/\*

straight arrow detect in airplane door

author: Cheng Yupeng (Neil)

version: 3rd

describe: detect the big banded arrow, and mark the five points which build up the arrow

\*/

#include<opencv2/core/core.hpp>

#include<opencv2/highgui/highgui.hpp>

#include<opencv2/imgproc/imgproc.hpp>

#include<opencv2/ml/ml.hpp>

#include<fstream>

#include<iostream>

#include<vector>

#include <stdlib.h>

#include<dirent.h>

#include<string.h>

#include<math.h>

#include<Windows.h>

#include<Winbase.h>

#include<TCHAR.h>

using namespace cv;

using namespace std;

static Mat src; Mat src\_gray;

static float n,m,ratio; //前两点和高度之比

static float a,b,ratio\_shape; //长宽之比

static float c,d,ratio\_shape\_h;

static float e,f,ratio\_shape\_best;

static int best=0;

static int muchpoly;

static vector<Point> model;

static int template\_number=0;

static int matched=0;

string read\_value(string section, string key, string config\_path)

{

char read\_value[100];

DWORD len = 100;

char section\_char[100],key\_char[100],config\_path\_char[100];

strcpy(section\_char, section.c\_str());

strcpy(key\_char, key.c\_str());

strcpy(config\_path\_char, config\_path.c\_str());

GetPrivateProfileString(TEXT(section\_char), TEXT(key\_char),TEXT("Not Found"),read\_value,len,TEXT(config\_path\_char));

return read\_value;

}

string GetFilename (const string& str)

{

size\_t found;

string file;

found=str.find\_last\_of("/\\");

file = str.substr(found+1);

return file;

}

string GetFoldername (const string& str)

{

size\_t found;

string folder;

found=str.find\_last\_of("/\\");

folder = str.substr(0,found);

return folder;

}

static float Angle(Point cen, Point first, Point second)

{

float dx1=0, dx2=0, dy1=0, dy2=0;

float angle=0;

dx1 = first.x - cen.x;

//dx1=fabs(dx1);

dy1 = first.y - cen.y;

// dy1=fabs(dy1);

dx2 = second.x - cen.x;

// dx2=fabs(dx2);

dy2 = second.y - cen.y;

// dy2=fabs(dy2);

float c = (float)sqrt(dx1 \* dx1 + dy1 \* dy1) \* (float)sqrt(dx2 \* dx2 + dy2 \* dy2);

if (c == 0) return -1;

angle = (float)acos((dx1 \* dx2 + dy1 \* dy2) / c);

return angle;

}

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

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Detect Curved Arrow\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

IplImage \*desimg,\*srcimg;

string config\_path = argv[1];//"C:\\Config\\Config.ini";

string path = argv[2];

string dirName = GetFoldername(path);//read\_value("Aerobridge","dummy",config\_path);

string flightName = GetFilename(dirName);

cout<<flightName<<endl;

DIR \*dir;

dir = opendir(dirName.c\_str());

string imgName;

struct dirent \*ent;

if (dir != NULL)

{

while ((ent = readdir (dir)) != NULL)

{

imgName= GetFilename(path);//ent->d\_name;

if(imgName.compare(".")!= 0 && imgName.compare("..")!= 0)

{

string aux;

aux.append(dirName);

aux.append("\\");

aux.append(imgName);

//aux.append(orig\_path);

cout << aux << endl;

Mat src= imread(aux);

//imshow(aux,src);

waitKey(0);

//build the template

string template\_arrow\_str = read\_value("CurvedArrow","template\_arrow",config\_path);

cout<<template\_arrow\_str<<endl;

//char rect\_file\_char[100];

//strcpy(rect\_file\_char, rect\_file\_str.c\_str());

Mat src\_T = imread(template\_arrow\_str);

Mat gray\_T,bw\_T;

Mat src\_gray\_T;

GaussianBlur(src\_T, src\_gray\_T, Size(3, 3), 2, 2);

cvtColor(src\_gray\_T,gray\_T,CV\_BGR2GRAY);

Canny(gray\_T,bw\_T,1200,600,5,true);

vector<vector<Point> > template\_handle;

vector<Vec4i> hierarchy\_T;

findContours(bw\_T.clone(), template\_handle,hierarchy\_T,CV\_RETR\_EXTERNAL, CV\_CHAIN\_APPROX\_SIMPLE);

//cout<<template\_handle.size()<<endl;

float max=0;

for (int i=0;i<template\_handle.size();i++)

{

float area;

area= contourArea(template\_handle[i]);

if( area > max)

{

max=area;

template\_number=i;

}

}

approxPolyDP(Mat(template\_handle[template\_number]), model, arcLength(Mat(template\_handle[template\_number]), true)\*0.0095, true);

