NAME : B.PRIYADHARSHINI

ROLL.NO :18BIT139

DEPARTMENT : BSC.IT –“B”

DATE : 12.08.2020

E MAIL : [priyabala2617@gmail.com](mailto:priyabala2617@gmail.com)

QUESTION :

To convert an image to binary image and perform distance transform.

CODING :

#include <[opencv2/core.hpp](https://docs.opencv.org/master/d0/d9c/core_2include_2opencv2_2core_8hpp.html)>

#include <[opencv2/imgproc.hpp](https://docs.opencv.org/master/d1/d4f/imgproc_2include_2opencv2_2imgproc_8hpp.html)>

#include <[opencv2/highgui.hpp](https://docs.opencv.org/master/d4/dd5/highgui_8hpp.html)>

#include <iostream>

using namespace [std](https://docs.opencv.org/master/d8/dcc/namespacestd.html);

using namespace [cv](https://docs.opencv.org/master/d2/d75/namespacecv.html);

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

{

// Load the image

[CommandLineParser](https://docs.opencv.org/master/d0/d2e/classcv_1_1CommandLineParser.html) parser( argc, argv, "{@input | cards.png | input image}" );

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) src = [imread](https://docs.opencv.org/master/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)( [samples::findFile](https://docs.opencv.org/master/d6/dba/group__core__utils__samples.html#ga3a33b00033b46c698ff6340d95569c13)( parser.get<[String](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga1f6634802eeadfd7245bc75cf3e216c2)>( "@input" ) ) );

if( src.[empty](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#abbec3525a852e77998aba034813fded4)() )

{

cout << "Could not open or find the image!\n" << endl;

cout << "Usage: " << argv[0] << " <Input image>" << endl;

return -1;

}

// Show source image

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Source Image", src);

// Change the background from white to black, since that will help later to extract

// better results during the use of Distance Transform

for ( int i = 0; i < src.[rows](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#abed816466c45234254d25bc59c31245e); i++ ) {

for ( int j = 0; j < src.[cols](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#aa3e5a47585c9ef6a0842556739155e3e); j++ ) {

if ( src.[at](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#aa5d20fc86d41d59e4d71ae93daee9726)<[Vec3b](https://docs.opencv.org/master/d6/dcf/classcv_1_1Vec.html)>(i, j) == [Vec3b](https://docs.opencv.org/master/d6/dcf/classcv_1_1Vec.html)(255,255,255) )

{

src.[at](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#aa5d20fc86d41d59e4d71ae93daee9726)<[Vec3b](https://docs.opencv.org/master/d6/dcf/classcv_1_1Vec.html)>(i, j)[0] = 0;

src.[at](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#aa5d20fc86d41d59e4d71ae93daee9726)<[Vec3b](https://docs.opencv.org/master/d6/dcf/classcv_1_1Vec.html)>(i, j)[1] = 0;

src.[at](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#aa5d20fc86d41d59e4d71ae93daee9726)<[Vec3b](https://docs.opencv.org/master/d6/dcf/classcv_1_1Vec.html)>(i, j)[2] = 0;

}

}

}

// Show output image

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Black Background Image", src);

// Create a kernel that we will use to sharpen our image

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) kernel = ([Mat\_<float>](https://docs.opencv.org/master/df/dfc/classcv_1_1Mat__.html)(3,3) <<

1, 1, 1,

1, -8, 1,

1, 1, 1); // an approximation of second derivative, a quite strong kernel

// do the laplacian filtering as it is

// well, we need to convert everything in something more deeper then CV\_8U

// because the kernel has some negative values,

// and we can expect in general to have a Laplacian image with negative values

// BUT a 8bits unsigned int (the one we are working with) can contain values from 0 to 255

// so the possible negative number will be truncated

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) imgLaplacian;

[filter2D](https://docs.opencv.org/master/d5/df1/group__imgproc__hal__functions.html#ga42c2468ab3a1238fbf48458c57169081)(src, imgLaplacian, [CV\_32F](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga4a3def5d72b74bed31f5f8ab7676099c), kernel);

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) sharp;

src.[convertTo](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#adf88c60c5b4980e05bb556080916978b)(sharp, [CV\_32F](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga4a3def5d72b74bed31f5f8ab7676099c));

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) imgResult = sharp - imgLaplacian;

