ДОДАТОК А

Лістинг програми 1

```
// Файл 1.cpp

#include <afxwin.h>
#include <cxcore.h>
#include <cv.h>
#include <cvcam.h>
#include <highgui.h>
#include "1.h"
#include <string.h>
#include <assert.h>
#include <math.h>
#include <float.h>
#include <limits.h>
#include <time.h>
#include <ctype.h>

void mycallback(IplImage *img);
IplImage *image1,*src2,*dst,*dst2,*dst3,*dst4,*gray,*dst5,*dst6,*dst7,*dst8;
int contNum=0;
int harrist=0;
int cannyt=0;
int thresh=0;
int thresh2=0;
CvRect camSpace;
CvMoments moments;
CvPoint centre_of_mass;
CvSeq *contour;
CvMemStorage *storage,*storage2;
bool bCreate=true;
double m=2;
CvPoint2D32f center;
static CvHaarClassifierCascade* cascade = 0;
const char* cascade_name = "haarcascade_frontalface_alt2.xml";
int Haart= 0;
BOOL CApp::InitInstance()
{
        m_pMainWnd=new CMainWin;
        m_pMainWnd->ShowWindow(m_nCmdShow);
        m_pMainWnd->UpdateWindow();
        return TRUE;
```

```
}
BEGIN_MESSAGE_MAP(CMainWin,CFrameWnd)
ON_WM_CLOSE()
END_MESSAGE_MAP()
CApp App;
CMainWin::CMainWin()
{ Create (NULL, "OpenCV");
HWND w= this->GetSafeHwnd();
int ncams=cvcamGetCamerasCount();
if (ncams)
{       bCreate=true;
VidFormat vidFmt={800,600,20.0};
cvcamSetProperty(0,CVCAM_PROP_ENABLE,CVCAMTRUE);
cvcamSetProperty(0,CVCAM_PROP_CALLBACK,mycallback);
cvcamSetProperty(0,CVCAM_PROP_WINDOW,&w);
cvcamSetProperty(0,CVCAM_PROP_SETFORMAT,&vidFmt);
cvNamedWindow(cvGetWindowName(w),CV_WINDOW_AUTOSIZE);
cvNamedWindow( "Canny", 1 );
cvCreateTrackbar("CannyTrack","Canny",&cannyt,200,NULL);
cvResizeWindow("Canny",320,200);
cvNamedWindow( "Contour", 3 );
cvNamedWindow("Sobel",2);
cvResizeWindow("Sobel",320,200);
cvNamedWindow("CornerDetect",9);
cvNamedWindow("Harris",5);
cvResizeWindow("Harris",320,200);
cvCreateTrackbar("HarrisTrack","Harris",&harrist,200,NULL);
cvNamedWindow( "threshold", 7 );
cvCreateTrackbar("threshold","threshold",&thresh,200,NULL);
cvNamedWindow( "Adaptive", 8 );
cvCreateTrackbar("Adaptive","Adaptive",&thresh2,200,NULL);
cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 );
if(!cvcamInit())    MessageBox("Error");
        else    cvcamStart();
}
else MessageBox ("Camera not found");
}
void mycallback(IplImage *src)
{       image1 = src;
        if (bCreate)
        {                       storage=cvCreateMemStorage();
        src2=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,3);
        dst=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,3);
        dst2=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_32F,3);
```

```
        dst3=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,1);
        dst4=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,1);
        dst5=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_32F,1);
        dst6=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_32F,1);
        dst7=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,1);
        dst8=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_32F,1);
        gray=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,1);
                bCreate=false;
        }
        for (int k=0;k<image1->height;k+=600)
        for(int j=(image1->widthStep)*k;j<(image1->widthStep)*(k+1);j+=image1-
>nChannels)
                {       image1->imageData[j]=(char) 255;
                        image1->imageData[j+1]=0;
                        image1->imageData[j+2]=0;
                }
                        cvFlip(src,src2);
