WenQuanYi Micro Hei [Scale=0.9]WenQuanYi Micro Hei Mono song-WenQuanYi Micro Hei sfWenQuanYi Micro Hei "zh" = 0pt plus 1pt

OpenCV

åŔŚåÿČ 1.0

2019 åźť 05 ælJĹ 09 æŮě

Contents

1 OpenCVåÿÿçŤÍæŞ■ä¡IJ **1**
1.1 èŐůåŔŰæĽğèąŇæŮűéŮt' 1

2 åŕźåŻ¿åČŔçŽĎå§žæIJňæŞ■ä¡IJ **2**
2.1 åŁăè¡¡ãĂĄä£őæŤźåŠŇä£Ĺå■ŸåŻ¿åČŔ 2
2.2 æŸ¿çd'žåŻ¿åČŔïijĹcv::namedWindos äÿŐcv::imshowïijL' 2
2.3 Matåŕźèšą 3
2.4 åŕźåČŔçt'ăçŽĎæŞ■ä¡IJ 6

3 åd'ĎçŘĘåŻ¿åČŔçŽĎåÿÿçŤÍæŰźæşŢ **9**
3.1 ç§ľéŸţæŐl'èĘIJæŞ■ä¡IJ 9
3.2 å§žæIJňéŸĹåĂijæŞ■ä¡IJ 11

4 åŻ¿åČŔæżd'æşć **16**
4.1 èĞłåőŽäźL'çž£æĂğæżd'æşć 16

5 åd'ĎçŘĘè¿źçijŸ **18**
5.1 å■ůçğŕè¿źçŢŇéŮőécŸåŔŁåĚűåd'ĎçŘĘ 18
5.2 SobelçőŮå■Ř 21
5.3 LaplaciançőŮå■Ř 24
5.4 CannyçőŮæşŢ 25

6 æĺąæİ£åŇźéĚ■ **29**
6.1 æĺąæİ£åŇźéĚ■äżŃçż■ 29
6.2 æĺąæİ£åŇźéĚ■çŤĺåĹřçŽĎçőŮæşŢ 29
6.3 APIäżŃçż■ 29
6.4 äżčçăĄæijŤçd'ž 30

7 æŢřæ■őäżŃçż■ **33**

8 æŢřæ■őécĎåd'ĎçŘĘ **34**

9 ä¡£çŤÍLeNetåőđçŐřçňŚèĎÿæčĂæţŃåĹĘçśż **37**

10 äžžèĎÿæčĂæţŃåőđçŐř **40**
10.1 detector.detectMultiScale 42

CHAPTER 1

OpenCVåÿÿçŤÍæŞ■ä¡IJ

1.1 èŐůåŔŰæĽğèąŇæŮűéŮť

getTickCount()ïijŽçŤÍäžŐè£ŤåŻđäžŐæŞ■ä¡IJçşżçż§åŘŕåŁĺåĹřå¡ŞåĽ■æĽĂçżŔçŽĎèőąæŮűåŠĺæI

getTickFrequency()ïijŽçŤÍäžŐè£ŤåŻđCPUçŽĎéćŞçŐĞãĂĆget Tick Frequency㥹ŹéĞŇçŽĎå■Ţä¡■æŸŕçğŠïijŇäź§åřśæŸŕäÿĂçğŠåĘĚéĞ■åď■çŽĎæňąæŢřãĂĆ

```
double t = getTickCount();
Mat kernal = (Mat_<char>(3, 3) << 0, -1, 0,
                                  -1,␣
↪5, -1,
                                  0, -1,␣
↪0);
filter2D(src, dst, src.depth(), kernal);
double timecousume = (getTickCount() - t) / getTickFrequency();
printf("time consume %.3f", timecousume);
```

è£ŹéĞŇæŸŕåŕźç§ľéŸţæŐľèĘIJæŞ■ä¡IJçŽĎäÿĂäÿłæŮűéŮťçż§èőąïijŇåĚűäÿ■srcåŠŇdstéČ¡æŸŕäźŃåĽ

CHAPTER 2

åŕźåŻ¿åČŔçŽĎå§žæIJňæŞ■ä¡lJ

2.1 åŁăè¡¡ãĂĄä£őæŤźåŠŇä£İå■ŸåŻ¿åČŔ

2.1.1 åŁăè¡¡åŻ¿åČŔïijĹcv::imreadïijĽ

imreadåŁ§èČ¡æŸŕåŁăè¡¡åŻ¿åČŔæŰĞäžűæĹŘäÿžäÿĂäÿłMatåŕźèśąïijŇåĚűäÿ■çňňäÿĂäÿłåŔĆæŢřèąĺçc
çňňäžŇäÿłåŔĆæŢřïijŇèąĺçd'žåŁăè¡¡çŽĎåŻ¿åČŔæŸŕäżĂäźĹçśžåđŃïijŇæŤŕæŇĄåÿÿèğĄçŽĎäÿL'äÿłåŔ

- IMREAD_UNCHANGED (<0) èąĺçd'žåŁăè¡¡åŐ§åŻ¿ïijŇäÿ■åĄŽäżżä¡ŢæŤźåŔŸ
- IMREAD_GRAYSCALE (0)èąĺçd'žæŁŁåŐ§åŻ¿ä¡IJäÿžçĄřåžęåŻ¿åČŔåŁăè¡¡è£Żæİě
- IMREAD_COLOR (>0) èąĺçd'žæŁŁåŐ§åŻ¿ä¡IJäÿžRGBåŻ¿åČŔåŁăè¡¡è£Żæİě

æşĺæĎŔïijŽOpenCVæŤŕæŇĄJPGãĂĄPNGãĂĄTIFFç■L'åÿÿèğĄæăijåijŔåŻ¿åČŔæŰĞäżűåŁăè¡¡

2.2 æŸ¿çd'žåŻ¿åČŔïijĹcv::namedWindos äÿŐcv::imshowïijĽ

namedWindosåŁ§èČ¡æŸŕåĹŻåżžäÿĂäÿłOpenCVçłŮåŔčïijŇåőČæŸŕçŤśOpenCVèĞłåŁĺåĹŻåżžäÿŐéĞŁ
åÿÿèğĄçŤĺæşŢnamedWindow(âĂIJWindow TitleâĂİ, WINDOW_AUTOSIZE)

- WINDOW_AUTOSIZEäijŽèĞłåŁĺæăźæ■őåŻ¿åČŔåd'ğářŔïijŇæŸ¿çd'žçłŮåŔčåd'ğářŔïijŇäÿ■èČ¡äžžäÿ
- WINDOW_NORMAL,èů§QTéŻĘæĹŘçŽĎæŮűåĂŹäijŽä¡£çŤĺïijŇåĚĄèőÿä£őæŤźçłŮåŔčåd'ğářŔ

imshowæăźæ■őçłŮåŔčåŘ■çğřæŸ¿çd'žåŻ¿åČŔåĹřæŇĞåőŽçŽĎçłŮåŔčäÿŁåŐżïijŇçňňäÿĂäÿłåŔĆæŢřæ

2.2.1 ä£őæŤźåŻ¿åČŕïijĹcv::cvtColorïijĽ

cvtColorçŽĎåŁ§èČ¡æŸŕæŁŁåŻ¿åČŕäżŐäÿĂäÿłå¡l'èL'šçl'žéŮt'è¡ňæ■ćåĹřåŕęåd'ŰäÿĂäÿłèL'šå¡l'çl'žéŮt ãĂĄCOLOR_BGR2GRAYç■L'

```
cvtColor( image, gray_image, COLOR_BGR2GRAY );
```

2.2.2 ä£Ìå■ŸåŻ¿åČŕïijĹcv::imwriteïijĽ

ä£Ìå■ŸåŻ¿åČŕæŰĞäżűåĹřæŇĞåőŽçŻőå¡Ţèůŕå¿Ď

åŕłæIJL'8ä¡■ãĂĄ16ä¡■çŽĎPNGãĂĄJPGãĂĄTiffæŰĞäżűæäijåijŔèĂŇäÿŤæŸŕå■ŢéĂŽéĄŞæĹŰèĂĚäÿ

ä£Ìå■ŸPNGæäijåijŔçŽĎæŮűåĂŹåŕŕäżěä£Ìå■ŸéĂŕæŸŐéĂŽéĄŞçŽĎåŻ¿çL'Ğ

åŕŕäżěæŇĞåőŽåŐŃçijl'åŔĆæŢř

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;

int main(int argc, char** argv) {
        Mat src = imread("D:/1.jpg");
        if (src.empty()) {
                printf("Could not load image...\n");
                return -1;
        }

        namedWindow("test opencv setup", CV_WINDOW_AUTOSIZE);
        imshow("test opencv setup", src);

        namedWindow("output windows", CV_WINDOW_AUTOSIZE); //
↪èĞłåŁĺå¡ćåijŔåŻ¿åČŔåd'ğåřŔïijŇèĂŇäÿŤäÿ■åŕŕæŤźåŔŸ
        Mat output_image;
        cvtColor(src, output_image, CV_BGR2HLS);
        imshow("output windows", output_image);

        imwrite("D:/2.png", output_image);

        waitKey(0);
        return 0;
}
```

2.3 Matåŕźèšą

MatåŕźèšąOpenCV2.0äźŃåŘŐåijŢè£ŽçŽĎåŻ¿åČŔæŢřæ■őçżŞædĎãĂĄèĞłåŁĺåĹĘéĚ■åĘĚå■ŸãĂĄäÿ■

cv::MatåĹĘäÿžäÿd'äÿłéČĺåĹĘïijŇåd'ť'éČĺäÿŐæŢřæ■őéČĺåĹĘãĂĆåd'ť'éČĺåŇĚåŘńäžĘç§l'éŸţçŽĎæL'Ăa
æŢřæ■őåİŮåŇĚåŘńäžĘåŻ¿åČŕäÿ■æL'ĂæIJL'åČŕçť'ăçŽĎåĂijãĂĆ

2.3.1 MatåŕźèśąæđĎéĂăåĞ¡æŢřäÿŐåÿÿçŤĺæŰźæşŢ

åÿÿçŤĺæđĎéĂăåĞ¡æŢř

Mat ()
Mat (int rows, int cols, int type)
Mat (Size size, int type)
Mat (int rows, int cols, int type, const Scalar &s)
Mat (Size size, int type, const Scalar &s)
Mat (int ndims, const int *sizes, int type)
Mat (int ndims, const int *sizes, int type, const Scalar &s)
image

æIJĂäÿżèęĄçŽĎæŸŕåL'■éİćäÿd'äÿL'äÿłïijŇåÿÿçŤĺãĂĆ

åÿÿçŤĺæŰźæşŢ

- **Mat clone()ïijŽåőŇåĚĺæŃůèť'İ** éęŰåĚĹsrcéĂŽè£ĞimreadèŕźåŔŰäÿĂåijååŻ¿åČŕïijŇæŐěäÿŃæİěïijŽ

```
Mat dst = src.clone();
imshow("output", dst);
```

åŕśäijŽåőŇåĚĺåĚŃéŽĘäÿĂåijääÿĂæĺąäÿĂæăůçŽĎåŻ¿åČŔãĂĆ

- **void copyTo(Mat mat)ïijŽåőŇåĚĺæŃůèť'İ**

```
Mat dst;
src.copyTo(dst);
imshow("output", dst);
```

éČĺåĹĘåd'■åĹűäÿĂèĹňæČĚåĘţäÿŃåŔłäijŽåd'■åĹűMatåŕźèśąçŽĎåd'ť'åŠŇæŇĞéŠĹéČĺåĹĘïijŇäÿ■äijŽ

```
Mat A= imread(imgFilePath);
Mat B(A);
```

åőŇåĚĺåd'■åĹűåęĆæđIJæČşæŁŁMatåŕźèśąçŽĎåd'ť'éČĺåŠŇæŢřæ■őéČĺåĹĘäÿĂèţůåd'■åĹűïijŇåŔŕäžěé

```
Mat F = A.clone(); æĹŰ Mat G; A.copyTo(G);
```

- **void convertTo(Mat dst, int type)ïijŽçŤĺäžŐè£ŻèąŇæŢřæ■őçśžåđŃè¡ňæ■ć**
-

åęĆæŁŁCV_8UC1è¡ňæ■ćåĹřCV32F1åőđçŐřåęĆäÿŃïijŽ

```
src.convertTo(dst, CV_32F);
```

- **int channels()ïijŽæ§ěçIJŃåŻ¿åČŔéĂŽéĄŞæČĚåEţ**