// convert back to 8bits gray scale

imgResult.[convertTo](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#adf88c60c5b4980e05bb556080916978b)(imgResult, [CV\_8UC3](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga88c4cd9de76f678f33928ef1e3f96047));

imgLaplacian.convertTo(imgLaplacian, [CV\_8UC3](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga88c4cd9de76f678f33928ef1e3f96047));

// imshow( "Laplace Filtered Image", imgLaplacian );

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)( "New Sharped Image", imgResult );

// Create binary image from source image

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) bw;

[cvtColor](https://docs.opencv.org/master/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab)(imgResult, bw, [COLOR\_BGR2GRAY](https://docs.opencv.org/master/d8/d01/group__imgproc__color__conversions.html#gga4e0972be5de079fed4e3a10e24ef5ef0a353a4b8db9040165db4dacb5bcefb6ea));

[threshold](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57)(bw, bw, 40, 255, [THRESH\_BINARY](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59) | [THRESH\_OTSU](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a95251923e8e22f368ffa86ba8bce87ff));

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Binary Image", bw);

// Perform the distance transform algorithm

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) dist;

[distanceTransform](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ga8a0b7fdfcb7a13dde018988ba3a43042)(bw, dist, [DIST\_L2](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa2bfbebbc5c320526897996aafa1d8ebaff0d1f5be0fc152a56a9b9716d158b96), 3);

// Normalize the distance image for range = {0.0, 1.0}

// so we can visualize and threshold it

[normalize](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga1b6a396a456c8b6c6e4afd8591560d80)(dist, dist, 0, 1.0, [NORM\_MINMAX](https://docs.opencv.org/master/d2/de8/group__core__array.html#ggad12cefbcb5291cf958a85b4b67b6149fa9f0c1c342a18114d47b516a88e29822e));

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Distance Transform Image", dist);

// Threshold to obtain the peaks

// This will be the markers for the foreground objects

[threshold](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57)(dist, dist, 0.4, 1.0, [THRESH\_BINARY](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59));

// Dilate a bit the dist image

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) kernel1 = Mat::ones(3, 3, [CV\_8U](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga32b18d904ee2b1731a9416a8eef67d06));

[dilate](https://docs.opencv.org/master/d4/d86/group__imgproc__filter.html#ga4ff0f3318642c4f469d0e11f242f3b6c)(dist, dist, kernel1);

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Peaks", dist);

// Create the CV\_8U version of the distance image

// It is needed for findContours()

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) dist\_8u;

dist.[convertTo](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html" \l "adf88c60c5b4980e05bb556080916978b)(dist\_8u, [CV\_8U](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga32b18d904ee2b1731a9416a8eef67d06));

// Find total markers

vector<vector<Point> > contours;

[findContours](https://docs.opencv.org/master/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f260e0)(dist\_8u, contours, [RETR\_EXTERNAL](https://docs.opencv.org/master/d3/dc0/group__imgproc__shape.html#gga819779b9857cc2f8601e6526a3a5bc71aa7adc6d6608609fd84650f71b954b981), [CHAIN\_APPROX\_SIMPLE](https://docs.opencv.org/master/d3/dc0/group__imgproc__shape.html#gga4303f45752694956374734a03c54d5ffa5f2883048e654999209f88ba04c302f5));

// Create the marker image for the watershed algorithm

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) markers = Mat::zeros(dist.[size](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#a146f8e8dda07d1365a575ab83d9828d1)(), [CV\_32S](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga4067910fc388075c3ea3aa14393e83b9));

// Draw the foreground markers

for (size\_t i = 0; i < contours.size(); i++)

{

[drawContours](https://docs.opencv.org/master/d6/d6e/group__imgproc__draw.html#ga746c0625f1781f1ffc9056259103edbc)(markers, contours, static\_cast<int>(i), [Scalar](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga599fe92e910c027be274233eccad7beb)(static\_cast<int>(i)+1), -1);

}

// Draw the background marker

[circle](https://docs.opencv.org/master/d9/db7/group__datasets__gr.html#gga610754124ced68d1f05760b5948fbb76a6f0d8b2d9e3e947b2a5c1eff9e81ee95)(markers, [Point](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga1e83eafb2d26b3c93f09e8338bcab192)(5,5), 3, [Scalar](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga599fe92e910c027be274233eccad7beb)(255), -1);

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Markers", markers\*10000);

// Perform the watershed algorithm

[watershed](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ga3267243e4d3f95165d55a618c65ac6e1)(imgResult, markers);

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) mark;

markers.[convertTo](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#adf88c60c5b4980e05bb556080916978b)(mark, [CV\_8U](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga32b18d904ee2b1731a9416a8eef67d06));

