# 缺陷检测算子快速入门手册

目录

[缺陷检测算子快速入门手册 1](#_Toc15351)

[1．引言 3](#_Toc11693)

[1.1编写目的 3](#_Toc12583)

[1.2其它说明 3](#_Toc1945)

[2. 算子说明 3](#_Toc24982)

[2.1 缺陷检测算子 3](#_Toc31703)

[2.1.1类型定义及说明 3](#_Toc13271)

[2.1.2相关数据结构 3](#_Toc14650)

[2.1.3 调用流程 4](#_Toc7131)

[2.1.4 使用案例 5](#_Toc27339)

[2.1.5 注意事项 16](#_Toc27636)

[2.1.6 算子配置 16](#_Toc26426)

[3. 附录 17](#_Toc24342)

# 1．引言

## 1.1编写目的

此文档是为了说明缺陷检测算子的功能，以及调用算子的具体流程，方便其他人使用开发算子。

## 1.2其它说明

算子实例的通用构造、执行接口、算法配置文件等说明可见附录《X\_vision SDK快速入门手册》。

# 2. 算子说明

## 2.1 缺陷检测算子

### 2.1.1类型定义及说明

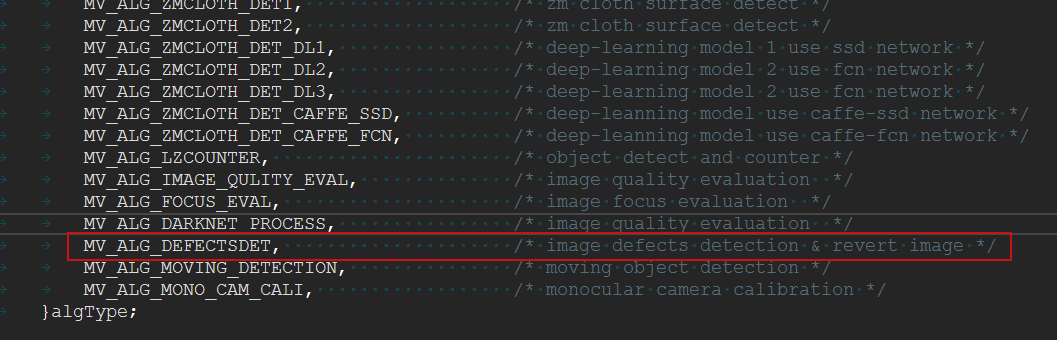
算子类型为：algType::MV\_ALG\_DEFECTSDET，主要功能是进行缺陷检测、图像旋转以及定位功能。

### 2.1.2相关数据结构

相关数据结构定义如下：

/\* alg sub\_type \*/

#define MV\_ALG\_DEFECTSDET 34



typedef struct defectsFeature

{/\* match Objects \*/

int index; /\* object index \*/

int mat\_index; /\* matched object index \*/

int valid; /\* valid flag \*/

int num\_pixels;

float mean\_gray;

float area;

float wh\_rate;

float angle;

float contrast;

float homogeneity;

float entropy;

float energy;

float correlation;

int reserved1;

float reseved2;

}defectsFeature;

typedef struct

{

mvRGBImage tmp\_img; /\* template imgage \*/

mvRGBImage cur\_img; /\* current process imgage \*/

float hmat[9]; /\* 3x3 transform mat \*/

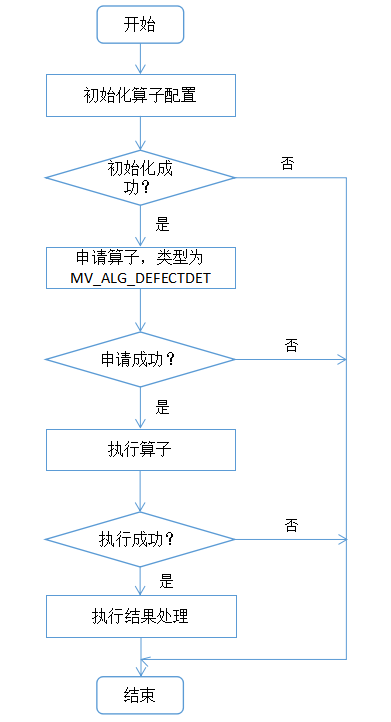
float angle; /\* image rotation angle \*/

std::vector<defectsFeature> matObjFeature; /\* defects feature \*/

}defectsUserResult;