```
Mat dst;
cvtColor(src, dst, CV_BGR2GRAY);
printf("input image channels : %d\n", src.channels());
printf("output image channels : %d", dst.channels());
imshow("output", dst);
```

åŘŕäžěåŔŚçŐřåŔŸæĹŘçĄřåžęåŻ¿åČŔäźŃåŘŐè¿ŞåĞžéĂŽéĄŞæŢřåŔŸäÿž1

- **uchar* ptr(i=0)ïijŽèŐůåŔŰåŻ¿çL'ĞåČŔçt'ăåĚůä¡ŞåĂijïijŇièąĺçd'žèąŇæŢř**

```
Mat dst;
cvtColor(src, dst, CV_BGR2GRAY);
const uchar *firstRow = dst.ptr<uchar>(0);
printf("first pixel value : %d", *firstRow);
imshow("output", dst);
```

è£ŹæåůæĹŚäžňåřšåŔŕäžěèŐůåŔŰçňňäÿĂèąŇçňňäÿĂäÿłåČŔçt'ăçŽĎçĄřåžęåĂijä£ąæĄŕäžEąĂĆ

- **.cols, .rowsïijŽèŐůåŔŰèąŇæŢřåŠŇåĹŮæŢř**

```
Mat dst;
cvtColor(src, dst, CV_BGR2GRAY);
int cols = dst.cols;
int rows = dst.rows;
printf("rows = %d cols = %d", rows, cols);
imshow("output", dst);
```

æđĎéĂååĞ¡æŢřçżğçż■äÿ¿ä¿Ń(MatåŕźèśąåĹŻåžž)

åęĆæđIJæŸŕäÿL'äÿłéĂŽéĄŞïijŽ

```
Mat M(3, 3, CV_8UC3, Scalar(0, 0, 255));  //
↪scaleèęĄåŠŇéĂŽéĄŞæŢřçŻőäÿĂèĞt'
cout << "M:" << endl << M << endl;
```

åĚűäÿ■åL'■äÿd'äÿłåŔĆæŢřåĹEąĹńèąĺçd'žèąŇ(row)èů§åĹŮ(column)ãĂĄçňňäÿL'äÿłCV_8UC3äÿ■

ScalarçżŹæŕŘäÿĂäÿłéĂŽéĄŞètŃäÿĂäÿłåĂijïijŇçňňäÿĂäÿłåŠŇçňňäžŇäÿłéĂŽéĄŞçŽĎåĂijåĚĺéČĺéČ¡æ

```
[  0,   0, 255,   0,   0, 255,   0,   0, 255;
   0,   0, 255,   0,   0, 255,   0,   0, 255;
   0,   0, 255,   0,   0, 255,   0,   0, 255]
```

åęĆæđIJåŔŸæĹŘäžEąÿĂäÿłéĂŽéĄŞïijŽ

```
Mat M(100, 100, CV_8UC1, Scalar(127));  //
↪scaleèęĄåŠŇéĂŽéĄŞæŢřçŻőäÿĂèĞt'
```

creatåĹŻåżžåŕźèšą

```
Mat m1;
m1.create(src.size(), src.type());
m1 = Scalar(0, 0, 255);
imshow("output", m1);
```

éŻűåĹìåğŃåŇŰ

```
Mat m2 = Mat::eye(2, 2, CV_8UC1);
cout << "m2 = " << endl << m2 << endl;
```

2.4 åŕźåČŔçťăçŽĎæŞ■ä¡IJ

2.4.1 èŕżåĘŹåČŔçťă

èŕżåĘŹå■ŢéĂŽéĄŞåČŔçťă

- **èŕżäÿĂäÿłGRAYåČŔçťăçĆźçŽĎåČŔçťăåĂijïijĹCV_8UC1ïijL'**

```
Scalar intensity = img.at<uchar>(y, x);
//æĹŰèĂĚ
Scalar intensity = img.at<uchar>(Point(x, y));
```

åĚůä¡Şäżčçă Ąïij Ž

```
        //å■ŢéĂŽéĄŞ
        Mat gray_src;
        cvtColor(src, gray_src, CV_BGR2GRAY);
        int height = gray_src.rows;
        int width = gray_src.cols;
        for (int row = 0; row < height; row++) {
                for (int col = 0; col < width; col++) {
                        int gray = gray_src.at<uchar>(row, col);
                        gray_src.at<uchar>(row, col) = 255 - gray;
                }
        }
        imshow("output", gray_src);
```

è£Źäÿłäżčçă ĄåőđçŐřæŕŔäÿłåČŔçť ăçŽĎåČŔçť ăåĂijå¿Ůç£żè¡ňæŞ■ä¡IJãĂĆ

èŕžåĘŹäÿĽéĂŽéĄŞçŽĎdzžåžŔæŸŕblueãĂągreenãĂąredçŽĎucharçśžåđŃæŢřæ■őãĂĆ

```
Vec3f intensity = img.at<Vec3f>(y, x);
float blue = intensity.val[0];
float green = intensity.val[1];
float red = intensity.val[2];
```

Vec3båŕźåžŤäÿL'éĂŽéĄŞçŽĎéąžåžŔæŸŕblueãĂągreenãĂąredçŽĎucharçśžåđŃæŢřæ■őãĂĆ

Vec3fåŕźåžŤäÿL'éĂŽéĄŞçŽĎfloatçśžåđŃæŢřæ■ő

åĚůä¡Şäžčçăąåę́äÿŃïijŽ

```
        //äÿL'éĂŽéĄŞ
        Mat dst;
        dst.create(src.size(), src.type());
        int height = src.rows;
        int width = src.cols;
        int nc = src.channels();

        for (int row = 0; row < height; row++) {
                for (int col = 0; col < width; col++) {
                        if (nc == 1) //
↪åę́ĆæđIJæŸŕå■ŢéĂŽéĄŞïijŇæŇL'çĚğåŐ$åĚĹçŽĎæŰźåijŔè£ŽèąŇåd'ĎçŘĘ
                        {
                                int gray = dst.at<uchar>(row, col);
                                dst.at<uchar>(row, col) = 255 -␣
↪gray;
                        }
                        else if (nc == 3)  //åę́ĆæđIJæŸŕäÿL'éĂŽéĄŞ
                        {
                                int b = dst.at<Vec3b>(row, col)[0];
                                int g = dst.at<Vec3b>(row, col)[1];
                                int r = dst.at<Vec3b>(row, col)[2];
                                dst.at<Vec3b>(row, col)[0] = 255 -␣
↪b;
                                dst.at<Vec3b>(row, col)[1] = 255 -␣
↪g;
                                dst.at<Vec3b>(row, col)[2] = 255 -␣
↪r;
                        }

                }
        }
```

å¡ŞçĎűïijŇæIJL'æŻť'äÿžçőĂå■ŢçŽĎæŞ■ä¡IJïijŇçŤĺäžŐåőđçŐřåŻ¿åČŔåČŔçť'ă255 - pixel

```
        //äÿL'éĂŽéĄŞ
        Mat dst;
        dst.create(src.size(), src.type());
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
        bitwise_not(src, dst);
```

bitwiseæŸŕä¡■æŞ■ä¡IJïijŇnotæŸŕéİđæŞ■ä¡IJãĂĆäź§åřśæŸŕ1åŔŸæĹŘ0,0åŔŸæĹŘ1ãĂĆ

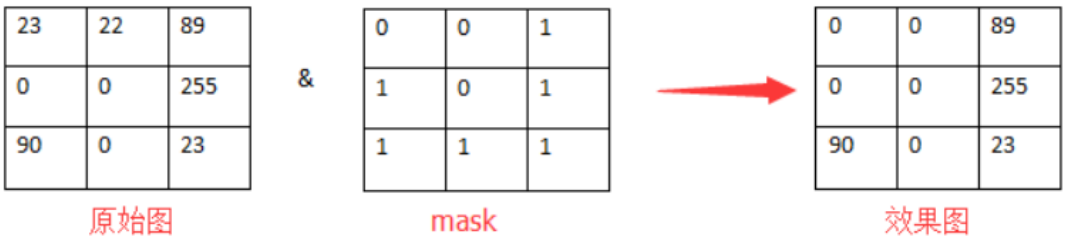
CHAPTER 3

åďĎçŘĘåŻ¿åČŔçŽĎåÿÿçŤĺæŰźæşŢ

3.1 ç§ľéŸţæŐľèĘIJæŞ■ä¡IJ

æĽĂèřŞæŐľèĘIJåĚűåőđåřśæŸŕäÿĂäÿłç§ľéŸţïijŇçĎűåŘŐæăźæ■őè£ŹäÿłÇ§ľéŸţéĞ■æŰřèőąçőŮåŻ¿çĽ

比如一个3 * 3的图像与3 * 3的掩膜进行运算，得到的结果图像就是：



image

è£ŹéĞŇïijŇæĹŚäžňçŤĺæŐľèĘIJæİěæŔŘéńŸåŻ¿åČŔåŕźæŕŤåžęãĂĆçŤĺåĹřçŽĎæŐľèĘIJæŸŕïijŽ

$$I(i,j) = 5 * I(i,j) - [I(i-1,j) + I(i+1,j) + I(i,j-1) + I(i,j+1)]$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

image

çžćèL’šæŸŕäÿ■å£ČåČŔçť ăïijŇäżŐäÿŁåĹřäÿŃïijŇäżŐåůęåĹřåŔşåŕźæŕŔäÿłåČŔçť ăåĄŽåŘŇæăůçŽĎåď’ĺ

ä¡£çŤĺæŰźæşŢåęĆäÿŃïijŽ

åőŽäźL’æŐl’èĘIJïijŽMat kernel = (Mat_(3,3) << 0, -1, 0, -1, 5, -1, 0, -1, 0);

filter2D(src, dst, src.depth(), kernel);åĚűäÿ■srcäÿŐdstæŸŕ-MatçśžåđŃåŔŸéĞŔãĂĄsrc.depthèąĺçd’žä¡■åŻ¿æůšåžęïijŇæIJL’32ãĂĄ24ãĂĄ8ç■L’ãĂĆ

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace cv;
using namespace std;

int main(int argc, char** argv) {
        Mat src;
//        src = imread("D:/1.jpg");
        src = imread("D:/WireRope/change.jpg");
        if (!src.data) {
                printf("could not load image...\n");
                return -1;
        }
        namedWindow("input image", CV_WINDOW_AUTOSIZE);
        imshow("input image", src);

        Mat dst;
        dst = Mat(src.size(), src.type());
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
        Mat kernal = (Mat_<char>(3, 3) << 0, -1, 0,
↪-1, 5, -1,
↪ 0, -1, 0);
        filter2D(src, dst, src.depth(), kernal);
        imshow("contract image demo", dst);
        imwrite("D:/WireRope/contrast_change.jpg", dst);

        waitKey(0);
        return 0;
}
```

çŤÍè£Źçğ■æŰźæşŢåřśæŔŘéńŸäžĘåŻ¿åČŔçŽĎåŕźæŕŤåžęãĂĆæĹŚäžňçżŹfilter2DäżĂäźĹæăůçŽĎæŐľè

3.1.1 åČŔçťäèŇČåŻťåďĎçŘĘsaturate_cast

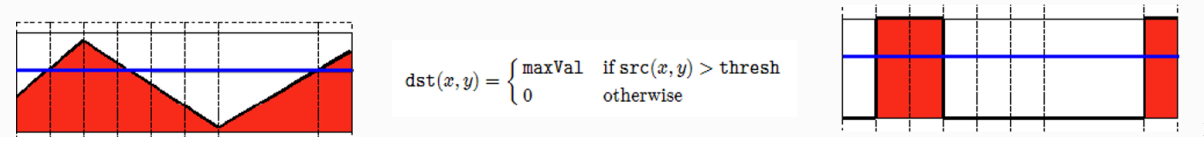
- saturate_castïijĹ-100ïijL'ïijŇè£ŤåŻđ 0ãĂĆ
- saturate_castïijĹ288ïijL'ïijŇè£ŤåŻđ255
- saturate_castïijĹ100ïijL'ïijŇè£ŤåŻđ100

è£ŹäÿłåĞ¡æŢřçŽĎåŁ§èČ¡æŸŕçąőä£ĺRGBåĂijå¿ŮèŇČåŻť'åIJĺ0~255äźŃéŮť'