[bitwise\_not](https://docs.opencv.org/master/d2/de8/group__core__array.html#ga0002cf8b418479f4cb49a75442baee2f)(mark, mark);

// imshow("Markers\_v2", mark); // uncomment this if you want to see how the mark

// image looks like at that point

// Generate random colors

vector<Vec3b> colors;

for (size\_t i = 0; i < contours.size(); i++)

{

int b = [theRNG](https://docs.opencv.org/master/d2/de8/group__core__array.html#ga75843061d150ad6564b5447e38e57722)().[uniform](https://docs.opencv.org/master/d1/dd6/classcv_1_1RNG.html#acde197860cea91e5aa896be8719457ae)(0, 256);

int g = [theRNG](https://docs.opencv.org/master/d2/de8/group__core__array.html#ga75843061d150ad6564b5447e38e57722)().[uniform](https://docs.opencv.org/master/d1/dd6/classcv_1_1RNG.html#acde197860cea91e5aa896be8719457ae)(0, 256);

int r = [theRNG](https://docs.opencv.org/master/d2/de8/group__core__array.html#ga75843061d150ad6564b5447e38e57722)().[uniform](https://docs.opencv.org/master/d1/dd6/classcv_1_1RNG.html#acde197860cea91e5aa896be8719457ae)(0, 256);

colors.push\_back([Vec3b](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga7e6060c0b8d48459964df6e1eb524c03)(([uchar](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html" \l "ga65f85814a8290f9797005d3b28e7e5fc))b, ([uchar](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html" \l "ga65f85814a8290f9797005d3b28e7e5fc))g, ([uchar](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html" \l "ga65f85814a8290f9797005d3b28e7e5fc))r));

}

// Create the result image

[Mat](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html) dst = Mat::zeros(markers.[size](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#a146f8e8dda07d1365a575ab83d9828d1)(), [CV\_8UC3](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga88c4cd9de76f678f33928ef1e3f96047));

// Fill labeled objects with random colors

for (int i = 0; i < markers.[rows](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#abed816466c45234254d25bc59c31245e); i++)

{

for (int j = 0; j < markers.[cols](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#aa3e5a47585c9ef6a0842556739155e3e); j++)

{

int [index](https://docs.opencv.org/master/d9/db7/group__datasets__gr.html#gga82775e152f8a74c5fe06f5a7343e0233a9dfc90ef6dc3ba62850d76cc3534572c) = markers.[at](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#aa5d20fc86d41d59e4d71ae93daee9726)<int>(i,j);

if (index > 0 && index <= static\_cast<int>(contours.size()))

{

dst.[at](https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#aa5d20fc86d41d59e4d71ae93daee9726)<[Vec3b](https://docs.opencv.org/master/d6/dcf/classcv_1_1Vec.html)>(i,j) = colors[index-1];

}

}

}

// Visualize the final image

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Final Result", dst);

[waitKey](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga5628525ad33f52eab17feebcfba38bd7)();

return 0;

}

## Explanation / Result

* Load the source image and check if it is loaded without any problem, then show it:

// Load the image

CommandLineParser parser( argc, argv, "{@input | cards.png | input image}" );

Mat src = [imread](https://docs.opencv.org/master/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)( [samples::findFile](https://docs.opencv.org/master/d6/dba/group__core__utils__samples.html#ga3a33b00033b46c698ff6340d95569c13)( parser.get<[String](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga1f6634802eeadfd7245bc75cf3e216c2)>( "@input" ) ) );

if( src.empty() )

{

cout << "Could not open or find the image!\n" << endl;

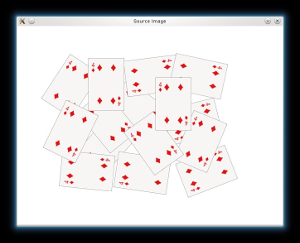
cout << "Usage: " << argv[0] << " <Input image>" << endl;

return -1;

}

// Show source image

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Source Image", src);



* Then if we have an image with a white background, it is good to transform it to black. This will help us to discriminate the foreground objects easier when we will apply the Distance Transform:

// Change the background from white to black, since that will help later to extract

// better results during the use of Distance Transform

for ( int i = 0; i < src.rows; i++ ) {

for ( int j = 0; j < src.cols; j++ ) {

if ( src.at<[Vec3b](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga7e6060c0b8d48459964df6e1eb524c03)>(i, j) == [Vec3b](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga7e6060c0b8d48459964df6e1eb524c03)(255,255,255) )

