unsigned char>(y,x);

}

cout << endl;

}

int a=0;

while(a<10)

{

for(i=0; i<64; i++)//interior 1:white 2:white && 8connected

{

for(j=0; j<64; j++)

{

if(symbolic.at<unsigned char>(i,j) == 255)

{

if(symbolic.at<unsigned char>(i+1, j) == 255 && symbolic.at<unsigned char>(i, j+1) == 255 && symbolic.at<unsigned char>(i-1, j) == 255 && symbolic.at<unsigned char>(i, j-1) ==255)

{

interior[i][j] = 2;

}

else

{

interior[i][j] = 1;

}

}

}

}

for(i=0; i<64; i++)//mark 1:mark point

{

for(j=0; j<64; j++)

{

if(interior[i][j] == 2)

{

//mark[i-1][j-1] = 1;

mark[i-1][j] = 1;

//mark[i-1][j+1] = 1;

mark[i][j-1] = 1;

mark[i][j] = 1;

mark[i][j+1] = 1;

//mark[i+1][j-1] = 1;

mark[i+1][j] = 1;

//mark[i+1][j+1] = 1;

}

}

}

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

if(interior[i][j] == 2)

{

mark[i][j] = 2;

}

cout << mark[i][j];

}

cout << endl;

}

cout << "======================" << endl;

/\*int timeForQ = 0;

int timeForR = 0;

if(symbolic.at<unsigned char>(i,j) == 255)

{

h4num = hFunction(symbolic,i,j);

for(int i = 0 ; i< 4; i++)

cout << "h4num:" << h4num[i] << endl;

for(k=0; k<4; k++)

{

if(h4num[k] == 1)

{

timeForQ++;

}

else if(h4num[k] == 2)

timeForR++;

}

}\*/

int timeForQ = 0;

int timeForR = 0;

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

h4num = hFunction(symbolic,i,j);

for(k=0; k<4; k++)

{

if(h4num[k] == 1)

timeForQ++;

else if(h4num[k] == 2)

timeForR++;

h4num[k] = 0;

}

if(timeForQ == 1) // cal deletable

{

dele[j] = 1;

}

if(timeForR == 4)

{

ans[i][j] = 5;

}

else

ans[i][j] = timeForQ;

timeForQ = 0;

timeForR = 0;

}

/\*for(k=0; k<64; k++)

{

cout << dele[k];

}\*/

//cout << endl;

for(k=0; k<64; k++)

{

if(dele[k] == 1)

{

if(mark[i][k] == 1)//8-connected

{

symbolic.at<unsigned char>(i,k) = 0;

}

}

}

for(int w=0; w<64; w++)

dele[w] = 0;

}

a++;

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

interior[i][j] = 0;

mark[i][j] = 0;

dele[j] = 0;

}

}

}

//dila = picInitialSrc(dila,src,row,col);

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

if(symbolic.at<unsigned char>(i,j) != 0 )

{

cout << ans[i][j];

}

else

cout << " ";

}

cout << endl;

}

showSavePicture(symbolic,"symbolic","symbolic.bmp");

waitKey(0);

return 0;

}+1, j) == 255 && symbolic.at<unsigned char>(i, j+1) == 255 && symbolic.at<unsigned char>(i-1, j) == 255 && symbolic.at<unsigned char>(i, j-1) ==255)

{

interior[i][j] = 2;

}

else

{

interior[i][j] = 1;

}

}

}

}

for(i=0; i<64; i++)//mark 1:mark point

{

for(j=0; j<64; j++)

{

if(interior[i][j] == 2)

{

//mark[i-1][j-1] = 1;

mark[i-1][j] = 1;

//mark[i-1][j+1] = 1;

mark[i][j-1] = 1;

mark[i][j] = 1;

mark[i][j+1] = 1;

//mark[i+1][j-1] = 1;

mark[i+1][j] = 1;

//mark[i+1][j+1] = 1;

}

}

}

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

if(interior[i][j] == 2)

{

mark[i][j] = 2;

}

cout << mark[i][j];

}

cout << endl;

}

cout << "======================" << endl;

/\*int timeForQ = 0;

int timeForR = 0;

if(symbolic.at<unsigned char>(i,j) == 255)

{

h4num = hFunction(symbolic,i,j);

for(int i = 0 ; i< 4; i++)

cout << "h4num:" << h4num[i] << endl;

for(k=0; k<4; k++)

{

if(h4num[k] == 1)

{

timeForQ++;

}

else if(h4num[k] == 2)

timeForR++;

}

}\*/

int timeForQ = 0;

int timeForR = 0;

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

h4num = hFunction(symbolic,i,j);

for(k=0; k<4; k++)

{

if(h4num[k] == 1)

timeForQ++;

else if(h4num[k] == 2)

timeForR++;

h4num[k] = 0;

}

if(timeForQ == 1) // cal deletable

{

dele[j] = 1;

}

if(timeForR == 4)

{

ans[i][j] = 5;

}

else

ans[i][j] = timeForQ;

timeForQ = 0;

timeForR = 0;

}

/\*for(k=0; k<64; k++)

{

cout << dele[k];

}\*/

//cout << endl;

for(k=0; k<64; k++)

{

if(dele[k] == 1)

{

if(mark[i][k] == 1)//8-connected

{

symbolic.at<unsigned char>(i,k) = 0;

}

}

}

for(int w=0; w<64; w++)

dele[w] = 0;

}

a++;

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

interior[i][j] = 0;

mark[i][j] = 0;

dele[j] = 0;

}

}

}

//dila = picInitialSrc(dila,src,row,col);

for(i=0; i<64; i++)

{

for(j=0; j<64; j++)

{

if(symbolic.at<unsigned char>(i,j) != 0 )

{

cout << ans[i][j];

}

else

cout << " ";

}

cout << endl;

}

showSavePicture(symbolic,"symbolic","symbolic.bmp");

waitKey(0);

return 0;

}

2.your parameters

Mat src;

src = imread("lena.bmp",CV\_LOAD\_IMAGE\_GRAYSCALE);

int i,j,k;

int row = src.rows;

int col = src.cols;

int\* h4num;

int interior[64][64] = {0};

int mark[64][64] = {0};

int dele[64] = {0};

int ans[64][64] = {0};

3.resulting images