3.2 å§žælJňéŸĹåĂijæŞ■ä¡lJ

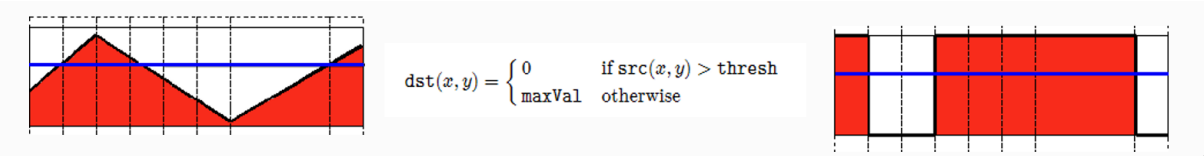
- **éŸĹåĂijäžŇåĂijåŇŰ(threshold binary)**

èŞĺèL'šèąĺçd'žéŸĹåĂijçž£ïijŇçžćèL'šèąĺçd'žåČŔçt'äçŽĎåĹĘåÿČæČĚåĘţãĂĆéIJĂèęĄæşĺæĎŔçŽĎæŸŕè



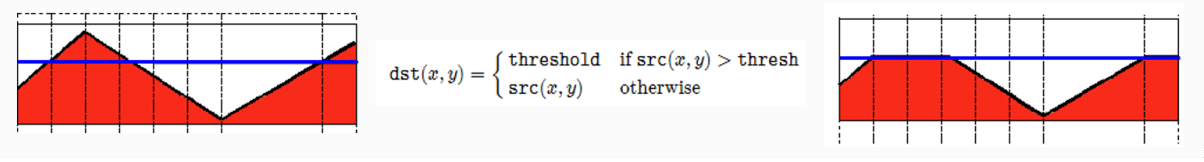
image

- **éŸĹåĂijåŔ■äžŇåĂijåŇŰ(threshold binary Inverted)**



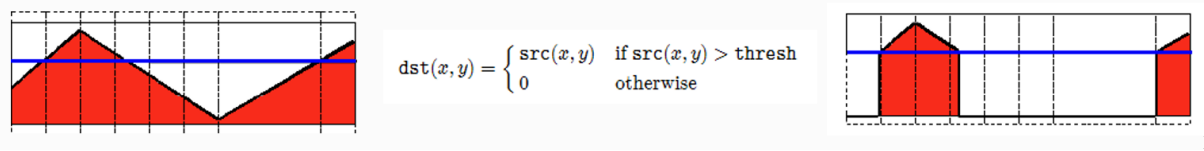
image

- **æĹłæŰ■ (truncate)**



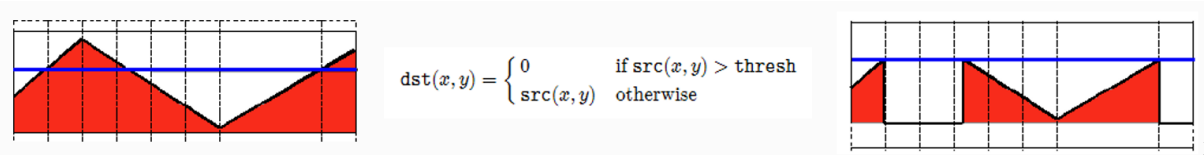
image

- **éŸÍåĂijåŔŰéŽű (threshold to zero)**



image

- **éŸÍåĂijåŔ■åŔŰéŽű (threshold to zero inverted)**



image

è£ŹéĞŇçżŹåĞžæŇĞäżd'åŘ■çğřïijŽ

Enumerator	
THRESH_BINARY	$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC	$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_TOZERO	$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_MASK	
THRESH_OTSU	flag, use Otsu algorithm to choose the optimal threshold value
THRESH_TRIANGLE	flag, use Triangle algorithm to choose the optimal threshold value

image

èęĄç§ěéĄŞçŽĎæŸŕïijŇè£ŹäžŤäÿłåĂijåžŤèŕěèćńåőŔåőŽäźL'è£ĞïijŇåðđéŹĚäÿŁåĚűèČŇåŘŐåĹĘåĹńæ
4ïijŇæL'ĂäżěåĘŹæĹŘæŢřå■Ů0-4äź§æŸŕåŔŕäžěçŽĎãĂĆ

äÿŃéÌćæŸŕé**ŸÍåĂijäžŇåĂijåŇŰ**çŽĎäżčçăĄïijŽ

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
using namespace cv;

Mat src, dst, gray_src;
int threshold_value = 127; //åČŕçt'ăåĂijæŸŕ0-
↪255ïijŇåŔŰäÿ■éŮt'åĂijåřśæŸŕ127
int threshold_max = 255;
const char* output_title = "binary image";

void Threshold_Demo(int, void*);

int main(int argc, char** argv) {
        src = imread("D:/2.jpg");
        if (!src.data) {
                printf("could not load image...\n");
                return -1;
        }
        namedWindow("input image", CV_WINDOW_AUTOSIZE);
        namedWindow(output_title, CV_WINDOW_AUTOSIZE);
        imshow("input image", src);


        createTrackbar("Threshold Value", output_title, &threshold_
↪value, threshold_max, Threshold_Demo); //åĹżåżžäÿĂäÿłæŃŰåŁĺæİą
        Threshold_Demo(0, 0);

        waitKey(0);
        return 0;
}


void Threshold_Demo(int, void*) {
        cvtColor(src, gray_src, CV_BGR2GRAY);
        threshold(gray_src, dst, threshold_value, threshold_max,␣
↪THRESH_TOZERO_INV);  //
↪æŻt'æŤźè£ŹéĞŃçŽĎåĂijïijŇåřśåŔŕäżěåőđçŐř5çğ■éŸĹåĂijæŞ■ä¡IJè¡ňæ■ć
        imshow(output_title, dst);
}
```

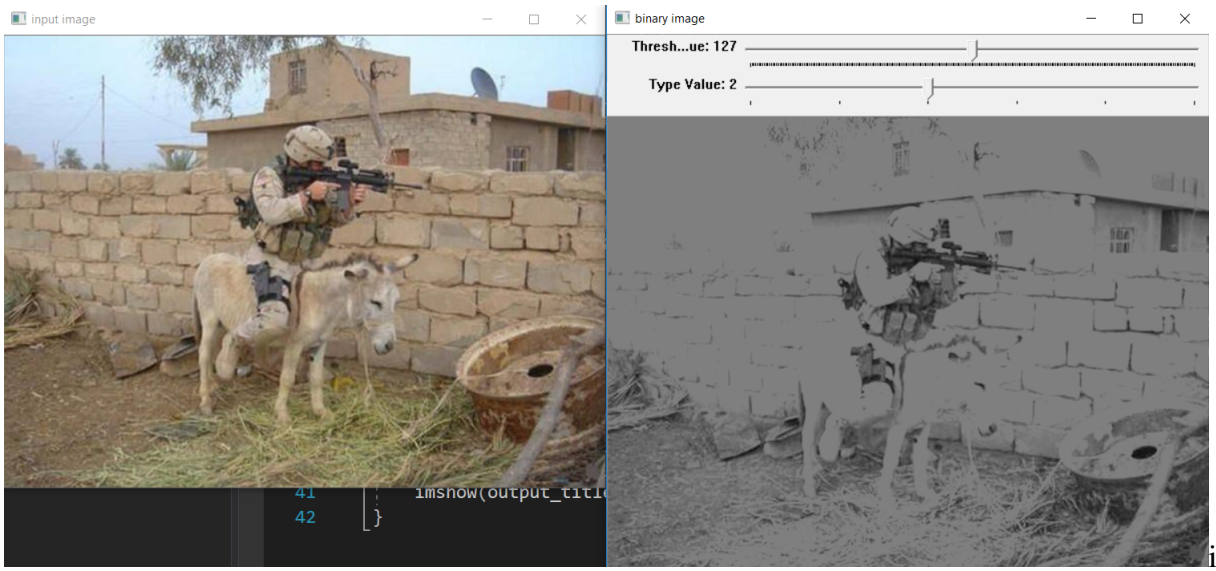
äÿŃéİćæŸŕ5äÿłåĂijäÿĂèţůåĹŻåżžæŃŰåŁĺæİąçŽĎæŢĹæđIJïijŽ

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace cv;

Mat src, dst, gray_src;
int threshold_value = 127; //åČŕçt'ăåĂijæŸŕ0-
↪255ïijŇåŔŰäÿ■éŮt'åĂijåřśæŸŕ127
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
int threshold_max = 255;

int type_value = 2;
int type_max = 4;   //åőŽäźL'5çğ■æŞ■ä¡IJ,
↪5çğ■æŞ■ä¡IJæL'Ääŕźåžt̃çŽĎåőđéŹĚåĂijåĹĘåĹńæŸŕ0-4

const char* output_title = "binary image";

void Threshold_Demo(int, void*);

int main(int argc, char** argv) {
        src = imread("D:/test.jpg");
        if (!src.data) {
                printf("could not load image...\n");
                return -1;
        }
        namedWindow("input image", CV_WINDOW_AUTOSIZE);
        namedWindow(output_title, CV_WINDOW_AUTOSIZE);
        imshow("input image", src);


        createTrackbar("Threshold Value", output_title, &threshold_
↪value, threshold_max, Threshold_Demo); //åĹżåżžäÿĂäÿłæŃŰåŁĺæİą
        createTrackbar("Type Value", output_title, &type_value,␣
↪type_max, Threshold_Demo); //åĹżåżžäÿĂäÿłæŃŰåŁĺæİą
        Threshold_Demo(0, 0);

        waitKey(0);
        return 0;
}


void Threshold_Demo(int, void*) {
        cvtColor(src, gray_src, CV_BGR2GRAY);
        threshold(gray_src, dst, threshold_value, threshold_max,␣
↪type_value);  //
↪æŻt'æŤźè£ŹéĞŃçŽĎåĂijïijŇåřśåŔŕäżěåőđçŐř5çğ■éŸĹåĂijæŞ■ä¡IJè¡ňæ■ć
        imshow(output_title, dst);
}
```



image

- **èĞłåŁÍèőąçőŮéŸÍåĂij**

  åőđçŐřæŰźæşŢæŸŕTHRESH_OTSUåŠŇTHRESH_TRIANGLE

  è£Źäÿď'çğ■æŰźæşŢåŔŕäżěçŤÍäžŐèĞłåŁÍèőąçőŮéŸÍåĂijïijŇèČŇåŘŐéČ¡æIJL'äÿĂäžŻåŕźåžŤçŽĎæŢřå

```
threshold(gray_src, dst, threshold_value, threshold_max, THRESH_
↪OTSU | type_value); //
↪æŻt'æŤźè£ŹéĞŇçŽĎåĂijïijŇåřśåŔŕäżěåőđçŐř5çğ■éŸÍåĂijæŞ■ä¡IJè¡ňæ■ć
```

  æŻť'æŤźè£ŹéĞŇåřśåŔŕäżěåőđçŐřèĞłåŁÍæśĆéŸÍåĂijãĂĆ

CHAPTER 4

åŻ¿åČŔæżďæşć

4.1 èĞłåőŽäźĽçž£æĂğæżďæşć

4.1.1 RobertçőÛå■Ř

```
        //RobertçőÛå■ŘåIJÍxæŰźåŘŚ
Mat kernal_x = (Mat_<int>(2, 2) << 1, 0, 0, -1);
filter2D(src, dst, src.depth(), kernal_x, Point(-1, -1), 0.0);
imshow("output image", dst);

        //RobertçőÛå■ŘåIJÍyæŰźåŘŚ
Mat kernal_y = (Mat_<int>(2, 2) << 0, 1, -1, 0);
filter2D(src, dst, src.depth(), kernal_y, Point(-1, -1), 0.0);
imshow("output image", dst);
```

4.1.2 SoberçőÛå■Ř

```
        // SobelçőÛå■ŘåIJÍxæŰźåŘŚ
Mat kernal_x = (Mat_<int>(3, 3) << -1, 0, 1, -2, 0, 2, -1, 0, 1);
filter2D(src, dst, src.depth(), kernal_x, Point(-1, -1), 0.0);

        // SobelçőÛå■ŘåIJÍyæŰźåŘŚ
Mat kernal_y = (Mat_<int>(3, 3) << -1, -2, -1, 0, 0, 0, 1, 2, 1);
filter2D(src, dst, src.depth(), kernal_y, Point(-1, -1), 0.0);
```

4.1.3 æŃĽæŹőæŃĽæŰŕçőŮå■Ř

```
Mat kernal = (Mat_<int>(3, 3) << 0, -1, 0, -1, 4, -1, 0, -1, 0);
filter2D(src, dst, src.depth(), kernal, Point(-1, -1), 0.0);
```