                        cvSobel(src2,dst,1,3,7);
                        cvShowImage("Sobel",dst);
                        cvCvtColor(src2,gray,CV_RGB2GRAY);
                        cvCanny(gray,dst3,25,100+cannyt,3);
                        cvShowImage("Canny",dst3);
CvScalar
color=CV_RGB(rand()&255,rand()&255,rand()&255);contNum=cvFindContours(
dst3,storage,&contour,sizeof(CvContour),CV_RETR_CCOMP,CV_CHAIN_APP
ROX_SIMPLE);
                        for(;contour!=0;contour=contour->h_next)
                { cvDrawContours(dst7,contour,color,color,-1,1,8);
cvMoments(contour,&moments,0);
centre_of_mass.x=moments.m10/moments.m00;
centre_of_mass.y=moments.m01/moments.m00;
cvCircle(dst7,centre_of_mass,5,CV_RGB(255,255,255),1,8,0);}
cvShowImage("Contour",dst7);
cvCornerHarris(gray,dst8,3,7,harrist/1000.0);
cvShowImage("Harris",dst8);
static CvScalar colors[] =
   {            {{0,0,255}},
                {{0,128,255}},
                {{0,255,255}},
                {{0,255,0}},
                {{255,128,0}},
                {{255,255,0}},
                {{255,0,0}},
                {{255,0,255}}
```

```
    };
    double scale = 1.3;
    IplImage* gray2 = cvCreateImage( cvSize(src->width,src->height), 8, 1 );
    IplImage* small_img = cvCreateImage( cvSize( cvRound (src->width/scale),
    cvRound (src->height/scale)),8, 1 );    int i;
    cvCvtColor( src, gray2, CV_BGR2GRAY );
    cvResize( gray2, small_img, CV_INTER_LINEAR );
    cvEqualizeHist( small_img, small_img );
    cvClearMemStorage( storage );
    if( cascade )
    {  double t = (double)cvGetTickCount();
     CvSeq* face = cvHaarDetectObjects( small_img, cascade,
storage,1.1,1,0,cvSize(30,30));
     t = (double)cvGetTickCount() - t;
     printf( "detection time = %gms\n", t/((double)cvGetTickFrequency()*1000.) );
       for( i = 0; i < (face ? face->total : 0); i++ )
       {  CvRect* r = (CvRect*)cvGetSeqElem( face, i );
          CvPoint center;
          int radius;
          center.x = cvRound((r->x + r->width*0.5)*scale);
          center.y = cvRound((r->y + r->height*0.5)*scale);
          radius = cvRound((r->width + r->height)*0.5*scale);
          cvCircle( src, center, radius, colors[i%8], 3, 8, 0 );
       }
    }      cvPreCornerDetect(dst3,dst5,5);
           cvShowImage( "CornerDetect", dst5 );
           cvLogPolar( dst, src2, cvPoint2D32f(src->width/2,src->height/2), 40,
CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS+CV_WARP_INVERSE_
MAP );
           cvShowImage( "log-polar", dst );
           cvShowImage( "inverse log-polar", src2 );
           cvWaitKey();
           cvThreshold(src2,dst,2,10+thresh,CV_THRESH_BINARY_INV);
           cvShowImage( "threshold", dst );
cvAdaptiveThreshold(dst3,dst4,5+thresh2,CV_ADAPTIVE_THRESH_MEAN_C,
CV_THRESH_BINARY,3,5);
cvShowImage( "Adaptive", dst4 );
cvShowImage( "result", src );
cvReleaseImage( &gray2 );
cvReleaseImage( &small_img );
cvZero(dst);
cvZero(gray);;
cvZero(dst2);
cvZero(dst3);
```

```
cvZero(dst6);
cvZero(dst7);
cvZero(dst8);
cvZero(dst5);
}
void CMainWin::OnClose()
{ cvcamStop();
cvDestroyAllWindows();
cvcamExit();
if(!bCreate)cvReleaseImage(&dst);
DestroyWindow();
}

// Файл 1.h

#if _MSC_VER > 1000
#pragma once
class CMainWin : public CFrameWnd
{
public:
                CMainWin();
                void OnClose();
                DECLARE_MESSAGE_MAP();
};
class CApp : public CWinApp
{
public:
        BOOL InitInstance();
};
```