### 2.1.3 调用流程

缺陷检测算子调用流程如下：



缺陷检测流程图

### 2.1.4 使用案例

#include <stdio.h>

#include <stdlib.h>

#include <opencv2/opencv.hpp>

#include "DllmvInterface.h"

#include <windows.h>

#include <iostream>

#include <vector>

#include<opencv2/ml/ml.hpp>

using namespace cv;

using namespace std;

static RNG rng(12345);

algDllHandle \*pAlg = NULL;

//alg算法配置

//算法缩放系数

#define MV\_ALG\_SCL 0.25

//算法配件路径

#if 1

#define MV\_CONFIG\_PATH "D:/imvcfg/"

#define TEST\_SAMPLE\_PATH "D:/pic/aoi/"

#define TEST\_PATH "imageList.txt"

#else

#define MV\_CONFIG\_PATH "./imvcfg/"

#define TEST\_SAMPLE\_PATH "./pic/"

#define TEST\_PATH "imageList.txt"

#endif

//输入图像缩放系数

#define MV\_INTPUT\_IMAGE\_SCAL 1

//算法类型

#define MV\_ALG\_TYPE (MV\_ALG\_DEFECTSDET)

typedef struct defectsFeature

{/\* match Objects \*/

int index; /\* object index \*/

int mat\_index; /\* matched object index \*/

int valid; /\* valid flag \*/

int num\_pixels;

float mean\_gray;

float area;

float wh\_rate;

float angle;

float contrast;

float homogeneity;

float entropy;

float energy;

float correlation;

int reserved1;

float reseved2;

}defectsFeature;

typedef struct

{

mvRGBImage tmp\_img; /\* template imgage \*/

mvRGBImage cur\_img; /\* current process imgage \*/

float hmat[9]; /\* 3x3 transform mat \*/

float angle; /\* image rotation angle \*/

std::vector<defectsFeature> matObjFeature; /\* defects feature \*/

}defectsUserResult;

/\*\*\*

load images from file-list

\*/

void load\_images(const string & prefix, const string & filename, vector< Mat > & img\_lst, int flag, Size ct, int \*ww, int \*hh)

{

string line;

ifstream file;

Mat img1, img2;

int width = ct.width;

int height = ct.height;

file.open((prefix + filename).c\_str());

if (!file.is\_open())

{

cerr << "Unable to open the list of images from " << filename << " filename." << endl;

exit(-1);

}

bool end\_of\_parsing = false;

while (!end\_of\_parsing)

{

getline(file, line);

if (line.empty()) // no more file to read

{

end\_of\_parsing = true;

break;

}

Mat img = imread((prefix + line).c\_str()); // load the image

if (img.empty()) // invalid image, just skip it.

continue;

#ifndef \_DEBUGX

//imshow("image", img);

//waitKey(10);

#endif

if (flag)

{

//resize

resize(img, img2, Size(width, height));

#ifdef \_DEBUG

imshow("image", img2);

waitKey(10);

#endif

img\_lst.push\_back(img2.clone());

}

else

img\_lst.push\_back(img.clone());

\*ww = img.cols;

\*hh = img.rows;

}

}

void load\_images(const string & prefix, const string & filename, vector< Mat > & img\_lst, int flag, float scl, int \*ww, int \*hh)

{

string line;

ifstream file;

Mat img1, img2;

file.open((prefix + filename).c\_str());

if (!file.is\_open())

{

cerr << "Unable to open the list of images from " << filename << " filename." << endl;

exit(-1);

}

bool end\_of\_parsing = false;

while (!end\_of\_parsing)

{

getline(file, line);

if (line.empty()) // no more file to read

{

end\_of\_parsing = true;

break;

}

Mat img = imread((prefix + line).c\_str()); // load the image

if (img.empty()) // invalid image, just skip it.

continue;

#ifndef \_DEBUGX

//imshow("image", img);

//waitKey(10);

#endif

if (flag)

{

//resize

resize(img, img2, Size(img.cols \* scl, img.rows \* scl));

#ifdef \_DEBUG

imshow("image", img2);

waitKey(10);

#endif

img\_lst.push\_back(img2.clone());

\*ww = img2.cols;