4.1.4 èĞłåőŽäźĽå■ůçğŕfilter2D

```
filter2D(
Mat src, //è¿ŞåĚěåŻ¿åČŔ
Mat dst, // æĺąçşŁåŻ¿åČŔ
int depth, // åŻ¿åČŔæůśåžę32/
 ↪8ïijŇåęĆæđIJäÿ■çŞěéĄŞäÿĂèĹňéČ¡åĘŹézŸèőd'æIJĂåd'ğæůśåžę
Mat kernel, // å■ůçğŕæäÿ/
 ↪æĺąæİ£ïijŇè¿ŞåĚěçŽĎå■ůçğŕæäÿåd'ğářŔäÿĂèĹňéČ¡æŸŕåŞžæŢř3ãĂĄ5ãĂĄ7ãĂĄ9ç■L'ç■L'
Point anchor, // éŤŽçĆźä¡■ç¡őïijŇéŤŽçĆźçŻt'æŐěåĘŹïijĹ-1,-
 ↪1ïijL'åřśäijŽèĞłåŁĺåÿőä¡ăæL'¿åĹřéŤŽçĆźçŽĎäÿ■å£Čä¡■ç¡ő
double delta // èőąçőŮåĞžæİěçŽĎåČŔçt'ă+delta
)
```

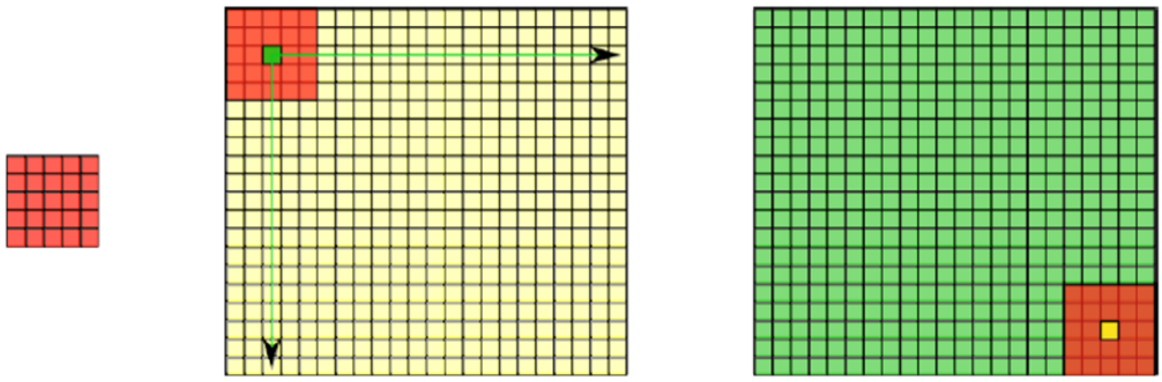
åĚűäÿ■ kernalæŸŕåŔŕäżěèĞłåőŽäźL'çŽĎå■ůçğŕæäÿ

CHAPTER 5

åďĎçŘĘè¿źçijŸ

5.1 å■ůçğŕè¿źçŢŇéŮőécŸåŔŁåĚűåďĎçŘĘ

5.1.1 è¿źçŢŇéŮőécŸ



å■ůçğŕè¿źçŢŇéŮőécŸæŸŕæŇĞçŽĎåŻ¿åČŔå■ůçğŕçŽĎæŮűåĂŹè¿źçŢŇåČŔçť ăïijŇäÿ■èČ¡èćńå■ůçğŕæ

5.1.2 åďĎçŘĘ

åIJÍå■ůçğŕåijĂåğŇäźŃåL’■åćđåŁãè¿źçijŸåČŔçť ăïijŇåąńåĚĚçŽĎåČŔçť ăåĂijäÿž0æĹŰèĂĚRGBéżŚèL’š
BORDER_DEFAULTïijŇæ■ď’åď’Ůåÿÿç ŤÍçŽĎè£ŸæIJL’åęĆäÿŃåĞăçğ■ïijŽ

- **BORDER_CONSTANTïijŽçŤÍæŇĞåőŽåČŔçť’ăåąńåĚĚè¿źçijŸ**
- **BORDER_REPLICATEïijŽçŤÍåůšç§ěè¿źçijŸåČŔçť’ăåĂijæİěåąńåĚĚè¿źçijŸåČŔçť’ăåĂij**

- **BORDER_WRAPïijŽçŤĺåŔęåd'ŰäÿĂè¿źçŽĎåČŔçt'ăæİěèąěåĄ£åąńåĚĚ**

  äÿŃéİćæŸŕçżŹåŻ¿åČŔèĞłåőŽäźL'æůżåŁăè¿źçijŸ

  copyMakeBorderïijŽçżŹåŻ¿åČŔæůżåŁăè¿źçijŸAPI

```
copyMakeBorderïijĹ
  Mat src, // è¿ŞåĚěåŻ¿åČŔ
  Mat dst, // æůżåŁăè¿źçijŸåŻ¿åČŔ
  int top, //␣
↪è¿źçijŸéŢ£åžęïijÑäÿĂèĹñäÿŁäÿŃåůęåŔşéČ¡åŔŰçŻÿåŘÑåĂijïijÑ
  int bottom,
  int left,
  int right,
  int borderType // è¿źçijŸçśżåđŃ
  Scalar value // ScalarçŤĺäžŐæÑĞåőŽécIJèL'š,è¿źçijŸçśżåđŃäÿž␣
↪BORDER_CONSTANT æŮűïijÑæIJL'æŢĹ
ïijL'
```

äÿŃéİćäžčçăĄåśŢçd'žäÿĂäÿŃåęĆä¡ŢåĚůä¡Şä¡£çŤĺïijŽ

```
int top = (int)0.05*src.rows;
int bottom = (int)0.05*src.rows;
int left = (int)0.05*src.cols;
int right = (int)0.05*src.cols;

Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.
↪uniform(0, 255));
copyMakeBorder(src, dst, top, bottom, left, right, borderType,␣
↪color);
imshow(OUTPUT_WIN, dst);
```

è£ŹæŸŕåőÑæŢt'æijŤçd'žåŻŻçğ■æŰźæşŢåęĆä¡ŢåĹĞæ■ćçŽĎäžčçăĄïijŽ

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace cv;

Mat src, dst, kernal;

int main(int argc, char** argv) {
        src = imread("D:/1.jpg");
        if (!src.data) {
                printf("could not load image...\n");
                return -1;
        }

        char INPUT_WIN[] = "input image";
        char OUTPUT_WIN[] = "result image";
```

(äÿŃéąţżğçż■)

(çż■äÿŁéąţ)

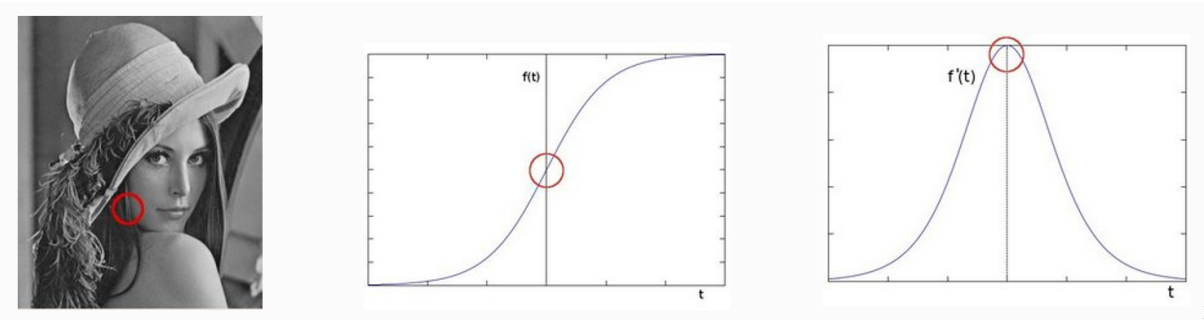
```
        namedWindow(INPUT_WIN, CV_WINDOW_AUTOSIZE);
        namedWindow(OUTPUT_WIN, CV_WINDOW_AUTOSIZE);
        imshow("input image", src);

        int top = 0.05*src.rows;
        int bottom = 0.05*src.rows;
        int left = 0.05*src.cols;
        int right = 0.05*src.cols;

        RNG rng(12345); //çŤŞæĹŘéŽŔæIJžæŢř
        int borderType = BORDER_DEFAULT;

        int c = 0;
        while (true)
        {
                c = waitKey(500);
                if ((char)c ==27 ) //
↪æŇL'äÿŃéŤőçŻŸESCåŕźåžŤçŽĎæŢřåĂijåřśæŸŕ27ïijŇäź$åřśæŸŕæŇL'äÿŃéŤőçŻŸæŐĺåĞžwhileå
                {
                        break;
                }
                if ((char)c == 'r')
                {
                        borderType = BORDER_REPLICATE;
                }
                else if ((char)c == 'w')
                {
                        borderType = BORDER_WRAP;
                }
                else if ((char)c == 'c')
                {
                        borderType = BORDER_CONSTANT;
                }
                else if((char)c == 'd')
                {
                        borderType = BORDER_DEFAULT;
                }
                Scalar color = Scalar(rng.uniform(0, 255), rng.
↪uniform(0, 255), rng.uniform(0, 255)); //çŤŞæĹŘ0-
↪255äźŃåL'■éŽŔæIJžécIJèL'šåĂij
                copyMakeBorder(src, dst, top, bottom, left, right,␣
↪borderType, color);
                imshow(OUTPUT_WIN, dst);
        }

        return 0;
}
```

5.2 SobelçőÙå■Ř

5.2.1 å■ůçğŕçŽĎåžŤçŤĺïijŽåŻ¿åČŕè¿źçijŸæŔŘåŔŰ

è¿źçijŸæŸŕåŻ¿åČŕåČŕçť´ăåŔŚçŤ§æŸ¿èŚÙèůČè£ĄçŽĎåIJřæŰźïijŇéĂŽè£ĞæśĆäÿĂéŸŰåŕijæŢřåŔŕäż

delta = f(x) – f(x-1), deltaèűŁåď´ğïijŇèŕť´æŸŐåČŕçť´ăåIJÍXæŰźåŘŚåŔŸåŇŰèűŁåď´ğïijŇè¿źçijŸä£ąåŔůè



image

5.2.2 SobelçőÙå■Ř

SobelçőÙå■ŘåŔĹèćńçğřäÿžäÿĂéŸűå¿őåĹĘçőÙå■ŘïijŇæśĆåŕijçőÙå■ŘïijŇåIJÍæŕť´åźşåŠŇåđĆçŻť´äÿď´ä
differentiation operatorïijL'ïijŇçŤĺæİěèőąçőÙ**çĄřåžęåŻ¿åČŔ**çŽĎè£ŠäijijæćŕåžęãĂĆ

SobleçőÙå■ŘåŁ§èČ¡éŻĘåŘĹéńŸæŰŕåźşæżŚåŠŇå¿őåĹĘæśĆåŕijãĂĆ

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

image

æĹŚäżñäžěæŕť´åźşæćŕåžęäÿžä¿ŃãĂĆäżŰçŽĎæŕť´åźşæŰźåŘŚäÿŁéİćåŔŸåŇŰå■ĄåĹĘçŽĎæŸŐæŸ¿ïijŇå

$$G = \sqrt{G_x^2 + G_y^2}$$

$$G = |G_x| + |G_y|$$

image

æIJĂçżĹåŻ¿åČŔæćŕåžęåęĆäÿŁåŻ¿æL'Ăçđ´žïijŇäÿĂèĹñäÿžäžĘèől'èőąçőÙæIJžçőÙçŽĎæŻť´å£ńäÿĂäžŻ

5.2.3 SobelçőÙå■ŘAPI