ДОДАТОК Б

Лістинг програми 2

```
#include "cv.h"
#include "cxcore.h"
#include "highgui.h"
#include "math.h"
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <sstream>

using namespace std;
double angle( CvPoint* pt1, CvPoint* pt2, CvPoint* pt0 )
{
 double dx1 = pt1->x - pt0->x;
 double dy1 = pt1->y - pt0->y;
 double dx2 = pt2->x - pt0->x;
 double dy2 = pt2->y - pt0->y;
return (dx1*dx2 + dy1*dy2)/sqrt((dx1*dx1 + dy1*dy1)*(dx2*dx2 + dy2*dy2) +
1e-10);}
CvSeq* findSquares4( IplImage* img, CvMemStorage* storage )
{ double s = 0, t = 0;
   CvSeq* contours = NULL;
   CvSeq* result = NULL;
CvSeq* squares = cvCreateSeq( 0, sizeof( CvSeq), sizeof( CvPoint), storage );
cvFindContours( img, storage, &contours, sizeof( CvContour),
CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint( 0, 0) );
  while( contours )
    { result = cvApproxPoly( contours, sizeof( CvContour), storage,
CV_POLY_APPROX_DP, cvContourPerimeter( contours)*0.02, 0 );
  if( result->total == 4 && fabs( cvContourArea( result,
CV_WHOLE_SEQ)) > 1000 && fabs( cvContourArea( result,
CV_WHOLE_SEQ)) <( img->height * img->width/2 ) &&
cvCheckContourConvexity( result) )
      { s = 0;
        for( int i = 0; i < 5; i++ )
        { if( i >= 2 ) { t = fabs( angle( (
CvPoint*)cvGetSeqElem( result, i ), ( CvPoint*)cvGetSeqElem( result, i-2 ), (
CvPoint*)cvGetSeqElem( result, i-1 )));
  s = s > t ? s : t; } }
```

```
if( s < 0.5 )for( int i = 0; i < 4; i++ )cvSeqPush( squares,
(CvPoint*)cvGetSeqElem( result, i )); }
contours = contours->h_next; }
cout<<">>> rectangles: "<< squares->total/4 <<endl;
return squares;}
void drawSquares(IplImage *img, CvSeq* squares )
{ CvFont font;
cvInitFont( &font, CV_FONT_HERSHEY_SIMPLEX, 0.4f, 0.4f, 0,1, 8 );
int i,j = 0;
CvSeqReader reader;
cvStartReadSeq( squares, &reader, 0 );
for( i = 0; i < squares->total; i += 4 )
        {       j++;
                CvPoint pt[4], *rect = pt;
                int count = 4;
                // read 4 vertices
                CV_READ_SEQ_ELEM( pt[0], reader );
                CV_READ_SEQ_ELEM( pt[1], reader );
                CV_READ_SEQ_ELEM( pt[2], reader );
                CV_READ_SEQ_ELEM( pt[3], reader );
                cvCircle( img, pt[1], 2, CV_RGB(200,0,0),1, 8, 0 );
                cvCircle( img, pt[2], 4, CV_RGB(200,0,0),1, 8, 0 );
                cvLine( img, pt[1], pt[2], CV_RGB(255,255,255),1, 8, 0 )
double angle = abs(pt[1].y-pt[2].y)/sqrt((pt[1].x-pt[2].x)*(pt[1].x-pt[2].x)+(pt[1].y-
pt[2].y)*(pt[1].y-pt[2].y)+0.00001);   char st[255];   sprintf(st, "%2f", angle);
cvLine( img, cvPoint(0,img->height/2), cvPoint(img->width,img->height/2),
CV_RGB(200,200,200),1, 8, 0 );
cvLine( img, cvPoint(img->width/3,0), cvPoint(img->width/3,img->height),
CV_RGB(200,200,200),1, 8, 0 );
cvLine( img, cvPoint(img->width/3*2,0), cvPoint(img->width/3*2,img->height),
CV_RGB(200,200,200),1, 8, 0 );
cvPutText( img, st, pt[1], &font, CV_RGB(200,0,0));