\*hh = img2.rows;

}

else

{

img\_lst.push\_back(img.clone());

\*ww = img.cols;

\*hh = img.rows;

}

}

}

/\*\*\*

draw mvCCLItems

\*/

void cclTtemsDraw(mvCCLItems \*pComps, cv::Mat img)

{

int i, j, ind;

int tmpx, tmpy;

int ww, hh;

int flag;

mvCCLItem \*pComp;

mvRect rc;

mvPoint pt;

int x, xx, y, yy;

cv::Point pp1, pp2;

for (i = pComps->num\_comp - 1; i >= 0; i--)

{

cv::Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.uniform(0, 255));

pComp = &pComps->ccomps[i];

if (!pComp->vflag)

continue;

x = pComp->rec\_bound.up\_left.x;

y = pComp->rec\_bound.up\_left.y;

xx = pComp->rec\_bound.dw\_right.x;

yy = pComp->rec\_bound.dw\_right.y;

rc = pComp->rec\_bound;

pt = pComp->pnt\_mass;

//mvImageDrawLine(pImage, pt, pt, 255, 1);

//mvImageDrawRectangle(pImage, 255, rc.up\_left.y, rc.dw\_right.y,

// rc.up\_left.x, rc.dw\_right.x);

cv::rectangle(img, cv::Rect(x, y, xx-x, yy-y), Scalar(0, 0, 255));

for (int k = 0; k < pComp->pnt\_countour.nump; k++)

{

//mvLog("maxOds:%f,", pods[k]);

pp1.x = pComp->pnt\_countour.pnts[k].x;

pp1.y = pComp->pnt\_countour.pnts[k].y;

cv::line(img, pp1, pp1, color, 1, 8, 0);

}

//draw "+"

int len = 10;

pp1.x = pComp->pnt\_center.x;

pp1.y = pComp->pnt\_center.y - len;

pp2.x = pComp->pnt\_center.x;

pp2.y = pComp->pnt\_center.y + len;

rectangle(img, pp1, pp2, CV\_RGB(0, 255, 0));

pp1.x = pComp->pnt\_center.x - len;

pp1.y = pComp->pnt\_center.y;

pp2.x = pComp->pnt\_center.x + len;

pp2.y = pComp->pnt\_center.y;

rectangle(img, pp1, pp2, CV\_RGB(0, 255, 0));

/\*object centor \*/

pp1.x = pComp->pnt\_center.x, pp1.y = pComp->pnt\_center.y;

line(img, pp1, pp1, cvScalar(0, 0, 255),2);

}

imshow("ccls", img);

return;

}

void\* algProcessDemo(algType type)

{

int ret;

mvInputImage orgImage;

initParam param;

mvResult \*pRes;

mvEngineCfg pa;

int keyval = 0, key;

CvCapture \*pCap;

vector< Mat > test\_lst;

int frameNum;

int ww, hh;

defectsUserResult \*eval;

std::string text;

char sss[100];

string test\_dir = TEST\_SAMPLE\_PATH;

string test\_sample = TEST\_PATH;

//设置alg param初始化参数

strcpy(param.cfg\_path, MV\_CONFIG\_PATH);

//param.tmp\_offset = offset;

//param.tmp\_roi = loc;

param.online\_temp = 0; //not used

param.tmp\_img.pframe = NULL;

param.tmp\_img.width = 0; //temp->width;

param.tmp\_img.height = 0; //temp->height;

param.tmp\_img.channels = 0; //temp->nChannels;

param.tmp\_img.depth = 0; //temp->depth;

param.tmp\_img.wstep = 0; //temp->wstep;

param.tmp\_img.type = MV\_CV\_IPL; //MV\_CV\_IPL;

//allocation INPUT\_WIDTH X INPUT\_HEIGHT

IplImage \*org = NULL;

if (0)

{//缩放模式

//读取本地图像

load\_images(test\_dir, test\_sample, test\_lst, 1, MV\_ALG\_SCL, &ww, &hh);

pAlg = (algDllHandle\*)mvInstanceAlloc(ww, hh, type, &param);

if (!pAlg)

return NULL;

pAlg->alg\_params.disp\_level |= (0x01 << 21);

pAlg->alg\_params.disp\_level |= (0x01 << 22);

pRes = (mvResult\*)&pAlg->result;