```
cv::Sobel (
InputArray Src // è¿ŞåĚěåŻ¿åČŔ
OutputArray dst// è¿ŞåĞžåŻ¿åČŔïijŇåd'ğåŕŔäÿŐè¿ŞåĚěåŻ¿åČŔäÿĂèĞt'
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
int depth // è¿ŞåĞžåŻ¿åČŕæůśåžę.
Int dx.  //␣
↪XæŰźåŘŚïijÑåĞäéŸűåŕijæŢřïijÑåęĆæđIJæČşæśĆxæŰźåŘŚçŽĎæŮűåĂŹåřśèől'è£ŹäÿłæŢřåŔŰ1ï
int dy // YæŰźåŘŚïijÑåĞäéŸűåŕijæŢř.
int ksize, SOBELçőŮå■Řkernelåd'ğåřŔïijÑå£ĚéążæŸŕ1ãĂĄ3ãĂĄ5ãĂĄ7ãĂĄ
double scale  = 1
double delta = 0
int borderType = BORDER_DEFAULT
)
```

Input depth (src.depth())	Output depth (ddepth)
CV_8U	-1/CV_16S/CV_32F/CV_64F
CV_16U/CV_16S	-1/CV_32F/CV_64F
CV_32F	-1/CV_32F/CV_64F
CV_64F	-1/CV_64F

image

è£ŹéĞŇåĚşäžŐæůśåžęèŕť'äÿĂäÿŃïijÑåŻääÿžèĂČèŹŚäÿd'äÿłåŻ¿åČŕåČŕçť'ääźŃéŮť'çŽĎåůőåĂijïijÑåŀ
1åřśæŸŕèąĺçd'žéĂL'æŃľ'åŠŇåŐ§åĚĹçŽĎäÿĂæăůïijL'

SobelçőŮå■ŘæŤźè£ŻçL'ĹïijŽScharr

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$$

image

```
cv::Scharr (
InputArray Src // è¿ŞåĚěåŻ¿åČŕ
OutputArray dst// è¿ŞåĞžåŻ¿åČŕïijÑåd'ğåřŕäÿŐè¿ŞåĚěåŻ¿åČŕäÿĂèĞt'
int depth // è¿ŞåĞžåŻ¿åČŕæůśåžę.
Int dx.  // XæŰźåŘŚïijÑåĞäéŸűåŕijæŢř
int dy // YæŰźåŘŚïijÑåĞäéŸűåŕijæŢř.
double scale  = 1
double delta = 0
int borderType = BORDER_DEFAULT
)
```

åd'ĎçŘĘæţĄçÍŃïijŽ

- **GaussianBlur(src, dst, Size(3,3), 0, 0, BORDER_DEFAULT);**

- **cvtColor(src, gray, COLOR_RGB2GRAY);**
- **addWeighted(A, 0.5,B, 0.5, 0, AB);**
- **convertScaleAbs(A, B)// èőąçőŮåŻ¿åČŕAçŽĎåČŕçt'ăçżĺåŕźåĂijïijÑè¿ŞåĞžåĹřåŻ¿åČŕB**

Sobelåőđç Őřäžčçă Ąåę ĆäÿŇïijŇScharräžčçă Ąäÿ Ăæăůã ĂĆ

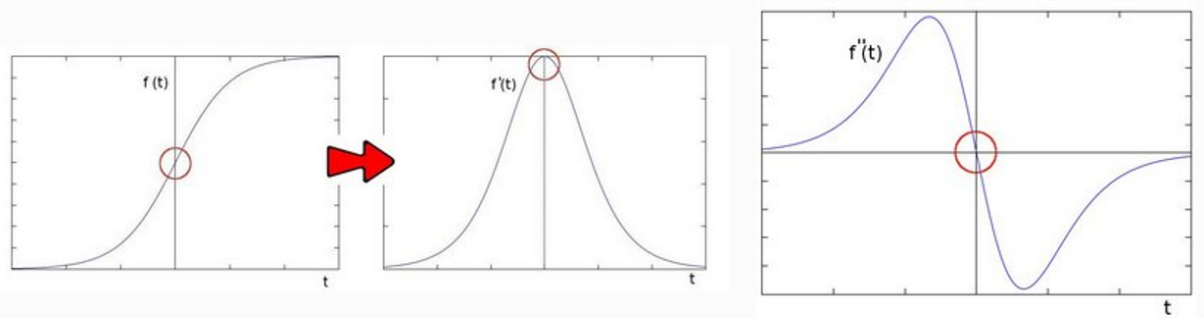
```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace cv;

Mat src, dst;

int main(int argc, char** argv) {
        src = imread("D:/1.jpg");
        if (!src.data) {
                printf("could not load image...\n");
                return -1;
        }
        imshow("input image", src);

        GaussianBlur(src, dst, Size(3, 3), 0, 0);
        Mat gray_src;
        cvtColor(src, gray_src, CV_BGR2GRAY);
        imshow("gray image", gray_src);

        Mat xgrad, ygrad;
        Sobel(gray_src, xgrad, CV_16S, 1, 0, 3);  //
↪æĹŚäżňè£ŹéĞŇårźäžŐCV_
↪8UçŽĎè¿ŞåĚěåŻ¿åČŕijŇåŘŚäÿŁåŘŰäÿĂäÿłæŢřéĞŕçžğïijŇä¡£å¿Ůäÿ■äijžåŘŚçŤŞèúĚè£Ğ255è
        Sobel(gray_src, ygrad, CV_16S, 0, 1, 3);
        convertScaleAbs(xgrad, xgrad);  //
↪èőąçőŮçŽĎæŮűåĂŹäźŞåŘŕèČ¡åĞžçŐřèt'ŞæŢřïijŇèt'ŞæŢřçŽĎèŕİåŻääÿžäÿ■æŸŕ0+
↪255äźŇéŮt'ïijŇäijŽèćńåijžåĹűåŘŸæĹŘ0ïijŇè£Źäÿ■æŸŕæĹŚäżňæČşèęĄçŽĎïijŇæĹŚäżňè£Źæă
        convertScaleAbs(ygrad, ygrad);
        imshow("xgrad", xgrad);
        imshow("ygrad", ygrad);



        Mat xygrad = Mat(xgrad.size(), xgrad.type());

        /*
↪æşĺéĞŁçŽĎäžčçăĄæŸŕäÿ■ä¡£çŤĺåĞ¡æŢřæśĆxgradåŠŇygradçŽĎåŘĹèţůæİěçŽĎåĂij
↪*/
        //int width = xgrad.cols;
        //int height = ygrad.rows;
        //for (int row = 0; row < height; row++)
        //{
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
        //         for (int col = 0; col < width; col++)
        //         {
        //                 int xg = xgrad.at<uchar>(row, col);
        //                 int yg = ygrad.at<uchar>(row, col);
        //                 int xy = xg + yg;
        //                 xygrad.at<uchar>(row, col) = saturate_cast
↪<uchar>(xy);
        //         }
        //}
        addWeighted(xgrad, 0.5, ygrad, 0.5, 0, xygrad);
        imshow("Final result", xygrad);
        waitKey(0);
        return 0;
}
```

5.3 LaplaciançőŮå■Ř



image

åIJÍäžŇéŸűåŕijæŢřçŽĎæŮűåĂŹïijŇæIJĂåď ğåŔŸåŇŰåď ĎçŽĎåĂijäÿžéŻűå■şè¿źçijŸæŸŕéŻűåĂijãĂĆé

5.3.1 cv::Laplacian

```
Laplacian(
InputArray src,
OutputArray dst,
int depth, //æůšåžęCV_16S
int kisze, // 3
double scale = 1,
double delta =0.0,
int borderType = 4
)
```

åď ĎçŘEæţAçÍŃæŸŕ

- **énŸæŰŕræĺąçşŁ – åŐżåŹłåčřGaussianBlur()**
- **è¡ňæ■ćäÿžçAřåžęåŻ¿åČŔcvtColor()**

- **æŃL'æŹőæŃL'æŰŕ – äžŇéŸűåŕijæŢřèőąçőŮLaplacian()**
- **åŔŰçżİåŕźåĂijconvertScaleAbs()**
- **æŸ¿çd'žçżŞæđIJ**

  è£ŹéĞŇåĘ■èŕť'äÿĂäÿŃåŔŰçżİåŕźåĂijçŽĎæĎŔäźL'ïijŇäÿ■çőąçőŮçŽĎåĂijæŸŕèť'§çŽĎè£ŸæŸŕæ■čçŽ

  åĚůä¡ŞåŻ¿åČŔåď'ĎçŘĘäżčçăĄïijŽ

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace cv;

Mat src, dst;

int main(int argc, char** argv) {
        src = imread("D:/1.jpg");
        if (!src.data) {
                printf("could not load image...\n");
                return -1;
        }
        namedWindow("input image", CV_WINDOW_AUTOSIZE);
        imshow("input image", src);

        Mat gray_src, edge_image;
        GaussianBlur(src, dst, Size(3, 3), 0, 0);
        cvtColor(dst, gray_src, CV_BGR2GRAY);

        Laplacian(gray_src, edge_image, CV_16S, 3);
        convertScaleAbs(edge_image, edge_image);
        threshold(edge_image, edge_image, 0, 255, THRESH_OTSU |␣
↪THRESH_BINARY); //
↪çŤÍOtsuçőŮæşŢèŐůåŔŰæIJĂäijŸäžŇåĂijåŇŰçŽĎåĂijè£żèąŇåŻ¿åČŔäžŇåĂijåŇŰåď'ĎçŘĘ
        imshow("output image", edge_image);

        waitKey(0);
        return 0;
}
```

5.4 CannyçőŮæşŢ

CannyæŸŕè¿źçijŸæčĂæţŃçőŮæşŢïijŇåIJÍ1986åźť'æŔŘåĞžçŽĎãĂĆæŸŕäÿĂäÿłå■ĄåĹĘåÿÿçŤĺåŠŇåőđç

5.4.1 çőŮæşŢæţĄçÍŃ

çőŮæşŢåď'ğèĞť'æţĄçÍŃïijŽ

- **énŸæŰŕæĺąçşŁ - GaussianBlur**
- **çĄřåžęè¡ňæ■ć - cvtColor**
- **èőąçőŮæćŕåžę – Sobel/Scharr**
- **éİđæIJĂåd'ğä£ąåŔůæŁŚåĹű**
- **énŸä¡ŐéŸĹåĂijè¿ŞåĞžäžŇåĂijåŻ¿åČŔ**

è£ŹéĞŇèŕť'äÿĂäÿŃéńŸæŰŕæĺąçşŁçŽĎä¡IJçŤĺïijŇéńŸæŰŕæĺąçşŁçŽĎäÿżèęĄä¡IJçŤĺåřśæŸŕéŹ■åŹłãĂĆ

éİđæIJĂåd'ğä£ąåŔůæŁŚåĹűæŸŕåĚşäžŐè¿źçijŸæĹŚäżňåŔłèČ¡æIJL'äÿĂäÿłåČŔçť'ääÿĂäÿłåĂijïijŇåĚ

énŸä¡ŐéŸĹåĂijè£đæŐěæŸŕéIđæIJĂåd'ğä£ąåŔůæŁŚåĹűäźŃåŘŐçŽĎåŻ¿åČŔéČ¡æŸŕäÿĂäžŽåČŔçť'ăç

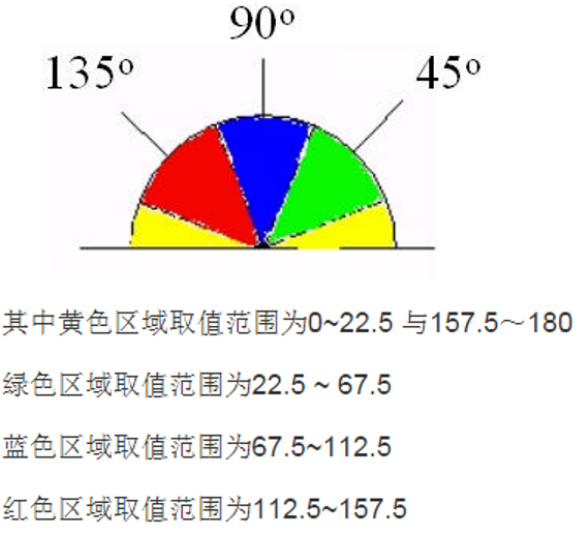
åd'ğæęĆè£ŹæŸŕåőŇæŢť'çŽĎä¡£çŤĺcannyçőŮæşŢçŽĎætĄçĺŃãĂĆ

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan(\frac{G_y}{G_x})$$



image

åęĆåŻ¿æL'Ăçd'žåŻ¿çL'Ğäÿ■çŽĎåůęä¿ğæŸŕSobelçőŮå■ŘïijŇθèąĺçd'žçŽĎæŸŕæćŕåžęçŽĎåŔŸåŇ

énŸä¡ŐéŸĹåĂijçŽĎéĂĽåŔŰ

äżĂäźĹæăůçŽĎéŸĹåĂijæŸŕåě¡çŽĎéŸĹåĂijïijŇéńŸä¡ŐéŸĹåĂijåĹřåžŢèŕěæĂŐäźĹéĂĽ'åŔŰåŚćïij§åIJ

æŐĺè■ŘçŽĎéńŸä¡ŐéŸĹåĂijæŕŤåĂijäÿž T2: T1 = 3:1/2:1 åĚűäÿ■T2äÿžéńŸéŸĹåĂijïijŇT1äÿžä¡ŐéŸĹåĂijãĂĆ