{

src.at<[Vec3b](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga7e6060c0b8d48459964df6e1eb524c03)>(i, j)[0] = 0;

src.at<[Vec3b](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga7e6060c0b8d48459964df6e1eb524c03)>(i, j)[1] = 0;

src.at<[Vec3b](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga7e6060c0b8d48459964df6e1eb524c03)>(i, j)[2] = 0;

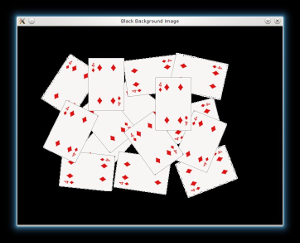
}

}

}

// Show output image

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Black Background Image", src);



* Afterwards we will sharpen our image in order to acute the edges of the foreground objects. We will apply a laplacian filter with a quite strong filter (an approximation of second derivative):

// Create a kernel that we will use to sharpen our image

Mat kernel = (Mat\_<float>(3,3) <<

1, 1, 1,

1, -8, 1,

1, 1, 1); // an approximation of second derivative, a quite strong kernel

// do the laplacian filtering as it is

// well, we need to convert everything in something more deeper then CV\_8U

// because the kernel has some negative values,

// and we can expect in general to have a Laplacian image with negative values

// BUT a 8bits unsigned int (the one we are working with) can contain values from 0 to 255

// so the possible negative number will be truncated

Mat imgLaplacian;

[filter2D](https://docs.opencv.org/master/d5/df1/group__imgproc__hal__functions.html#ga42c2468ab3a1238fbf48458c57169081)(src, imgLaplacian, [CV\_32F](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga4a3def5d72b74bed31f5f8ab7676099c), kernel);

Mat sharp;

src.convertTo(sharp, [CV\_32F](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga4a3def5d72b74bed31f5f8ab7676099c));

Mat imgResult = sharp - imgLaplacian;

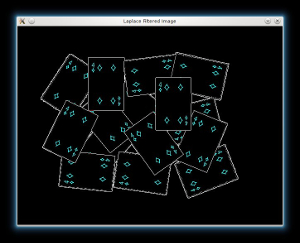
// convert back to 8bits gray scale

imgResult.convertTo(imgResult, [CV\_8UC3](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga88c4cd9de76f678f33928ef1e3f96047));

imgLaplacian.convertTo(imgLaplacian, [CV\_8UC3](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga88c4cd9de76f678f33928ef1e3f96047));

// imshow( "Laplace Filtered Image", imgLaplacian );

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)( "New Sharped Image", imgResult );



* Now we transform our new sharpened source image to a grayscale and a binary one, respectively:

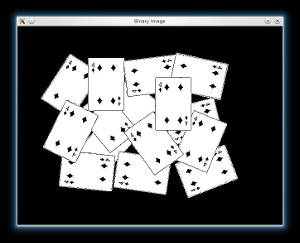
// Create binary image from source image

Mat bw;

[cvtColor](https://docs.opencv.org/master/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab)(imgResult, bw, [COLOR\_BGR2GRAY](https://docs.opencv.org/master/d8/d01/group__imgproc__color__conversions.html#gga4e0972be5de079fed4e3a10e24ef5ef0a353a4b8db9040165db4dacb5bcefb6ea));

[threshold](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57)(bw, bw, 40, 255, [THRESH\_BINARY](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59) | [THRESH\_OTSU](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a95251923e8e22f368ffa86ba8bce87ff));

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Binary Image", bw);



* We are ready now to apply the Distance Transform on the binary image. Moreover, we normalize the output image in order to be able visualize and threshold the result:

// Perform the distance transform algorithm

Mat dist;

[distanceTransform](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ga8a0b7fdfcb7a13dde018988ba3a43042)(bw, dist, [DIST\_L2](https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa2bfbebbc5c320526897996aafa1d8ebaff0d1f5be0fc152a56a9b9716d158b96), 3);

// Normalize the distance image for range = {0.0, 1.0}

// so we can visualize and threshold it

[normalize](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga1b6a396a456c8b6c6e4afd8591560d80)(dist, dist, 0, 1.0, [NORM\_MINMAX](https://docs.opencv.org/master/d2/de8/group__core__array.html#ggad12cefbcb5291cf958a85b4b67b6149fa9f0c1c342a18114d47b516a88e29822e));

[imshow](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)("Distance Transform Image", dist);