Mat template\_handle\_p = Mat::zeros(src\_T.rows,src\_T.cols, CV\_8UC1);

drawContours(template\_handle\_p, template\_handle,template\_number, Scalar(255),2,8,hierarchy\_T);

/\*namedWindow("template\_handle",CV\_WINDOW\_NORMAL);

imshow("template\_handle",template\_handle\_p);\*/

//input the image

//Mat src = cv::imread("000204.png");

//namedWindow("origin",CV\_WINDOW\_NORMAL);

//imshow("origin",src);

// get the edge of the input image

Mat gray,bw;

GaussianBlur(src, src\_gray, cv::Size(9, 9), 2, 2);

cvtColor(src\_gray,gray,CV\_BGR2GRAY);

Canny(gray,bw,1200,600,5,true);

/\* namedWindow("canny",CV\_WINDOW\_NORMAL);

imshow("canny",bw);\*/

//close operation

Mat src\_close;

Size kernalSize (7,7);

Mat element = getStructuringElement (MORPH\_ELLIPSE, kernalSize, Point(1,1) );

morphologyEx( bw, src\_close, MORPH\_CLOSE, element );

/\*namedWindow("morphology",CV\_WINDOW\_NORMAL);

imshow("morphology",src\_close);\*/

//find the contours

vector<vector<Point> > contours;

vector<cv::Vec4i> hierarchy;

findContours(src\_close.clone(), contours,hierarchy,CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE);

//cout<<hierarchy.size()<<endl;

//show the contours of the input image

Mat savedGrayMat = Mat::zeros(src.rows,src.cols, CV\_8UC1);

//drawContours(savedGrayMat, contours,-1, Scalar(255),1,8,hierarchy);

//imshow("contours", savedGrayMat);

//approximate of the contours,and get the convex

vector<vector<Point> > POLY;

vector<vector<Point> > POLY\_precise;

vector<vector<Point> > POLY\_rough;

vector<vector<Point> > POLY\_best;

vector<Point> approx;

vector<Point> convex\_hull;

//output image

Mat dst = src.clone();

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

{

approxPolyDP(Mat(contours[i]), approx, arcLength(Mat(contours[i]), true)\*0.0095, true);

POLY.push\_back(approx);

convexHull(approx, convex\_hull, true);

if (fabs(contourArea(approx)) <2000 || fabs(contourArea(approx))>10000) //将面积太小的轮廓直接跳过 注意continue用法

continue;

//calculate the up bounding rectangle

Rect bound=boundingRect(contours[i]);

float area\_rect,area\_approx,difference;

area\_rect=((bound.br().x-bound.tl().x)\*(bound.br().y-bound.tl().y));

area\_approx=fabs(contourArea(contours[i]));

difference=area\_rect-area\_approx;

//cout<<"area"<<"\t";

//cout<<area\_rect<<endl;

//cout<<area\_approx<<endl;

// if the

// if(POLY\_precise.size()==0)

{

int k;

int m=hierarchy[i][3];

if(convex\_hull.size() < approx.size() && difference<3\*area\_approx && difference>0.2\*area\_approx)

{

int len; //for this approx

len=approx.size();

int \*num=new int[len];

int \*nums=new int[len];

for(int j=0;j<len;j++)

{

num[j]=approx.at(j).y;

nums[j]=approx.at(j).x;

}

sort(num,num+len); //高度

sort(nums,nums+len); //宽度

a=num[len-1]-num[0];

b=nums[len-1]-nums[0];

ratio\_shape=b/a;

//cout<<"ratio"<<"\t";

//cout<<ratio\_shape<<endl;

//cout<<"child ratio"<<"\t";

//cout<<ratio\_shape\_h<<endl;

//cout<<endl;

//POLY\_rough.push\_back(contours[m]);

if(ratio\_shape>2 && ratio\_shape<8)

{

//cout<<"catch rough one"<<endl;

//cout<<"poly"<<"\t"; cout<<approx.size()<<endl;

//cout<<approx<<endl;

//cout<<ratio\_shape<<endl;

POLY\_rough.push\_back(approx);

}

}

}

}

if(POLY\_rough.size()>0)

{

for(int i=0;i<POLY\_rough.size();i++)

{

//find the most right point of every candidate contour

int right\_point, right=0, left\_point, left=100000;

int line\_1\_r=1, line\_1\_l=1;

int line\_2\_r=(-1), line\_2\_l=-1;

double right\_angle=0;

double left\_angle=0;

for(int j=0;j<POLY\_rough[i].size();j++)

{

if (right<POLY\_rough[i].at(j).x)

{

right=POLY\_rough[i].at(j).x;

right\_point=j;

}

if (left>POLY\_rough[i].at(j).x)