5.4.2 cv::Canny

```
CannyïijĹ
InputArray src, // 8-
 ↪bitçŽĎè¿ŞåĚěåŻ¿åČŔïijÑäÿ■æŤŕæŇĄå¡l'èL'šåŻ¿åČŔïijÑäÿĂåőŽèęĄæŔŘåL'■è¡ñäÿžçAřåžę
OutputArray edges,// è¿ŞåĞžè¿źçijŸåŻ¿åČŔïijÑ␣
 ↪äÿĂèĹňéČ¡æŸŕäžŇåĂijåŻ¿åČŔïijÑèČŇæŹŕæŸŕéżŚèL'š
double threshold1,// ä¡ŐéŸĹåĂijïijÑåÿÿåŔŰéńŸéŸĹåĂijçŽĎ1/2æĹŰèĂĚ1/3
double threshold2,// éńŸéŸĹåĂij
```

(äÿŇéąţçżğçż■)

(çż■äÿŁéąţ)

```
int aptertureSize,// SobleçőÛå■ŘçŽĎsizeïijÑéĂŽåÿÿ3x3ïijÑåŔŰåĂij3
bool L2gradient // éĂL'æŃl'␣
↪trueèąÍçd'žæŸŕL2æİěå¡ŠäÿĂåŇŰïijÑåŘęåĹŹçŤÍL1å¡ŠäÿĂåŇŰïijĹL2æŸŕäžŇèŇČæŢřïijÑL1æŸ
ïijL'
```

åĚşäžŐå¡ŠäÿĂåŇŰïijÑäÿĂèĹňæČĚåĘţäÿŃäÿžäžĘèőąçőÛéĂ§åžęïijÑéĂŽåÿÿéĂL'æŃl'L1å¡ŠäÿĂåŇŰãA

åőŇæŢt'äžčçăĄåőđçŐřåęĆäÿŃïijŽ

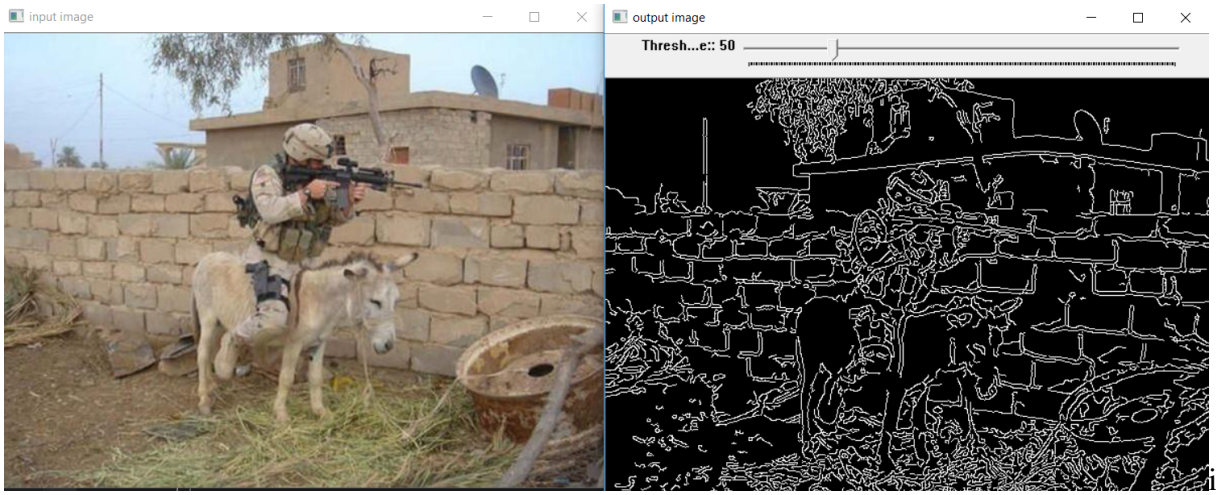
```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace cv;

Mat src, dst, gray_src;

int t1_value = 50;
int max_value = 255;

void Canny_Demo(int, void*);

int main(int argc, char** argv) {
        src = imread("D:/1.jpg");
        if (!src.data) {
                printf("could not load image...\n");
                return -1;
        }
        namedWindow("input image", CV_WINDOW_AUTOSIZE);
        namedWindow("output image", CV_WINDOW_AUTOSIZE);
        imshow("input image", src);

        cvtColor(src, gray_src, CV_BGR2GRAY);
        createTrackbar("Threshold Value:", "output image", &t1_
↪value, max_value, Canny_Demo);  //
↪åĹŽåżžäÿĂäÿłæŃŰåŁĺæİąïijÑèğęåŔŚæŃŰåŁĺæİąçŽĎåŻđèřČåĞ¡æŢřäÿžCanny_
↪Demo
        Canny_Demo(0, 0);

        waitKey(0);
        return 0;
}

void Canny_Demo(int, void*) {
        Mat edge_output;
        blur(gray_src, gray_src, Size(3, 3), Point(-1, -1), BORDER_
↪DEFAULT);
        Canny(gray_src, edge_output, t1_value, t1_value * 2, 3,␣
↪false);
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
        /*␣
↪æşÍéĞŁæŐL'éČÍåĹĘæŸŕçŤÍå¡l'èL'šåŻ¿åČŔæŸ¿çd'žcannyçőŮå■ŘïijŇåęĆæđIJäÿ■åŁăçŽĎèŕÍå
        dst.create(src.size(), src.type);
        src.copyTo(dst, edge_output);
        imshow("output image", dst);
        */

        imshow("output image", edge_output);
}
```

è£Źæăůåď'ĎçŘĘçŽĎåŻ¿çL'ĞæIJĂåŘŐæŸŕéżŠåžŢïijŇçŹ¡èL'šçŽĎè¿źïijŽ



image

åęĆæđIJç£żè¡ňè£ĞæÌěïijŇæŤźæĹŘçŹ¡è¿źéżŠåžŢåŘŕèČ¡çIJŃèţůæÌěæŢĹæđIJäijŽæŻť'åě¡ïijŇæĹŚäżňåŔ

```
imshow("output image", ~edge_output);  //~
↪èąÍçd'žåŔŰåŔ■ïijŇåČŘçt'ăåŔŰåŔ■åřśåŔŕäżěåŔŸæĹŘçŹ¡åžŢéżŚè¿źäžĘ
```

æIJĂåŘŐèŕť'äÿĂäÿŃïijŇå¡śåŞ■CannyçőŮæşŢçŽĎäÿżèęĄæĹŘåČŘåŻăçť'ăæŸŕä¡ŐéŸÝĹåĂijåŠŇéńŸéŸÝĹ

CHAPTER 6

æÍąæİ£åŇźéĚ■

6.1 æÍąæİ£åŇźéĚ■äżŃçż■

- æÍąæİ£åŇźéĚ■åřśæŸŕåIJÍæŢť'äÿłåŻ¿åČŔåŇžå§§åŔŚçŐřäÿŐçżŹåőŽå■ŘåŻ¿åČŔåŇźéĚ■çŽĎåřŔåİŮåŇ
- æĽĂäżěæÍąæİ£åŇźéĚ■éęŰåĚĹéIJĂèęĄäÿĂäÿłæÍąæİ£åŻ¿åČŔTïijĹçżŹåőŽçŽĎå■ŘåŻ¿åČŔïijĽ'
- åŔęåd'ŰéIJĂèęĄäÿĂäÿłå¿ĚæčĂæţŃçŽĎåŻ¿åČŔ-æžŘåŻ¿åČŔS
- åůěä¡IJæŰźæşŢïijŇåIJÍåÿęæčĂæţŃåŻ¿åČŔäÿŁïijŇäżŐåůęåĹřåŔşïijŇäżŐäÿŁåŘŚäÿŃèőąçŐŮæÍąæİ£åŻ

6.2 æÍąæİ£åŇźéĚ■çŤÍåĹřçŽĎçőŮæşŢ

OpenCVäÿ■åŇĚåŘńäžĘåĚ■çğ■æÍąæİ£åŇźéĚ■çŽĎçőŮæşŢïijŽ

6.3 APIäżŃçż■

```
matchTemplate(

InputArray image,// æžŘåŻ¿åČŔïijŇå£ĚéążæŸŕ8-bitæĹŰèĂĚ32-
 ↪bitæţőçĆźæŢřåŻ¿åČŔ

InputArray templ,// æÍąæİ£åŻ¿åČŔïijŇçśżåđŃäÿŐè¿ŞåĚěåŻ¿åČŔäÿĂèĞt'

OutputArray result,//␣
 ↪è¿ŞåĞžçżŞæđIJïijŇå£ĚéążæŸŕå■ŢéĂŽéAŞ32ä¡■æţőçĆźæŢřïijŇåĄĞèő¿æžŘåŻ¿åČŔWxH,
 ↪æÍąæİ£åŻ¿åČŔwxh,
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
                    åĹŹçżŞæđIJå£Ěéążäÿžw-w+1, H-
↪h+1çŽĎåd'ğåřŔãĂĆ(wæŸŕåő¡ïijŇhæŸŕéńŸ)
int method,//
↪ä¡£çŤĺçŽĎåŇźéĚ■æŰźæşŢïijŇäÿĂèĹňæŐĺè■Řä¡£çŤĺå¡ŠäÿĂåŇŰçŽĎæŰźæşŢ

InputArray mask=noArray()//(optional)
)
```

```
enum cv::TemplateMatchModes {
        cv::TM_SQDIFF = 0,
        cv::TM_SQDIFF_NORMED = 1,
        cv::TM_CCORR = 2,
        cv::TM_CCORR_NORMED = 3,
        cv::TM_CCOEFF = 4,
        cv::TM_CCOEFF_NORMED = 5
      }
```

image

6.4 äżčçăĄæijŤçď'ž

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace cv;

Mat src, temp, dst;

int match_method = CV_TM_SQDIFF;
int max_track = 5;

void Match_Demo(int, void*);

int main(int argc, char** argv) {
        src = imread("D:/temp/6.bmp");
        temp = imread("D:/temp/temp.png");
        if (!src.data || !temp.data) {
                printf("could not load image...\n");
                return -1;
        }
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
        namedWindow("input image", CV_WINDOW_NORMAL);
        namedWindow("output image", CV_WINDOW_NORMAL);
        namedWindow("template match-demo", CV_WINDOW_NORMAL);
        imshow("input image", src);
        const char* trackbar_title = "Match Algo Type";
        createTrackbar(trackbar_title, "output image", &match_
↪method, max_track, Match_Demo);
        Match_Demo(0, 0);  //
↪åĚĹèřČçŤĺäÿĂäÿŃïijÑä£ÌèŕĄåĹÌåğŃåĂijäÿ■äÿžçl'ž

        waitKey(0);
        return 0;
}

void Match_Demo(int, void*)
{
        int width = src.cols - temp.cols + 1;
        int height = src.rows - temp.rows + 1;
        Mat result(width, height, CV_32FC1);  //
↪å£Ěéążæŷŕ32ä¡■æţőçĆźæŢř

        matchTemplate(src, temp, result, match_method, Mat()); //
↪åĹřæŮűåĂŹäžŐtrackbaräÿŁéÌćèŐůåŔŰmatch_method

        //äÿŃéÌćæŸŕå¡ŠäÿĂåŇŰïijÑæŁŁçżŞæđIJåŔŸæĹŘ0åĹř1äźŃéŮt'
        normalize(result, result, 0, 1, NORM_MINMAX, -1, Mat());

        //
↪äÿŃéÌćèęĄæL'¿åĞžæÍąæÌ£åŇźéĚ■æIJĂåd'ğåĂijæIJĂåřŕåĂijçŽĎä¡■ç¡őïijÑäźSåřśæŸŕåŠŇåŞ
        Point minLoc; // æL'¿åĞžæIJĂåřŕåĂijçŽĎä¡■ç¡ő
        Point maxLoc; // æL'¿åĞžæIJĂåd'ğåĂijçŽĎä¡■ç¡ő
        double min, max;
        //OpenCVæŕŘä¿ŻAPIæÌěæL'¿åĞžæIJĂåd'ğæIJĂåřŕåĂijçŽĎä¡■ç¡ő
        minMaxLoc(result, &min, &max, &minLoc, &maxLoc, Mat());

