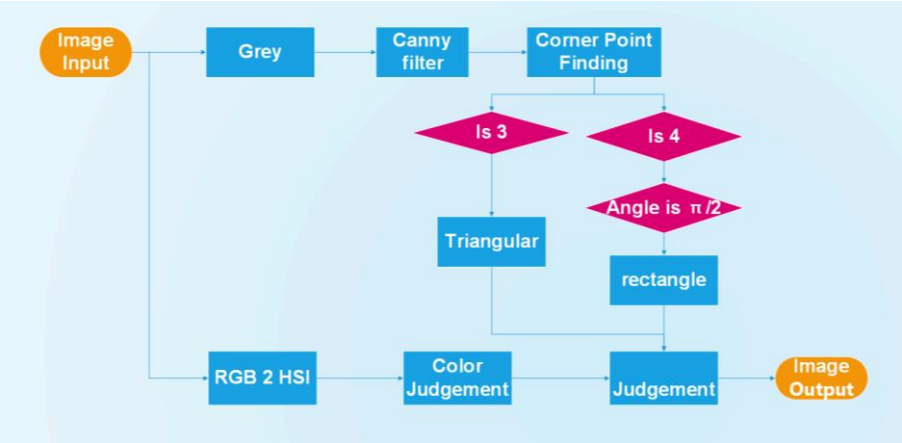
Image Recognition Documentation

Riverstone, Nanjing Institute of Technology, China
Editor: Jiang Yuwei, Wu Heng, Ma Peili
2018 04 05

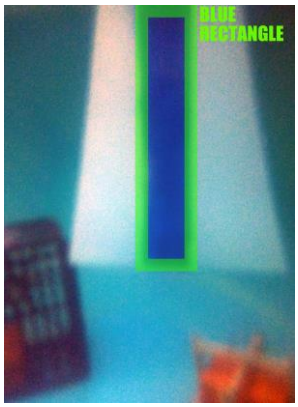
Algorithm Description:

During the competition, we need to recognize the ID of the tail wreckage of the aircraft through image recognition. We use OpenCV image processing library for corner and color analysis. According to the number of the corner points and the color of the image, the ID of the aircraft tail model can be recognized. The main steps are to first read and save the color image data transmitted by the Hikvision IP camera, and then convert the color image to a gray image. In order to reduce the influence of high-frequency noise on image edge detection, the algorithm first uses high-pass filtering algorithm to reduce high-frequency noise in the image, then the Canny operator to filter the image, and the Thread function to perform edge detection. Then use the findContours function and the approximate function approxPolyDP function to obtain the set of polygon points in the target image and use the number of points to recognize. When there are three points it is a triangle. If there are four points, you need to determine the cosine value of the angle between adjacent lines. If the cosine value is within the threshold range (-0.3 - 0.3), it is recognized as a rectangle. Finally, the color of the figure is identified. We convert the RGB color space to the HSI color space, and identify the color of the figure by determining the chromatic value in the central area of the figure. According to the shape of the figure and the color of the figure, the ID of the airplane tail model is comprehensively recognized, and the conclusion can be drawn.

Data flow graph：



Output Image:



Attached code:

```
/*********************************************************************************************/
//Group: Software Group, Riverstone Company, Nanjing Institute of Technology
//Author: Ma Peili
//Time: March 27, 2018
//Description: This documentation includes the related codes used in the subtask of Task1
//-ROV image recognition of 2018 MATE ROV Competition.
//Related functions in the OpenCV library are used to process images.
/*********************************************************************************************/
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <cmath>
#include <iostream>

using namespace cv;
using namespace std;


struct SimpleShapes {
	Mat img_src;
	Mat img_dst;
	Mat img_bw;
	vector<RotatedRect> rectangles;//旋转矩形类
	vector<vector<Point>> triangles;//取点三角
};

SimpleShapes Water_Shape(Mat, int);//返回该类

double angleCos(Point, Point, Point);

void setLabel(Mat&, const string, vector<Point>&);

uchar* PictureKind(int, int, Mat);

uchar* Average(uchar*, uchar*, uchar*);
库：water_competition.cpp
#include "water_competition.h"
/**
* 长方形，三角形识别
*/
IplImage iplimg_hsv;
uchar*finish_ptr;
SimpleShapes Water_Shape(Mat pic, int lowThreshold)
{
	static Mat img_src, img_dst; //原目标图像，目的目标图像
	img_src.release(); img_dst.release();//释放内存
	static Mat img_gray, img_canny;//描边，降噪处理图像
	img_gray.release(); img_canny.release();
	//Mat img_hsv;//转 hsv 图像
	//img_hsv.release();
	uchar* img1_c, *img2_c, *img3_c,*average_c;
	static SimpleShapes shapes;
	static vector<vector<Point>> contours; contours.clear();//findContours 获得点集
	static vector<Point> vertices_contour; vertices_contour.clear();//approxPolyDP 逼近点集
	static vector<vector<Point>> triangles; triangles.clear();
	static vector<RotatedRect> rectangles; rectangles.clear();
	static int RATIO = 4;
	static int kernel_size = 3;
	img_src = pic;
	//cvtColor(img_src, img_hsv, CV_BGR2HSV);
	//转化为灰度
	cvtColor(img_src, img_gray, CV_BGR2GRAY);
	GaussianBlur(img_gray, img_gray, Size(3, 3), 1);
	//blur(img_gray, img_gray, Size(3, 3));
	//canny 滤波
```

```
Canny(img_gray, img_canny, lowThreshold, RATIO*lowThreshold, kernel_size);//滞后阈值 1，滞后阈值 2，
//imshow("1", img_canny);
                                        //查找 canny 后轮廓，并保存一组由连续的 Point 点构成的点的集合的向量，
CV_RETR_EXTERNAL 只检测最外围轮廓
findContours(img_canny.clone(), contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);
img_dst = img_src.clone();

//find shapes
for (int i = 0; i < contours.size(); i++) {
//比例精度近似轮廓，去得到逼近点集
approxPolyDP(Mat(contours[i]), vertices_contour, arcLength(Mat(contours[i]), true)*0.02, true);

//跳过小的或非凸的物体，利用 isContourConvex 测试轮廓的凸性
if (fabs(contourArea(contours[i])) < 1 || !isContourConvex(vertices_contour)) {
continue;
}

if (vertices_contour.size() == 3) {//三角形
triangles.push_back(vertices_contour); //添加到检测到的三角形集合中。

                        //绘制，利用 vertices_countour 来获取两点线段
for (int i = 0; i < 3; i++)
line(img_dst, vertices_contour[i], vertices_contour[(i + 1) % 3], Scalar(0, 255, 0), 3);
int weight_average = (vertices_contour[0].x + vertices_contour[1].x + vertices_contour[2].x) / 3;
int height_average = (vertices_contour[0].y + vertices_contour[1].y + vertices_contour[2].y) / 3;//中心
img1_c=PictureKind(height_average,weight_average, img_src);
img2_c= PictureKind(height_average-5, weight_average-5, img_src);
img3_c= PictureKind(height_average + 5, weight_average+5, img_src);//去相应位置的像素值
average_c=Average(img1_c, img2_c, img3_c);
//printf("平均%c,%c,%c,\t", average_c[0], average_c[1], average_c[2]);
cout << (int)average_c[2] << "+" << (int)average_c[1] << "+" << (int)average_c[0]<<"\t";
if (average_c[2] > average_c[1] && average_c[2] > average_c[0])
{
setLabel(img_dst, "UH8", contours[i]);
}
else if (average_c[1]>average_c[2] && average_c[1] > average_c[0])
{
setLabel(img_dst, "L6R", contours[i]);
}
else if (average_c[0]>average_c[1] && average_c[0] > average_c[2])
{
setLabel(img_dst, "G7C", contours[i]);
}
//setLabel(img_dst, "TRI", contours[i]);
}
else if (vertices_contour.size() == 4) {
int nb_vertices = 4;

//从各个角计算余弦
vector<double> cos;
for (int j = 2; j < nb_vertices + 1; j++) {