{

left=POLY\_rough[i].at(j).x;

left\_point=j;

}

}

if(right\_point==POLY\_rough[i].size()) line\_1\_r=-POLY\_rough[i].size()+1;

if(right\_point==0) line\_2\_r=POLY\_rough[i].size()-1;

if(left\_point==POLY\_rough[i].size()) line\_1\_l=-POLY\_rough[i].size()+1;

if(left\_point==0) line\_2\_l=POLY\_rough[i].size()-1;

/\*circle( dst, POLY\_rough[i].at(right\_point), 4, Scalar(0,0,255), -1, 8, 0 );

circle( dst, POLY\_rough[i].at(left\_point), 4, Scalar(0,0,255), -1, 8, 0 );\*/

/\*line(dst,POLY\_rough[i].at(right\_point),POLY\_rough[i].at(right\_point+line\_1\_r),Scalar(0,0,255),2);

line(dst,POLY\_rough[i].at(right\_point),POLY\_rough[i].at(right\_point+line\_2\_r),Scalar(0,0,255),2);\*/

//line(dst,POLY\_rough[i].at(left\_point),POLY\_rough[i].at(left\_point+line\_1\_l),Scalar(0,0,255),5);

//line(dst,POLY\_rough[i].at(left\_point),POLY\_rough[i].at(left\_point+line\_2\_l),Scalar(0,0,255),5);

// left\_angle=Angle(POLY\_rough[i].at(left\_point),POLY\_rough[i].at(left\_point+line\_1\_l),POLY\_rough[i].at(right\_point+line\_2\_l));

right\_angle=Angle(POLY\_rough[i].at(right\_point),POLY\_rough[i].at(right\_point+line\_1\_r),POLY\_rough[i].at(right\_point+line\_2\_r));

right\_angle=right\_angle\*180/3.14;

if (right\_angle<70 && right\_angle>30)

POLY\_precise.push\_back(POLY\_rough[i]);

}

}

double comparing;

double min=20;

for(int j=0;j<POLY\_precise.size();j++)

{

// int max=0;

// int lenb; //FOR chirld contours

// lenb=POLY\_precise[i].size();

// int \*numb=new int[lenb];

// int \*numsb=new int[lenb];

// for(int j=0;j<lenb;j++)

// {

// numb[j]=POLY\_precise[i].at(j).y;

// numsb[j]=POLY\_precise[i].at(j).x;

// }

//sort(numb,numb+lenb); //高度

//sort(numsb,numsb+lenb); //宽度

//e=numb[lenb-1]-numb[0];

//f=numsb[lenb-1]-numsb[0];

//ratio\_shape\_best=f/e;

// if(ratio\_shape\_best>max )

//{

// max=ratio\_shape\_best;

// best=i;

//}

comparing = matchShapes(model, POLY\_rough[j],3, 0.0);

if ( comparing != 0 && comparing<min)

{

min=comparing;

best=j;

}

// POLY\_best.push\_back(POLY\_rough[best]);

}

//find the five point in the banded arrow

//find the five points

if(POLY\_precise.size()!=0)

{

vector<Moments> mu(1);

vector<Point2f> mc(1);

// compute the central momment

mu[0] = moments( POLY\_precise[best], false );

mc[0] = Point2f( mu[0].m10/mu[0].m00 , mu[0].m01/mu[0].m00 );

circle( dst, mc[0], 8, Scalar(0,0,255), -1, 8, 0 );

int right\_point, right=0,left\_point, left=100000;

for(int j=0;j<POLY\_precise[best].size();j++)

{

if (right<POLY\_precise[best].at(j).x)

{

right=POLY\_precise[best].at(j).x;

right\_point=j;

}

if (left>POLY\_precise[best].at(j).x)

{

left=POLY\_precise[best].at(j).x;

left\_point=j;

}

}

circle( dst, POLY\_precise[best].at(right\_point), 8, Scalar(0,0,255), -1, 8, 0 );

circle( dst, POLY\_precise[best].at(left\_point),8, Scalar(0,0,255), -1, 8, 0 );

float distance\_1,distance\_2,length;

int line\_1\_r=1,line\_2\_r=(-1);

int middle\_1, middle\_2;

int trangle\_1,trangle\_2;

if(right\_point==POLY\_precise[best].size()) line\_1\_r=-POLY\_precise[best].size()+1;

middle\_1= right\_point+line\_1\_r;

if(right\_point==0) line\_2\_r=POLY\_precise[best].size()-1;

middle\_2 = right\_point+line\_2\_r;

distance\_1=sqrt(pow((POLY\_precise[best].at(right\_point).x-POLY\_precise[best].at(middle\_1).x),2)+pow((POLY\_precise[best].at(right\_point).y-POLY\_precise[best].at(middle\_1).y),2));

distance\_2=sqrt(pow((POLY\_precise[best].at(right\_point).x-POLY\_precise[best].at(middle\_2).x),2)+pow((POLY\_precise[best].at(right\_point).y-POLY\_precise[best].at(middle\_2).y),2));

length=arcLength(POLY\_precise[best],true);

