Test_ImgProcess 项目工程使用说明

此项目工程文件仅供参考，如有问题，欢迎指正。

此示例文件是基于 opencv3.1 编写，在调试之前需先安装 opencv 环境，具体安装方式请查看：

http://docs.opencv.org/3.0.0/d6/d8a/tutorial_windows_visual_studio_Opencv.html

(一) SDK 环境配置

VIPL_SDK 由三部分组成，分别是检测（FD）、定位（FLL）、识别（FID）。

1) **模型文件引入：**在 ImgProcess.cpp 文件的 ImgProcess::ImgProcess 函数中修改模型文件（.dat）的路径。
2) **头文件引入：**分别将检测、定位、识别的头文件路径加入到项目工程属性管理器的 VC++ Directories/Include VC++ Directories（如出现无法找到的头文件问题，请确认这一步）。
3) **lib 文件引入：**将 lib 文件路径分别加入 VC++ Directories/Library VC++ Directories（如出现无法找的***.lib 问题，请确认这一步）；将.lib 文件的名字（如 VIPL_SDK_FD_DOG_Frontal.lib）加入到 Linker/Input/Additional Dependencies 中，每行一个，注意换行。

1) **dll 文件引入：**将 dll 文件复制到生成的 exe 文件夹下。（如出

现应用程序无法启动问题，请确认这一步）。

请注意区分 debug/release 版本的 lib 及 dll 文件。

(二) Opencv 环境配置

2) **头文件引入：**将 opencv 的头文件路径加入到项目工程属性管理器的 VC++ Directories/Include VC++ Directories;（如出现无法找到的头文件问题，请确认这一步）。
3) **lib 文件引入：**将 opencv 的 lib 文件路径分别加入 VC++ Directories/Library VC++ Directories;（如出现无法找的***.lib 问题，请确认这一步）；将.lib 文件的名字（如 opencv_world310.lib，注意通常以 d 结尾的文件为 debug 版）加入到 Linker/Input/Additional Dependencies 中，每行一个，注意换行。（如出现无法解决的外部依赖问题，请确认这一步）。
4) **dll 文件引入：**如已完成 opencv 的系统环境配置，将 opencv 的 dll 路径加入到系统变量中，不需要再次引入 dll。（如出现应用程序无法启动问题，请确认这一步）。

(三) 例程简介

1) 输入图像并显示，确保图像输入正确（请使用 cv::imread 函数读入图像，使用 cv::Mat 类，IPLImage 可能会导致无法检测到人脸）：

```
cv::Mat src1 = cv::imread("img1.jpg");
cv::Mat src2 = cv::imread("img2.jpg");
cv::Mat src_nomark1 = src1.clone();
cv::Mat src_nomark2 = src2.clone();
cv::namedWindow("ss1", 1);
cv::imshow("ss1", src1);
cv::waitKey(0);
cv::namedWindow("ss2", 1);
cv::imshow("ss2", src2);
cv::waitKey(0);
```

2) 对输入图像进行人脸检测并显示检测结果（参数设置请查阅参考文档）：

```
//参数分别为最小人脸，姿态、performance level 及speed level，具体设置方法请查看参考文档
//姿态中0表示准正面，1表示侧面，2 表示正面、侧面都检测，
//对于只有准正面的SDK这一参数不影响，设置为0即可
ImgProcessor.SetFDPara(20, 0, 4, 3);

ImgProcessor.GetFDResult(src1, facenum1, FDResult1);
ImgProcessor.GetFDResult(src2, facenum2, FDResult2);
if (facenum1 == 0 || facenum2 == 0)
    return 0;
ImgProcessor.DrawFDResult(src1, facenum1, FDResult1);
ImgProcessor.DrawFDResult(src2, facenum2, FDResult2);
cv::namedWindow("ss1", 1);
cv::imshow("ss1", src1);
cv::waitKey(0);
cv::namedWindow("ss2", 1);
cv::imshow("ss2", src2);
cv::waitKey(0);
```

3) 对输入图像进行特征点定位并显示结果：

```
ImgProcessor.GetPDResult(src1, facenum1, FDResult1, PDResult1);
ImgProcessor.GetPDResult(src2, facenum2, FDResult2, PDResult2);
ImgProcessor.DrawPDResult(src1, facenum1, FDResult1, PDResult1);
ImgProcessor.DrawPDResult(src2, facenum2, FDResult2, PDResult2);
cv::namedWindow("ss1", 1);
cv::imshow("ss1", src1);
cv::waitKey(0);
cv::namedWindow("ss2", 1);
cv::imshow("ss2", src2);
cv::waitKey(0);
```

4) 对输入图像进行人脸校正裁剪并输出结果：

```
float sim = 0;
float *feat1 = new float[featsize];
float *feat2 = new float[featsize];
cv::Mat CrFace1;
cv::Mat CrFace2;
ImgProcessor.GetCropFace(src_nomark1, CrFace1, PDResult1[0]);
ImgProcessor.GetCropFace(src_nomark2, CrFace2, PDResult2[0]);
ImgProcessor.ExtractFeature(CrFace1, feat1);
ImgProcessor.ExtractFeature(CrFace2, feat2);
cv::namedWindow("ss1", 1);
cv::imshow("ss1", CrFace1);
cv::imwrite("crface1.jpg", CrFace1);
cv::waitKey(0);
cv::namedWindow("ss2", 1);
cv::imshow("ss2", CrFace2);
cv::imwrite("crface2.jpg", CrFace2);
cv::waitKey(0);
```

5) 计算相似度并打印输出结果：

```
sim = ImgProcessor.CalCulateSim(feat1, feat2);
std::cout << sim << std::endl;
```