cos.push_back(angleCos(vertices_contour[j%nb_vertices], vertices_contour[j - 2], vertices_contour[j - 1]));
}

//“COS”按升序存储
sort(cos.begin(), cos.end());

//保存最大最小 cos
double mincos = cos.front();
double maxcos = cos.back();

//检查角是直角
if (mincos >= -0.3 && maxcos <= 0.3) {
RotatedRect rotRect = minAreaRect(contours[i]);//矫正矩形
rectangles.push_back(rotRect); //添加到检测到的矩形集合中。

//绘制
Point2f coins_rect[4];
```

```
		rotRect.points(coins_rect);
		for (int i = 0; i < 4; i++)
			line(img_dst, coins_rect[i], coins_rect[(i + 1) % 4], Scalar(0, 255, 0), 3);
		int weight_average = (coins_rect[0].x + coins_rect[1].x + coins_rect[2].x+ coins_rect[3].x) / 4;
		int height_average = (coins_rect[0].y + coins_rect[1].y + coins_rect[2].y+ coins_rect[3].y) / 4;
		img1_c = PictureKind(height_average, weight_average, img_src);
		img2_c = PictureKind(height_average - 5, weight_average - 5, img_src);
		img3_c = PictureKind(height_average + 5, weight_average + 5, img_src);
		average_c=Average(img1_c, img2_c, img3_c);
		if (average_c[2] > average_c[1] && average_c[2] > average_c[0])
		{
			setLabel(img_dst, "S1P", contours[i]);
		}
		else if(average_c[1]>average_c[2] && average_c[1] > average_c[0])
		{
			setLabel(img_dst, "JW3", contours[i]);
		}
		else if (average_c[0]>average_c[1] && average_c[0] > average_c[2])
		{
			setLabel(img_dst, "A2X", contours[i]);
		}
		//cout << (int)average_c[2] << "+" << (int)average_c[1] << "+" << (int)average_c[0]<<"+";
		//写出
		//stringstream sentence;
		//sentence << "RECT rot = " << rotRect.angle;//转动角度
		//sentence << "RECT";
		//string rotLabel = sentence.str();
		//setLabel(img_dst, rotLabel, contours[i]);
		}
		}
		}
		//保存数据
		shapes.img_src = img_src;
		shapes.img_dst = img_dst;
		shapes.img_bw = img_canny;
		shapes.rectangles = rectangles;
		shapes.triangles = triangles;
		return shapes;//返回数据
}
uchar* Average(uchar*img1_c, uchar*img2_c, uchar*img3_c)
{
		finish_ptr= img1_c;
		finish_ptr[0] = (img1_c[0] + img2_c[0] + img3_c[0])/3;
		finish_ptr[1] = (img1_c[1] + img2_c[1] + img3_c[1]) / 3;
		finish_ptr[2] = (img1_c[2] + img2_c[2] + img3_c[2]) / 3;
		return finish_ptr;
}

uchar* PictureKind(int height_average,int weight_average,Mat img_src)
{
		char text[20];
		iplimg_hsv = IplImage(img_src);
		IplImage* pBinary =&IplImage(iplimg_hsv);
		uchar* ptr = cvPtr2D(pBinary, height_average, weight_average, NULL);
		//cout<< (int)ptr[2]<<"+" <<(int)ptr[1] << "+" << (int)ptr[0];
		return ptr;
}
double angleCos(Point pt1, Point pt2, Point 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));
}

void setLabel(Mat& img, const string label, vector<Point>& contour) {
		static int fontface = FONT_HERSHEY_SIMPLEX;
		static double scale = 0.4;
```

```
        static int thickness = 1;
        static int baseline = 0;

        Size text = getTextSize(label, fontface, scale, thickness, &baseline);//设置 text 文本格式
        Rect r = boundingRect(contour); //returns the smallest rectangle containing the "contour"

        Point pt(r.x + ((r.width - text.width) / 2), r.y + ((r.height + text.height) / 2));
        rectangle(img, pt + Point(0, baseline), pt + Point(text.width, -text.height), CV_RGB(255, 255, 255), CV_FILLED);
        putText(img, label, pt, fontface, scale, CV_RGB(0, 0, 0), thickness, 8);
}
```

Main.cpp

```
#include "water_competition.h"

SimpleShapes shapes;
int edgeThresh = 1;
int low_Threshold = 50;
const int max_lowThreshold = 255;

int main()
{
        Mat imgsrc = imread("1.png");
        namedWindow("Image Result", CV_WINDOW_AUTOSIZE);
        namedWindow("Image Canny", CV_WINDOW_AUTOSIZE);
        createTrackbar("Min Threshold:", "Image Canny", &low_Threshold, max_lowThreshold);
        while (true) {
        //Extration of simple shape (triangles/rectangles/circles) from the frame
        shapes = Water_Shape(imgsrc, low_Threshold);
        imshow("Image Result", shapes.img_dst);
        imshow("Image Canny", shapes.img_bw);
        if (waitKey(30) == 27) {
        destroyWindow("Image Traitee");
        destroyWindow("Image Canny");
        destroyAllWindows();
        cout << "esc key is pressed by user" << endl;
        return 0;
        }
        }}
```