        //çŤĺçŞl'å¡ćæąĘæÌěæŁŁæIJĂåd'ğæIJĂåřŕåĂijæÌěæăĞåĞžæÌě
        src.copyTo(dst); //åIJldstäÿŁéÌćè£ŻèąŇçżŸåĹűåůěä¡IJ
        Point temLoc;
        if (match_method == CV_TM_SQDIFF ||  match_method == CV_TM_
↪SQDIFF_NORMED)  //
↪åŕźäžŐè£Źäÿdʼçğ■æŰźæşŢèĂŇèÍĂïijÑåžŤèŕěæŸŕæIJĂåřŕåĂijæÌěåŇźéĚ■ãĂĆåĚűäżŰæŰźæşŢæŸ
        {
                temLoc = minLoc;
        }
        else {
                temLoc = maxLoc;
        }
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

```
        rectangle(dst, Rect(temLoc.x, temLoc.y, temp.cols, temp.
↪rows), Scalar(0, 0, 255), 2, 8);  //
↪åIJĺæIJĂçżĹè¿ŞåĞžåŻ¿åČŔäÿŁçżŸåĹűäÿĂäÿłçŞl'å¡ć
        rectangle(result, Rect(temLoc.x, temLoc.y, temp.cols, temp.
↪rows), Scalar(0, 0, 255), 2, 8); //
↪åIJĺresultçżŞæđIJäÿŁéİćè¿ŞåĞžäÿĂäÿłçŞl'å¡ć

        imshow("output image", result);
        imshow("template match-demo", dst);
}
```

CHAPTER 7

æŢřæ■őäżŃçż■

æŢřæ■őåIJřåİĂïijŽhttps://github.com/hromi/SMILEsmileD

æŢřæ■őåŇĚåŘń13165åijăçĄřåžęåŻ¿çL’ĞïijŇæŕŔåijăåŻ¿çL’ĞçŽĎåřžåŕÿæŸŕ64*64ãĆč£ŹäÿłæŢřæ■őéŽ

CHAPTER 8

æŢřæ■őéćĎåďĎçŘĘ

éęŰåĚĹåŕijåĚěçŻÿåžŤçŽĎåŇĚïijŽ

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from keras.preprocessing.image import img_to_array
from keras.utils import np_utils
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import imutils
import cv2
import os

from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
```

```
dataset_dir = os.path.abspath(r"./SMILEs/") #smileæŢřæ■őéŻĘèůŕå¿Ď
model_dir = os.path.abspath(r"./model/lenet.hdf5")
 ↪#èő■çżČæĺąåđŇä£Ìå■Ÿèůŕå¿Ď

data = []
labels = []
```

```
for imagePath in sorted(list(paths.list_images(dataset_dir))):
    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)    #␣
↪è¡ňæ■ćæĹŘçĄřåžęåŻ¿åČŕ
    image = imutils.resize(image, width = 28)
↪#åřĘåŻ¿åČŕåřžåŕÿæŤźæĹŘ28*28
    image = img_to_array(image)    #ä¡£çŤÍKerasçŽĎimg_to_
↪arrayè¡ňæ■ćæĽŘætőçĆźåđŃåŠŇïijĹ28*28*1ïijL'ïijŇä¿£äžŐæŐëäÿŃæİëçěđçżŕç¡ŚçżIJå■ęä
    data.append(image)

    label = imagePath.split(os.path.sep)[-3]
    label = "smiling" if label == "positives" else "not_smiling"
↪#åęĆæđIJlabelå■ŮçňęäÿšéĞŇéİćæIJL'positiveåŕšéĞ■åŚ¡åŘ■äÿžsmiling
    labels.append(label)
```

```
# åřĘdataåŠŇlabelséČ¡è¡ňæ■ćäÿžnumpyçśżåđŃ
data = np.array(data, dtype= "float") / 255.0 #åřĘåČŕçt'äè¡ňæ■ćåĹř[0,
↪ 1]èŇČåżt'äźŃåĘĚ
labels = np.array(labels)

# åŕźlabelè£ŻèąŇone-hotçijŰçăĄ
le = LabelEncoder().fit(labels)    #␣
↪LabelEncoderåŔŕäżěåřĘæăĞç■¿åĹĘéĚ■äÿĂäÿł0âĂŤn_classes-
↪1äźŃéŮt'çŽĎçijŰçăĄ

# transformçŤĺæİěæăĞåĞĘåŇŰïijŇåřĘlabelsäÿ■'not_smiling
↪'åŠŇâĂŸsmilingâĂŹçŽĎæŢřæ■őè¡ňæ■ćæĹŘ0åŠŇ1çŽĎå¡ćåijŕ
labels = np_utils.to_categorical(le.transform(labels), 2)   #␣
↪2æŸŕnum_classèąĺçd'žè¿ŞåĞžçŽĎæŸŕ2åĹŮæŢřæ■ółçŽĎæĎŔæĂİ
```

äÿŃéİćéIJĂèęĄèğčåĘşäÿĂäÿŃæăůæIJňäÿ■åźşèąąéŮőéćŸãĂĆ

æŢřæ■őéŻĘéĞŇéİćæIJL'9475äÿłçňŚèĎÿæăůæIJňïijŇåŠŇ3690äÿłéİđçňŚèĎÿæăůæIJňãĂĆäÿŃéİćçŽĎäž
hotçijŰçăĄïijŇæL'Ăäžěçż§æđIJæŸŕ[9475, 3690] æĹŚäžňèęĄèğčåĘşæŢřæ■őäÿ■åźşèąąéŮőéćŸåŔŕäžěä¡£çŤĺ

```
classTotals = labels.sum(axis=0)
classWeight = classTotals.max() / classTotals
```

stratifyæŸŕäÿžäžĘä£ÍæŇĄsplitåL'■çśżçŽĎåĹĘåÿČãĂĆæŕŤåęĆæIJL'100äÿłæŢřæ■őïijŇ80äÿłåśđäžŐAçś
test_size=0.25, stratify = y_all), éĆčäźĹsplitäźŃåŘŐæŢřæ■őåęĆäÿŃïijŽ

training: 75äÿłæŢřæ■őïijŇåĚűäÿ■60äÿłåśđäžŐAçśżïijŇ15äÿłåśđäžŐBçśżãĂĆ

testing: 25äÿłæŢřæ■őïijŇåĚűäÿ■20äÿłåśđäžŐAçśżïijŇ5äÿłåśđäžŐBçśżãĂĆ

çŤĺäžĘstratifyåŔĆæŢřïijŇtrainingéŻĘåŠŇtestingéŻĘçŽĎçśżçŽĎæŕŤä¿ŃæŸŕ AïijŽB=
4ïijŽ1ïijŇç■L'åŘŇäžŐsplitåL'■çŽĎæŕŤä¿ŃïijĹ80ïijŽ20ïijL'ãĂĆéĂŽåÿÿåIJlè£Źçğ■çśżåĹĘåÿČäÿ■åźşèąąçŽĎ

```
(trainX, testX, trainY, testY) = train_test_split(data, labels,␣
↪test_size = 0.20,
                                                  stratify = labels,␣
↪random_state = 42)
```

(äÿŃéąţçżğçż■)

(çż■äÿŁéąţ)

CHAPTER 9

ä¡£çŤÍLeNetåőđçŐřçňŚèĎÿæčĂæţŃåĹĘçśż

äÿŃéİćæŸŕæÍąåđŃåőđçŐřéČÍåĹĘïijŽ

```
model = Sequential()

# first set of CONV => RELU => POOL layers
model.add(Conv2D(input_shape=(28, 28, 1), kernel_size=(5, 5),␣
↪filters=20, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='same'))

# second set of CONV => RELU => POOL layers
model.add(Conv2D(kernel_size=(5, 5), filters=50,  activation='relu',
↪ padding='same'))
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='same'))

# first (and only) set of FC => RELU layers
model.add(Flatten())
model.add(Dense(500, activation='relu'))

model.add(Dense(2, activation='softmax'))
```

```
model.compile(loss = "binary_crossentropy", optimizer = "adam",␣
↪metrics = ["accuracy"])

H = model.fit(trainX, trainY, validation_data = (testX, testY),
              class_weight = classWeight, batch_size = 64, epochs =␣
↪15, verbose = 1)  #verbose = 1æŸ¿çd'žè£Žåžęæİą
```

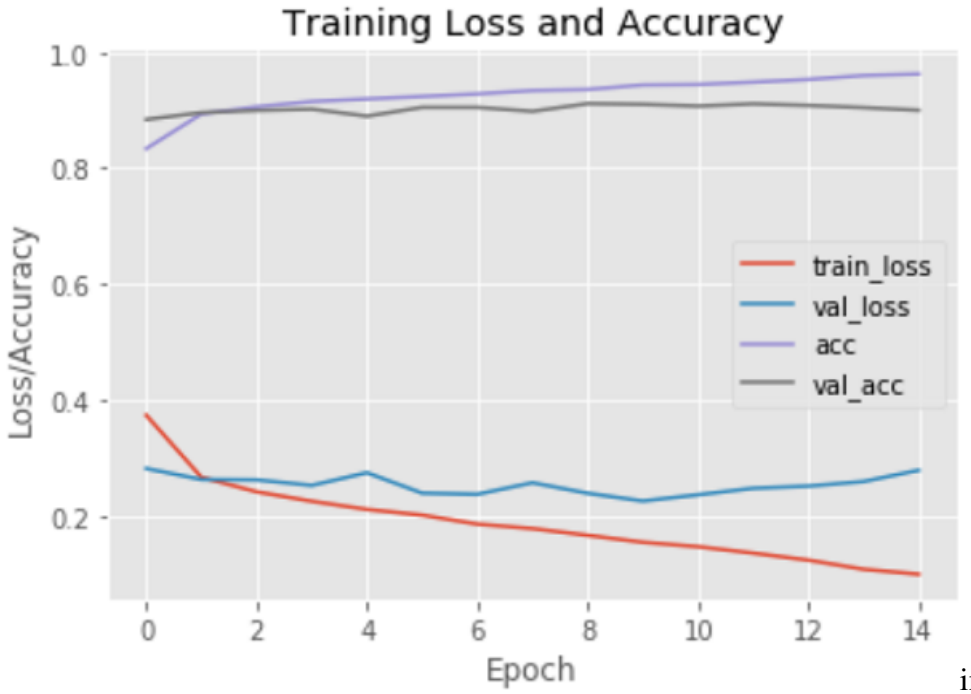
kerasæšąæIJL'çŻt'æŐěåŘŕäżěçż§èőąrecallåŠŇf1åĂijçŽĎåŁđæşŢãĂĆåŘŕäżěçŤĺsklearnãĂĆ
ä¡ĘæŸŕsklearnæšąæIJL'åŁđæşŢçŻt'æŐěåd'ĎçŘĘKerasçŽĎæŢřæ■őïijŇæL'Ăäżěèęąçżŕè£ĞäÿĂäžŽåd'ĎçŘĘ
1æăĞç■¿çŽĎåĹŠåĹĘäžĘãĂĆ

```
predictions = model.predict(testX, batch_size = 64)


print(classification_report(testY.argmax(axis = 1), predictions.
↪argmax(axis = 1),
                            target_names = le.classes_))   # le.
↪classesæŸŕ['not_smiling', 'smiling']çżĎæĹŘçŽĎæŢřçżĎ

model.save(model_dir)
```

è¿ŞåĞžçżŞæđIJïijŽ

```
              precision    recall  f1-score   support

not_smiling       0.95      0.91      0.93      1895
    smiling       0.79      0.87      0.83       738

avg / total       0.90      0.90      0.90      2633
```

```
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, 15), H.history["loss"], label = "train_loss")
plt.plot(np.arange(0, 15), H.history["val_loss"], label = "val_loss
↪")
plt.plot(np.arange(0, 15), H.history["acc"], label = "acc")
plt.plot(np.arange(0, 15), H.history["val_acc"], label = "val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch")le
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.show()
```


Training Loss and Accuracy
Loss/Accuracy
Epoch
train_loss
val_loss
acc
val_acc
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
10
12
14

image

CHAPTER 10

äžžèĎÿæčĂæţŃåðđçŐř

è£ŹéĞŇä¡£çŤĺOpenCVçŽĎHaarçL'źå¿ĄåŠŇçžğèĄŤåĹĘçśżåŹĺæİěåðđçŐřåðđæŮűäžžèĎÿæčĂæţŃïijŇå

æĹŚäžňåIJĺäžčçăĄäÿ■ä¡£çŤĺäžĘOpenCVè£ŹäÿłåůěåĚůæİěåĚůä¡ŞåðđçŐřïijŇåIJĺOpenCVäÿ■ïijŇçŻÿåžˇ
3.4\opencv\sources\data\haarcascadesèůřå¿ĎäÿŃåŘřäžěæL'¿åĹřãĂĆäÿŃéİćæŸřåĚůä¡Şäžčçă ĄåðđçŐřïijŽ