cvPolyLine( img, &rect, &count, 1, 1, CV_RGB(255,255,255), 1, CV_AA, 0 );}}
int main()  {  int c = 0;
CvCapture* capture = cvCaptureFromCAM(0);
f(!cvQueryFrame(capture)){cout<<"Video capture failed, please check the
camera."<<endl;}else{cout<<"Video camera capture status: OK"<<endl;};
        CvSize sz = cvGetSize(cvQueryFrame(capture));
        cvNamedWindow( "out", 0);
        cvNamedWindow( "src", 1);
        IplImage* src = cvCreateImage( sz, 8, 3 );
        IplImage* gray = cvCreateImage( sz, 8, 1 );
        IplImage* pyr = cvCreateImage( cvSize(sz.width/2, sz.height/2), 8, 3 );
        IplImage* tgray = cvCreateImage( sz, 8, 1 );
```

```
        CvMemStorage* storage = cvCreateMemStorage(0);
        CvSeq* contours = NULL;
        CvMemStorage* mainStorage = cvCreateMemStorage(0);
        CvMemStorage* cstorage = cvCreateMemStorage( 0);
        CvSeq* circles = NULL;
        while(c != 27)    {          IplImage* img = NULL;
                                     IplImage* out = NULL;
        src = cvQueryFrame( capture);
        img = cvCloneImage( src); out = cvCloneImage( src);
        cvPyrDown( img, pyr, 7 ); cvPyrUp( pyr, img, 7 );
        cvSetImageCOI( img, 2 );
        cvCopy( img, tgray, 0 );
        cvThreshold( tgray, gray, 100, 255, CV_THRESH_BINARY );
        cvShowImage( "src", gray);
        cvFindContours( gray, storage, &contours, sizeof(CvContour),
CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0) );
        circles =  cvHoughCircles( gray, cstorage, CV_HOUGH_GRADIENT, 1,
gray->height/16, 8, 10, 4, 50 );
        cout <<"Total circles: "<< circles->total <<endl;
        for(  int i  =  0; i < circles-> total; i++ )
                {       float* p  =  ( float*)cvGetSeqElem(  circles,  i );
        cvCircle(  out,  cvPoint( cvRound( p[0]), cvRound( p[1])),  2,  CV_RGB(
200, 0, 0),  -1,  8,  0 );
        cvCircle(  out,  cvPoint( cvRound( p[0]), cvRound( p[1])),  cvRound( p[2]),
CV_RGB( 200, 0, 0),  1,  8,  0 );        }
        drawSquares(out,findSquares4(gray,mainStorage));
        cvReleaseImage( &img);
        cvShowImage( "out", out);
        cvReleaseImage( &out);
         c = cvWaitKey(10);    }
           cvReleaseCapture( &capture );
           cvDestroyAllWindows();}
```

ДОДАТОК В

Демонстраційний матеріал

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮИК. 466.459.020ПЗ	Пояснювальна записка	104	с.
			Інші документи		
А4	2		Лістинг програми 1	5	с.
А4	3		Лістинг програми 2	3	с.
А4	4		Демонстраційний матеріал	16	*у форматі *.ppt*

					ГЮИК. 466.459.020ВД			
Змін	А	№ докум.	Підпи	Дата				
Розроб.		Гасанов Р.М.			Розробка системи автоматизації для ідентифікації виробів за допомогою машинного зору Відомість кваліфікаційної роботи	Літера	Аркуш	Аркушів
Переві		Аллахверанов Р.Ю.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.конт.		Стародубцев М.Г.						
Затв.		Невлюдов І.Ш.						