/\* 算法结果输出 \*/

eval = (defectsUserResult\*)pRes->user\_dat;

}

else

{//原图方式

load\_images(test\_dir, test\_sample, test\_lst, 0, MV\_ALG\_SCL, &ww, &hh);

pAlg = (algDllHandle\*)mvInstanceAlloc(ww, hh, type, &param);

if (!pAlg)

return NULL;

pAlg->alg\_params.disp\_level |= (0x01 << 21);

pAlg->alg\_params.disp\_level |= (0x01 << 22);

pRes = (mvResult\*)&pAlg->result;

/\* 算法结果输出 \*/

eval = (defectsUserResult\*)pRes->user\_dat;

}

vector< Mat >::const\_iterator img = test\_lst.begin();

vector< Mat >::const\_iterator end = test\_lst.end();

frameNum = 0;

for (int i = 0; img != end; img++)

{

static int flag = 0;

//cv::Mat frame = cv::imread(TEST\_IMAGE, -1);

cv::Mat frame = img->clone();

cv::Mat input2 = img->clone();

if (flag == 0)

{

org = cvCreateImage(cvSize(frame.cols, frame.rows), IPL\_DEPTH\_8U, 3);

flag = 1;

}

orgImage.index = frameNum; /\* index = 0, template, \*/

orgImage.pframe = (void\*)frame.data; /\* 图像数据地址 \*/

orgImage.width = frame.cols; /\* 图像宽度 \*/

orgImage.height = frame.rows; /\* 图像高度 \*/

orgImage.channels = frame.channels(); /\* 图像通道\*/

orgImage.wstep = frame.cols \* frame.channels(); /\* 图像 wstep = 宽度\* 通道数 \*/

orgImage.depth = frame.depth(); /\* 图像深度 \*/

orgImage.type = MV\_BGR24; /\*帧的格式\*/

int64 nTick = ::GetTickCount();

double ptime;

nTick = getTickCount();

/\* 算法处理 \*/

ret = mvAlgProcess(pAlg, (mvInputImage\*)&orgImage);

ptime = ((double)getTickCount() - nTick)\*1000. / getTickFrequency();

printf("processed time = %.2f ms\n", ptime);

if (ret > 0)

{//MV\_OK

cv::Mat tmp;

if (eval->tmp\_img.channels == 1)

tmp = Mat(eval->tmp\_img.height, eval->tmp\_img.width, CV\_8UC1, (void\*)eval->tmp\_img.pdata);

else if (eval->tmp\_img.channels == 3)

tmp = Mat(eval->tmp\_img.height, eval->tmp\_img.width, CV\_8UC3, (void\*)eval->tmp\_img.pdata);

imshow("tmp-img", tmp);

/\* 算法处理结果输出 \*/

eval = (defectsUserResult\*)pRes->user\_dat;

sprintf(sss, "angle = %.4f", eval->angle);

cout << sss << endl;

cout << "==============object feature==============" << endl;

for (int kk = 0; kk < pRes->mat\_objs.num\_obj; kk++)

{

matchObj matobj;

defectsFeature fea;

matobj = pRes->mat\_objs.mat\_obj[kk];

fea = eval->matObjFeature[kk];

cout << "index=" << matobj.mat\_index<<endl;

cout << "num\_pixels=" << fea.num\_pixels << endl;

cout << "area=" << fea.area << endl;

cout << "mean\_gray=" << fea.mean\_gray << endl;

cout << "angle=" << fea.angle << endl;

cout << "contrast=" << fea.contrast << endl;

cout << "homogeneity=" << fea.homogeneity << endl;

cout << "entropy=" << fea.entropy << endl;

cout << "correlation=" << fea.correlation << endl;

cout << "wh\_rate=" << fea.wh\_rate << endl;

}

cout << "==============object feature==============" << endl;

//绘制defects检测结果

mvMatchObjsDrawAndDisplay(pAlg, pRes);

//绘制组件

cclTtemsDraw(pRes->cc\_items, input2);

imwrite("ccls.png", input2);

//显示defects检测结果

imshow("det-result", frame);

imwrite("det-result.png", frame);

if (frameNum)

{

int font\_face = cv::FONT\_HERSHEY\_SIMPLEX;

mvFPoint mm, om1, om2;

//imshow("org-img", input2);