```
from keras.preprocessing.image import img_to_array
from keras.models import load_model
import numpy as np
import imutils
import cv2
import os
import argparse

ap = argparse.ArgumentParser()
ap.add_argument("-c", "--cascade", required= True,
                help= "path to where the face cascade resides")
ap.add_argument("-m", "--model", required= True,
                help= "path to pre-trained smile detector CNN")
ap.add_argument("-v", "--video",
                 help="path to the (optional) video file")
args = vars(ap.parse_args())


detector = cv2.CascadeClassifier(args["cascade"])
 ↪#äžŐåŕźåžŤèůŕå¿ĎäÿåŁăè¡¡äžžèĎÿæčĂæţŃçžğèĄŤåĹĘçśżåŹĺ
model = load_model(args["model"])

#␣
 ↪åŕźæŸŕäżŐçŻÿæIJžäÿ■æčĂæţŃäžžèĎÿè£ŸæŸŕäżŐèğĘécŚäÿ■æčĂæţŃäžžèĎÿåĄŽåĹd'æŰ■
if not args.get("video", False):
```

(äÿŃéąţçžğçż■)

(çż■äÿŁéąţ)

```
    camera = cv2.VideoCapture(0)
else:
    camera = cv2.VideoCapture(args["video"])

while True:

#␣
↪grabbedåŠŇframeæŸŕreadçŽĎäÿd'äÿłè£ŤåŻđåĂijïijŇgrabbedæŸŕåÿČåřŤçśžåđŃçŽĎè£ŤåŻđå
# frameæŸŕæŕŘäÿĂåÿğçŽĎåŻ¿åČŘïijŇæŸŕäÿĂäÿłäÿL'çżt'çŞl'éŸţ
    (grabbed, frame) = camera.read()

    if args.get("video") and not grabbed:
        break


    frame = imutils.resize(frame, width = 300)
↪#æŁŁåŻ¿åČŘåő¡åžęéĞ■æŰřæŇĞåőŽäÿž300åČŘçt'ă
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
↪#åŻăäÿžæĺąåđŃèő■çżČæŸŕåŕźçĄřåžęåŻ¿åČŘåd'ĎçŘĘïijŇæL'Ăäżěè£ŹéĞŇèęĄè¡ñæ■ćæĹŘçĄřåžę
    frameClone = frame.copy()
↪#éĞ■æŰřåĚŃéŽĘframeïijŇçŤĺäžŐæŐěäÿŃæİěçżŸåĹűè¿źçŢŇæąĘ

    rects = detector.detectMultiScale(gray, scaleFactor=1.1,␣
↪minNeighbors=5, minSize=(30, 30),
                                      flags=cv2.CASCADE_SCALE_IMAGE)

    for (fX, fY, fW, fH) in rects:
        roi = gray[fY:fY + fH, fX:fX + fW]
        roi = cv2.resize(roi, (28, 28))
        roi = roi.astype("float") / 255.0
        roi = img_to_array(roi)
        roi = np.expand_dims(roi, axis = 0)

        (notSmiling, smiling) = model.predict(roi)[0]
        label = "Smiling" if smiling > notSmiling else "Not Smiling"

        cv2.putText(frameClone, label, (fX, fY - 10), cv2.FONT_
↪HERSHEY_SIMPLEX,
                    0.45, (0, 0, 255), 2)
        cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH),
                      (0, 0, 255), 2)


    cv2.imshow("Face", frameClone)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

camera.release()
```

(äÿŃéąţçżğçż■)

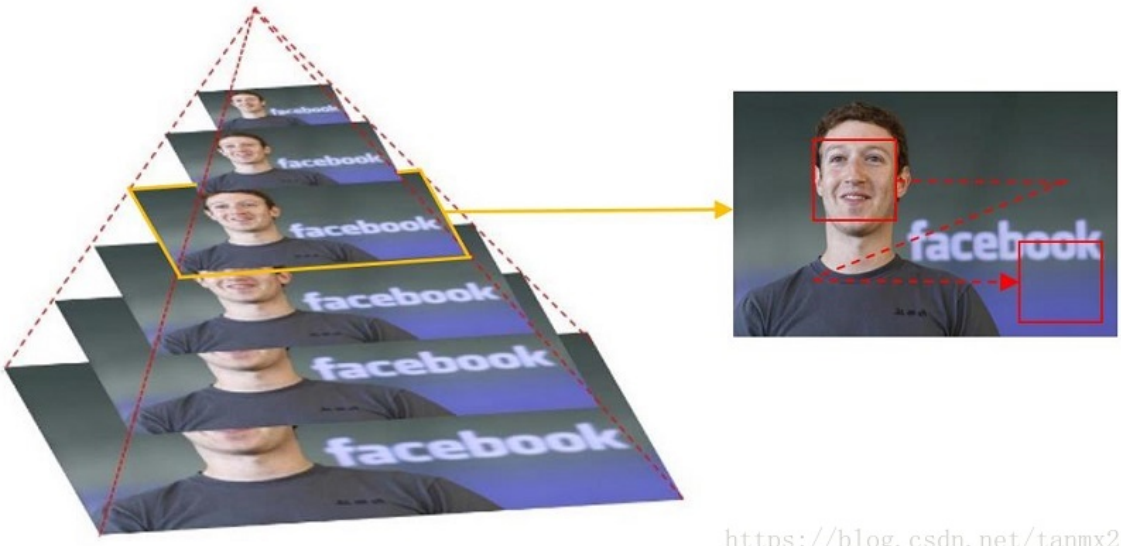
(çż■äÿŁéąţ)

```
cv2.destroyAllWindows()
```

10.1 detector.detectMultiScale

è£ŹéĞŇïijŇåŕźdetector.detectMultiScaleåĄŽäÿĂçĆźèŕťæŸŐïijŽ

äÿžäžĘæčĂæţŃåĹřäÿ■åŘŇåď˙gåřŔçŽĎçŻőæăĞïijŇäÿĂèĹňæIJL'äÿď˙çğ■åĄŽæşŢïijŽéĂŘæ■ěçijľ˙åřŔåŻ
or 1.2ïijL'éĂŘæ■ěçijľ˙åřŔïijŇçĎůåŘŐæčĂæţŃïijŻæŤ¿åď˙gæčĂæţŃçłŮåŔčæŸŕæŁŁæčĂæţŃçłŮåŔčéŢ£åő¡æ



image

çĎůåŘŐïijŇåŕźåžŤæŕŘåijäåŻ¿ïijŇçžğèĄŤåĹĘçśżåŹĺçŽĎåď˙gåřŔåŻžåőŽçŽĎæčĂæţŃçłŮåŔčåŹĺåijĂåğŃ

```
void CascadeClassifier::detectMultiScale( InputArray image,
 Âă Âă Âă Âă Âă Âă Âă Âă Âă Âă ÂăCV_OUT std::vector<Rect>& objects,
 Âă Âă Âă Âă Âă Âă Âă Âă Âă Âă Âădouble scaleFactor,
 Âă Âă Âă Âă Âă Âă Âă Âă Âă Âă Âăint minNeighbors, int flags,
 Âă Âă Âă Âă Âă Âă Âă Âă Âă Âă ÂăSize minSize,
 Âă Âă Âă Âă Âă Âă Âă Âă Âă Âă ÂăSize maxSize )
```

åŔĆæŢř1ïijŽimage–å¿ĚæčĂæţŃåŻ¿çL'ĞïijŇäÿĂèĹňäÿžçĄřåžęåŻ¿åČŔäžěåŁää£ńæčĂæţŃéĂ§åžęïijŻ

åŔĆæŢř2ïijŽobjects–èćńæčĂæţŃçL'ľä¡ŞçŽĎç§ľå¡ćæąĘåŘŚéĞŔçżĎïijŻäÿžè¿ŞåĞžéĞŔïijŇåęĆæ§ŘçL'

åŔĆæŢř3ïijŽscaleFactor–èąÍçď˙žåIJÍåL'■åŘŐäÿď˙æňąçŻÿçżğçŽĎæL'ńæŔŔäÿ■ïijŇæŘIJçt'ćçłŮåŔčçŽĎæ

åŔĆæŢř4ïijŽminNeighbors–èąÍçď˙žæđĎæĹŘæčĂæţŃçŻőæăĞçŽĎçŻÿéĆżç§ľå¡ćçŽĎæIJÅåřŔäÿłæŢř(éż
åęĆæđIJçżĎæĹŘæčĂæţŃçŻőæăĞçŽĎåřŔç§ľå¡ćçŽĎäÿłæŢřåŠŇåřŔäžŐ
min_neighbors - 1 éČ¡äijŽèćńæŐŠéŹď˙ãĂĆ åęĆæđIJmin_neighbors äÿž 0,
åĹŹåĞ¡æŢřäÿ■åĄŽäżżä¡ŢæŞ■ä¡IJåŕśè£ŤåŽđæL'ĂæIJL'çŽĎèćńæčĂåĂŹéĂL'ç§ľå¡ćæąĘïijŇ
è£Źçğ■èő¿åőŽåĂijäÿĂèĹňçŤĺåIJĺçŤĺæĹůèĞłåőŽäźL'åŕźæčĂæţŃçżŞæđIJçŽĎçżĎåŘĹçĺŃåžŔäÿŁïijŻ

åŔĆæŢř5ïijŽflags=0ïijŽåŔŕäžěåŔŰåęĆäÿŃè£ŹäžŻåĂijïijŽ CASCADE_DO_CANNY_PRUNING=1, åĹľçŤĺcannyè¿źçijŸæčĂæţŃæİěæŐŠéŹď'äÿĂäžŻè¿źçijŸå¿ĹåřŚæĹŰè CASCADE_SCALE_IMAGE=2, æ■čåÿÿæŕŤä¿ŃæčĂæţŃ CASCADE_FIND_BIGGEST_OBJECT=4, åŔłæčĂæţŃæIJĂåď'ğçŽĎçL'ľä¡Ş CASCADE_DO_ROUGH_SEARCH=8 åĹİçŢěçŽĎæčĂæţŃ 6. minObjectSize maxObjectSizeïijŽåŇźéĚ■çL'ľä¡ŞçŽĎåď'ğářŔèŇČåŻť'

åŔĆæŢř6ãĂĄ7ïijŽminSizeåŠŇmaxSizeçŤĺæİěéŹŘåĹűå¿ŮåĹřçŽĎçŻőæăĞåŇžå§§çŽĎèŇČåŻť'ãĂĆäź§åĞ¡æŢřäżŃçż■ïijŽ detectMultiscaleåĞ¡æŢřäÿžåď'Žåřžåžęåď'ŽçŻőæăĞæčĂæţŃïijŽ åď'ŽåřžåžęïijŽéĂŽåÿÿæŘIJçt'ćçŻőæăĞçŽĎæĺąæİ£åřžåŕÿåď'ğářŔæŸŕåŻžåőŽçŽĎïijÑä¡ĘæŸŕäÿ■åŘŇåŻ¿çL'Č åď'ŽçŻőæăĞïijŽéĂŽè£ĞæčĂæţŃçňęåŘĹæĺąæİ£åŇźéĚ■åŕźèśąïijŇåŔŕå¿ŮåĹřåď'ŽäÿłçŻőæăĞïijŇåİĞè¿ŞåĞž

minNeighbors=3ïijŽåŇźéĚ■æĹŘåŁ§æL'ĂéIJĂèęĄçŽĎåŚĺåŻť'ç§l'å¡ćæąĘçŽĎæŢřçŻőïijŇæŕŔäÿĂäÿłçL'

åŻăäÿžäżčçăĄäÿ■ä¡£çŤĺäžĘargparseïijŇæL'ĂäžěåŔŕäžěéĂŽè£ĞåŚ¡äžd'æİěæŇĞåőŽïijŇåęĆæđIJæŸŕæČ

```
python detect_smile.py --cascade haarcascade_frontalface_default.xml
                       --model output/lenet.hdf5
```

åęĆæđIJæŸŕçŤĺèğĘécŚçŽĎèŕİïijŇåŔŕäžěè¿ŞåĚěåęĆäÿŃåŚ¡äžd'ïijŽ

```
python detect_smile.py --cascade haarcascade_frontalface_default.xml
                       --model output/lenet.hdf5
                       --video path/to/your/video.mov
```