/\*line\_1\_r=1;

line\_2\_r=(-1);

if(distance\_1>0.05\*length) trangle\_1=middle\_1;

else{

if(middle\_1==POLY\_precise[best].size()) line\_1\_r=-POLY\_precise[best].size()+1;

trangle\_1= middle\_1+line\_1\_r;

}

if(distance\_2>0.05\*length) trangle\_2=middle\_2;

else{

if(middle\_2==0) line\_2\_r=POLY\_precise[best].size()-1;

trangle\_2 = right\_point+line\_2\_r;

}\*/

// make sure the distance between two point is bigger than threshold

while(distance\_1<0.05\*length)

{

line\_1\_r=1;

if(middle\_1==POLY\_precise[best].size()) line\_1\_r=-POLY\_precise[best].size()+1;

trangle\_1= middle\_1+line\_1\_r;

distance\_1=sqrt(pow((POLY\_precise[best].at(right\_point).x-POLY\_precise[best].at(trangle\_1).x),2)+pow((POLY\_precise[best].at(right\_point).y-POLY\_precise[best].at(trangle\_1).y),2));

middle\_1=trangle\_1;

}

trangle\_1=middle\_1;

while(distance\_2<0.05\*length)

{

line\_2\_r=-1;

if(middle\_2==0) line\_2\_r=POLY\_precise[best].size()-1;

trangle\_2 = middle\_2+line\_2\_r;

distance\_2=sqrt(pow((POLY\_precise[best].at(right\_point).x-POLY\_precise[best].at(trangle\_2).x),2)+pow((POLY\_precise[best].at(right\_point).y-POLY\_precise[best].at(trangle\_2).y),2));

middle\_2=trangle\_2;

}

trangle\_2=middle\_2;

circle( dst, POLY\_precise[best].at(trangle\_1), 8, Scalar(0,0,255), -1, 8, 0 );

circle( dst, POLY\_precise[best].at(trangle\_2), 8, Scalar(0,0,255), -1, 8, 0 );

string mycout\_str = read\_value("CurvedArrow","log\_file",config\_path);

char mycout\_char[100];

strcpy(mycout\_char, mycout\_str.c\_str());

ofstream mycout(TEXT(mycout\_char),ios::app);

/\*mycout<<"file"<<"\t";

mycout<<imgName<<"\t";

mycout<<"point1"<<"\t";

mycout<<POLY\_precise[best].at(trangle\_1)<<"\t";

mycout<<"point2"<<"\t";

mycout<<POLY\_precise[best].at(right\_point)<<"\t";

mycout<<"point3"<<"\t";

mycout<<POLY\_precise[best].at(trangle\_2)<<"\t";

mycout<<"point4"<<"\t";

mycout<<mc[0]<<"\t";

mycout<<"point5"<<"\t";

mycout<<POLY\_precise[best].at(left\_point)<<"\t";

mycout<<endl;\*/

mycout<<flightName<<";"<<imgName<<";"<<"1";

mycout<<";"<<POLY\_precise[best].at(trangle\_1);

mycout<<";"<<POLY\_precise[best].at(right\_point);

mycout<<";"<<POLY\_precise[best].at(trangle\_2);

mycout<<";"<<mc[0];

mycout<<";"<<POLY\_precise[best].at(left\_point)<<endl;

cout<<"1"<<endl;

return 1;

}

if(POLY\_precise.size()==0)

{

string mycout\_str = read\_value("CurvedArrow","log\_file",config\_path);

char mycout\_char[100];

strcpy(mycout\_char, mycout\_str.c\_str());

ofstream mycout(TEXT(mycout\_char),ios::app);

cout<<"0"<<endl;

mycout<<flightName<<";"<<imgName<<";"<<"0"<<endl;

return 0;

}

//Mat result(src.size(),CV\_8U,cv::Scalar(255));

//drawContours(dst, POLY\_precise,-1, Scalar(0,0,255), 3);

//drawContours(dst, POLY\_rough,-1, Scalar(255,0,0), 2);

drawContours(dst, POLY\_precise,best, Scalar(0,255,0), 2);

//imwrite( "D:/mobility lab/任务2 箭头/day 20 straight arrow detection/result/185.png", dst);

namedWindow("dst",CV\_WINDOW\_NORMAL);

imshow("dst", dst);

while(uchar(waitKey(1))!='Q');

}

}

closedir (dir);

} else {

cout<<"not present"<<endl;

}

return 0;

}