//标记点选择

mm.x = input2.cols / 2; mm.y = input2.rows / 2;

cv::circle(input2, cvPoint(mm.x, mm.y), 3, cv::Scalar(0, 255, 0));

cv::circle(tmp, cvPoint(mm.x, mm.y), 2, cv::Scalar(0, 255, 255));

//计算模板图中坐标om1

mvMap2DPoint(mm, eval->hmat, 0, om1);

cv::circle(tmp, cvPoint(om1.x, om1.y), 3, cv::Scalar(0, 255, 0));

//计算检测图中坐标om2

mvMap2DPoint(mm, eval->hmat, 1, om2);

cv::circle(input2, cvPoint(om2.x, om2.y), 3, cv::Scalar(0, 255, 255));

//imshow("tmp-img-map1", tmp);

//imshow("org-img-map1", input2);

//合并图像

cv::Mat merge, left, right;

cv::Size ss((tmp.cols + input2.cols), max(tmp.rows, input2.rows));

merge.create(ss, CV\_MAKETYPE(CV\_8U, 3));

left = merge(cv::Rect(0, 0, tmp.cols, tmp.rows));

tmp.copyTo(left);

right = merge(cv::Rect(tmp.cols, 0, input2.cols, input2.rows));

input2.copyTo(right);

//imshow("mergeright", merge);

cv::line(merge, cvPoint(om1.x, om1.y), cvPoint(mm.x + tmp.cols, mm.y), cv::Scalar(255, 255, 0), 1);

cv::line(merge, cvPoint(mm.x, mm.y), cvPoint(om2.x + tmp.cols, om2.y), cv::Scalar(127, 255, 0), 1);

cv::putText(merge, sss, cv::Point(20, 20), font\_face, 1, cv::Scalar(0, 255, 255), 2);

imshow("merge", merge);

imwrite("merge.png", merge);

//2D图像旋转变换

mvTransform2DImage((unsigned char\*)input2.data, input2.cols, input2.rows,

input2.channels(), eval->hmat);

imshow("org-img-tran", input2);

imwrite("org-img-tran.png", input2);

mvImage cc;

mvPoint ee;

cc.height = input2.rows;

cc.width = input2.cols;

cc.channels = input2.channels();

cc.pdata = input2.data;

cc.nsize = input2.step \* input2.rows;

ee.x = input2.cols / 2; ee.y = input2.rows / 2;

//任意2D图象变换，cc位旋转轴点,angle = -45, scale = 0.8

mvRotate2DImage(cc, ee,-45, 0.3);

imwrite("2d-trans.png", input2);

}

frameNum++;

cvWaitKey();

}

}

if (1)

{

printf("press anykey to continue...\n");

cvWaitKey();

mvInstanceDelete(pAlg);

}

printf("succussfully destroy alg handle!\n");

while (1);

return (void\*)pAlg;

}

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

{

int key;

////process

algProcessDemo(MV\_ALG\_TYPE);

key = cvWaitKey(0);

while (key != 'q' && key != 'Q')

{

cvWaitKey(0);

}

return 0;

}

### 2.1.5 注意事项

①初始化算子时，配置文件的文件夹路径要以“/”结束，例如：”D:/imvcfg/”

②存放图片的文件夹必须有图片的列表文件imagelist.txt,并将图片名字写在该文件中，每行一个，不能有空格。

### 2.1.6 算子配置

##############################

#algorithm type

algType=MV\_ALG\_DEFECTSDET

#algorithm metho: 0,1...，

sub\_metho=0

#object filter, default: 0

use\_filter=0

#edge detect metho, default: 80

#边缘敏感系数

edge\_thres=80

#detected object pixels, default: 16

min\_pixels=0

#mean-area

min\_area=3

#contour metho(0,1,2,3,4) ,default: 2,

#0,表示只检测最外层轮廓， 其他，检测所有轮廓，每种层次关系不一样。

match\_metho=2

#shape minimal pixels, default: 16

reserver5=0

#iteration value(>0), default: 100

corner\_thres=100

#gray threshold(0~255) , default: 80

diffVal=80

#iteration different stop theshold(>.0）,default 0.1

co\_thres=0.01

#display level

#disp\_level=3281073

##############################

# 3. 附录

[1] 《X\_vision SDK快速入门手册》