#include "stdafx.h"

#ifdef \_CH\_

#pragma package <opencv>

#endif

#define CV\_NO\_BACKWARD\_COMPATIBILITY

#ifndef \_EiC

#include <cv.h>

#include <highgui.h>

#include <stdio.h>

#include <ctype.h>

#using <System.dll>

using namespace System;

using namespace System::IO::Ports;

using namespace System::Threading;

#endif

// Declaration

IplImage \*image = 0, \*hsv = 0, \*hue = 0, \*mask = 0, \*backproject = 0, \*histimg = 0;

CvHistogram \*hist = 0;

// Datatype int backproject\_mode

int backproject\_mode = 0;

int select\_object = 0;

int track\_object = 0;

int show\_hist = 1;

CvPoint origin;

CvRect selection;

CvRect track\_window;

CvBox2D track\_box;

CvConnectedComp track\_comp;

int hdims = 16;

float hranges\_arr[] = {0,180};

float\* hranges = hranges\_arr;

int vmin = 10, vmax = 256, smin = 30;

void on\_mouse( int event, int x, int y, int flags, void\* param )

{

if( !image )

return;

if( image->origin )

y = image->height - y;

if( select\_object )

{

selection.x = MIN(x,origin.x);

selection.y = MIN(y,origin.y);

selection.width = selection.x + CV\_IABS(x - origin.x);

selection.height = selection.y + CV\_IABS(y - origin.y);

selection.x = MAX( selection.x, 0 );

selection.y = MAX( selection.y, 0 );

selection.width = MIN( selection.width, image->width );

selection.height = MIN( selection.height, image->height );

selection.width -= selection.x;

selection.height -= selection.y;

}

switch( event )

{

case CV\_EVENT\_LBUTTONDOWN:

origin = cvPoint(x,y);

selection = cvRect(x,y,0,0);

select\_object = 1;

break;

case CV\_EVENT\_LBUTTONUP:

select\_object = 0;

if( selection.width > 0 && selection.height > 0 )

track\_object = -1;

break;

}

}

// Function to convert hsv..color format to rgb..color format

CvScalar hsv2rgb( float hue )

{

int rgb[3], p, sector;

static const int sector\_data[][3]=

{{0,2,1}, {1,2,0}, {1,0,2}, {2,0,1}, {2,1,0}, {0,1,2}};

hue \*= 0.033333333333333333333333333333333f;

sector = cvFloor(hue);

p = cvRound(255\*(hue - sector));

p ^= sector & 1 ? 255 : 0;

rgb[sector\_data[sector][0]] = 255;

rgb[sector\_data[sector][1]] = 0;

rgb[sector\_data[sector][2]] = p;

return cvScalar(rgb[2], rgb[1], rgb[0],0);

}

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

{

CvCapture\* capture = 0;

int i=0;

//declaring 'data' to write serial port

array< unsigned char, 1> ^up;

array< unsigned char, 1> ^down;

array< unsigned char, 1> ^left;

array< unsigned char, 1> ^right;

//defining variable 'data' == b

up = gcnew array<unsigned char, 1>{'I'};

down = gcnew array<unsigned char, 1>{'D'};

left = gcnew array<unsigned char, 1>{'L'};

right = gcnew array<unsigned char, 1>{'R'};

//Creating the new serial port COM3 of computer

SerialPort^ mySerialPort = gcnew SerialPort("COM3");

//Setting up the properties of the Serial port

mySerialPort->BaudRate = 19200;

mySerialPort->Parity = Parity::None;

mySerialPort->StopBits = StopBits::One;

mySerialPort->DataBits = 8;

mySerialPort->Handshake = Handshake::None;

//To Open the Serial Port

mySerialPort->Open(); //'->' OR '.'

//WriteLine is inbuilt function of Class 'Console' to Write Line on the Console

//Console::WriteLine("Writing to COM3 \n");

//It sends the data in 'data' 5 times

/\*while(i<5)

{

i++;

//mySerialPort->Write(data,0,1);

mySerialPort->Write(data,0,1);

}\*/

//Closing the Serial Port

if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))

capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );

else if( argc == 2 )

capture = cvCaptureFromAVI( argv[1] );

if( !capture )

{

fprintf(stderr,"Could not initialize capturing...\n");

return -1;

}

printf( "Hot keys: \n"

"\tESC - quit the program\n"

"\tc - stop the tracking\n"

"\tb - switch to/from backprojection view\n"

"\th - show/hide object histogram\n"

"To initialize tracking, select the object with mouse\n" );

cvNamedWindow( "Histogram", 1 );

cvNamedWindow( "CamShift", 1 );

cvSetMouseCallback( "CamShift", on\_mouse, 0 );

cvCreateTrackbar( "Vmin", "CamShift", &vmin, 256, 0 );

cvCreateTrackbar( "Vmax", "CamShift", &vmax, 256, 0 );

cvCreateTrackbar( "Smin", "CamShift", &smin, 256, 0 );

for(;;)

