#include "loc\_detect.h"

#include <iostream>

#include <io.h>

loc\_detect::loc\_detect()

{

}

loc\_detect::~loc\_detect()

{

}

void loc\_detect::image\_cut\_process(Mat src, Mat & out\_img) {

//图片腐蚀操作

Mat s = getStructuringElement(MORPH\_RECT, Size(5, 3));

Mat E;

erode(src, E, s, Point(-1, -1), 10);

Canny(E, E, 50, 100);

//imshow("Canny", E);

//寻找矩形框

vector<vector<Point>> contours;

findContours(E, contours, CV\_RETR\_EXTERNAL, CV\_CHAIN\_APPROX\_SIMPLE);

int num = contours.size();

cout << num << endl;

//遍历矩形框，并且将其画在图上

int key = 0;

Rect out\_rect;

for (int i = 0; i < num; i++)

{

Rect rect = boundingRect(contours[i]);

if (rect.area() > out\_rect.area())

{

out\_rect = rect;

key = i;

//rectangle(src, rect, Scalar(100));

}

}

BOX\_X1X2Y1Y2 out\_box;

out\_box.x1 = out\_rect.x;

out\_box.x2 = out\_rect.x + out\_rect.width;

out\_box.y1 = out\_rect.y;

out\_box.y2 = out\_rect.y + out\_rect.height;

out\_img = resize\_out\_image(src, out\_box);

/\*Mat img1;

img1 = src(out\_rect);

imshow("输出矩形框", img1);

waitKey(0);\*/

}

void get\_image\_names(string file\_path, vector<string>& file\_names)

{

intptr\_t hFile = 0;

\_finddata\_t fileInfo;

cout << file\_path << endl;

hFile = \_findfirst(file\_path.c\_str(), &fileInfo);

cout << hFile << endl;

if (hFile != -1) {

do {

//如果为文件夹，可以通过递归继续遍历，此处我们不需要

if ((fileInfo.attrib & \_A\_SUBDIR)) {

continue;

}

//如果为单个文件直接push\_back

else {

file\_names.push\_back(fileInfo.name);

cout << fileInfo.name << endl;

}

} while (\_findnext(hFile, &fileInfo) == 0);

\_findclose(hFile);

}

}

//等比例缩放，实际背景填充到指定大小

loc\_detect::BOX\_X1X2Y1Y2 loc\_detect::resive\_save\_image(BOX\_X1X2Y1Y2 box, int width, int height)

{

float w1 = box.x2 - box.x1;

float h1 = box.y2 - box.y1;

float rate = float(fit\_w) / fit\_h;

//防止尺寸越界

float X1 = box.x1 > 0 ? box.x1 : 0;

float X2 = box.x2 < width ? box.x2 : width;

float Y1 = box.y1 > 0 ? box.y1 : 0;

float Y2 = box.y2 < height ? box.y2 : height;

//等比例缩放

float H, W;

if (w1 / h1 > rate)

{

H = w1 / rate;

Y1 = box.y1 - (H - h1) / 2;

Y2 = box.y2 + (H - h1) / 2;

if (Y1 < 0)

{

Y1 = 0;

Y2 = H;

}

if (Y2 > height)

{

Y2 = height;

Y1 = height - H;

}

}

else if ((w1 / h1) < rate)

{

W = h1 \* rate;

X1 = box.x1 - (W - w1) / 2;

X2 = box.x2 + (W - w1) / 2;

if (X1 < 0)

{

X1 = 0;

X2 = W;

}

if (X2 > width)

{

X2 = width;

X1 = width - W;

}

}

BOX\_X1X2Y1Y2 tmp;

//防止尺寸越界

tmp.x1 = X1 > 0 ? X1 : 0;

tmp.x2 = X2 < width ? X2 : width;

tmp.y1 = Y1 > 0 ? Y1 : 0;

tmp.y2 = Y2 < height ? Y2 : height;

return tmp;

}

//从原图中截取需要的区域

Mat loc\_detect::resize\_out\_image(Mat img, BOX\_X1X2Y1Y2 box)

{

int height = img.rows;

int width = img.cols;

//边界往外扩大的像素

int x1\_ = box.x1 - 5;

int x2\_ = box.x2 + 5;

int y1\_ = box.y1 - 2;

int y2\_ = box.y2 + 2;

BOX\_X1X2Y1Y2 box0;

box0.x1 = x1\_;

box0.x2 = x2\_;

box0.y1 = y1\_;

box0.y2 = y2\_;

//等比例缩放，填充实际背景区域

BOX\_X1X2Y1Y2 box1 = loc\_detect::resive\_save\_image(box0, width, height);

int X1 = box1.x1;

int Y1 = box1.y1;

int WIDTH = box1.x2 - box1.x1;

int HEIGHT = box1.y2 - box1.y1;

//从原图中截取指定区域保存

Mat mask = img(Rect(X1, Y1, WIDTH, HEIGHT));

Mat resize\_image;

resize(mask, resize\_image, Size(fit\_w, fit\_h));

return resize\_image;

}

//

//int main() {

// string path = R"(F:\datas\原始数据\20200630\_拍摄（OCR-辣条缺陷）\OCR-日期\类别1\_cut\20200623T\\*.jpg)";

// vector<string> file\_names;

// get\_image\_names(path, file\_names);

// string pre\_path = path.erase(path.find\_last\_of("\\"));//图片名称

// cout << pre\_path << endl;

// loc\_detect loc;

// Mat out\_img;

// for (int i = 0; i < file\_names.size(); i++)

// {

// string image\_path = pre\_path + "/" + file\_names.at(i);

// Mat src;

// cout << image\_path << endl;

// //读取图片

// src = imread(image\_path, 0);

// imshow("src", src);

// //开始模型检测

// loc.image\_cut\_process(src, out\_img);

//

// /\*imshow("out\_img", out\_img);

// waitKey(0);\*/

// }

//

// ////读取图片

// //Mat img1 = imread("C:/Users/pc/Desktop/img/2.jpg",0);

// //if (!img1.data)

// // cout << "没有读取图片。。。 " << endl;

// //loc\_detect loc;

// //Mat out\_img;

// //loc.image\_cut\_process(img1, out\_img);

// return 0;

//}