{

IplImage\* frame = 0;

int i, bin\_w, c;

frame = cvQueryFrame( capture );

if( !frame )

break;

if( !image )

{

/\* allocate all the buffers \*/

image = cvCreateImage( cvGetSize(frame), 8, 3 );

image->origin = frame->origin;

hsv = cvCreateImage( cvGetSize(frame), 8, 3 );

hue = cvCreateImage( cvGetSize(frame), 8, 1 );

mask = cvCreateImage( cvGetSize(frame), 8, 1 );

backproject = cvCreateImage( cvGetSize(frame), 8, 1 );

hist = cvCreateHist( 1, &hdims, CV\_HIST\_ARRAY, &hranges, 1 );

histimg = cvCreateImage( cvSize(320,200), 8, 3 );

cvZero( histimg );

}

cvCopy( frame, image, 0 );

cvCvtColor( image, hsv, CV\_BGR2HSV );

if( track\_object )

{

int \_vmin = vmin, \_vmax = vmax;

cvInRangeS( hsv, cvScalar(0,smin,MIN(\_vmin,\_vmax),0),

cvScalar(180,256,MAX(\_vmin,\_vmax),0), mask );

cvSplit( hsv, hue, 0, 0, 0 );

if( track\_object < 0 )

{

float max\_val = 0.f;

cvSetImageROI( hue, selection );

cvSetImageROI( mask, selection );

cvCalcHist( &hue, hist, 0, mask );

cvGetMinMaxHistValue( hist, 0, &max\_val, 0, 0 );

cvConvertScale( hist->bins, hist->bins, max\_val ? 255. / max\_val : 0., 0 );

cvResetImageROI( hue );

cvResetImageROI( mask );

track\_window = selection;

track\_object = 1;

cvZero( histimg );

bin\_w = histimg->width / hdims;

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

{

int val = cvRound( cvGetReal1D(hist->bins,i)\*histimg->height/255 );

CvScalar color = hsv2rgb(i\*180.f/hdims);

cvRectangle( histimg, cvPoint(i\*bin\_w,histimg->height),

cvPoint((i+1)\*bin\_w,histimg->height - val),

color, -1, 8, 0 );

}

}

cvCalcBackProject( &hue, backproject, hist );

cvAnd( backproject, mask, backproject, 0 );

cvCamShift( backproject, track\_window,

cvTermCriteria( CV\_TERMCRIT\_EPS | CV\_TERMCRIT\_ITER, 10, 1 ),

&track\_comp, &track\_box );

track\_window = track\_comp.rect;

if( backproject\_mode )

cvCvtColor( backproject, image, CV\_GRAY2BGR );

if( !image->origin )

track\_box.angle = -track\_box.angle;

cvEllipseBox( image, track\_box, CV\_RGB(255,0,0), 3, CV\_AA, 0 );

}

///////////////////////////////////////////////////////////////////////////

CvPoint2D32f centroid=track\_box.center;

printf("%f %f \n",centroid.x,centroid.y);

float x=centroid.x,y=centroid.y;

int imh=image->width,imw=image->height;

int orginx=imh/2,orginy=imw/2;

//printf("height = %d width = %d \n",image->height,image->width);

if (x<(orginx - 40))

{

printf("Move Left\n");

mySerialPort->Write(left,0,1);

//left ="L" send to serial port

}

else if (x>(orginx + 40))

{

printf("Move Right\n");

mySerialPort->Write(right,0,1);

//right = "R" send to serial port

}

if (y<(orginy - 40))

{

printf("Move Up\n %d",orginy-60);

mySerialPort->Write(up,0,1);

//up = "I" send to serial port

}

else if (y>(orginy + 40))

{

printf("Move Down\n");

mySerialPort->Write(down,0,1);

//Down = "D" send to serial port

}

/\*//////////////////////////////////////////////////////////////

'I'

(Camera Up)

'L' 'R'

(Camera left) 'C' (Camera right)

'D'

(Camera Down)

THIS CHARACTERS WILL BE SEND TO 8051

//////////////////////////////////////////////////////////////\*/

if( select\_object && selection.width > 0 && selection.height > 0 )

{

cvSetImageROI( image, selection );

cvXorS( image, cvScalarAll(255), image, 0 );

cvResetImageROI( image );

}

cvShowImage( "CamShift", image );

cvShowImage( "Histogram", histimg );

c = cvWaitKey(10);

if( (char) c == 27 )

break;

switch( (char) c )

{

case 'b':

backproject\_mode ^= 1;

break;

case 'c':

track\_object = 0;

cvZero( histimg );

break;

case 'h':

show\_hist ^= 1;

if( !show\_hist )

cvDestroyWindow( "Histogram" );

else

cvNamedWindow( "Histogram", 1 );

break;

default:

;

}

}

cvReleaseCapture( &capture );

cvDestroyWindow("CamShiftDemo");

//Serial Port Close CLASS followed by "->" to open properties and methods.

mySerialPort->Close();

return 0;

}