

gsoap 创建服务端、客户端进行 centerface 人脸检测

笔记本: gsoap

创建时间: 2019/12/11 10:09

更新时间: 2019/12/17 17:57

作者: 汪墩

URL: <https://www.cnblogs.com/liushui-sky/p/9723397.html>

概述

本文通过 gsoap 编写 webservice 服务端、客户端程序，在服务端利用 opencv_4.1.2 的 DNN模块加载 onnx 格式的 centerface 人脸检测模型，共提供两个接口函数，可分别接收 图片路径或base64格式的图片，人脸检测结果以结构体的形式返回。

一、gSOAP 下载

从官网下载[gsoap 2.8.96](#)

下载后解压即可，主要用到的文件有

gsoap-2.8/gsoap/bin/win64 目录下的 **soapcpp2.exe** 和 **wsdl2h.exe**，以及 gsoap-2.8/gsoap 目录下的 **stdsoap2.cpp** 和 **stdsoap2.h**，这些文件直接拿来用即可。

soapcpp2.exe 是一个根据.h文件生成若干支持webservice的代码生成工具，生成的代码文件包括webservice客户端和服务器的实现框架，XML数据绑定等。

wsdl2h.exe 可以根据输入的wsdl或XSD或URL，产生相应的C/C++形式的.h（不能直接引用），供soapcpp2使用。

二、编写 .h 接口头文件

接口头文件用于声明接口函数，每个函数前面需要以“ns__”开头，如以下 face.h 内容为

```
//gsoap ns service name: face
//gsoap ns service style:      rpc
//gsoap ns service encoding:    encoded

struct ns__SingleFaceInfo {
    float x1;
    float y1;
    float x2;
    float y2;
```

```

        float score;
        float landmarks[10];
    };
    struct FaceRes {
        int __size;
        struct ns__SingleFaceInfo **__ptr;
    };
    int ns__FaceDetectByImgpath(std::string img_path, struct FaceRes *face_res);
    int ns__FaceDetectByBase64 (std::string img_base64, struct FaceRes *face_res);

```

注意，最开始的注释文字是有用的，可以参考详细文档。

该接口文件只提供了两个接口函数，ns__FaceDetectByImgpath 是通过图片路径进行人脸检测，ns__FaceDetectByBase64 是通过 base64 格式的图片进行人脸检测。根据 gsoap 的要求，函数名前面必须以 “ns__” 开头，返回值必须为 int，但是这里的 int 并不是接口的返回值，而是 gsoap 内部的返回值。例如

```
int ns__add( int a, int b, int *c );
```

真正的返回值是 int *c。

同样，上述头文件中真正的返回值是

```
struct FaceRes *face_res。
```

根据 gsoap 规范，接口只能返回一个参数，如果有多个参数返回，需要用到结构体。

三、通过 .h 头文件生成服务端、客户端的相关依赖文件

建立一个 gsoaptest_vs2015_centerface 文件夹，分别将第一步提到的文件

- soapcpp2.exe
- wsdl2h.exe
- stdsoap2.cpp
- stdsoap2.h

以及第二步创建的

- face.h

文件复制到该 gsoaptest_vs2015_centerface 文件夹。

在 cmd 命令行窗口，移动到 gsoaptest_vs2015_centerface 文件所在目录，并执行以下命令

```

soapcpp2 -j -r -SL face.h
soapcpp2 -j -r -CL face.h

```

-j: 表示生成一个 c++ 代理类

-r: 表示生成一个报告

-SL: 表示只生成服务端的代码，而不会生成用不着的 lib 文件

-CL: 表示只生成客户端的代码，而不会生成用不着的 lib 文件

出现 compilation successful 表明生成成功，否则需要根据错误提示修改 face.h 内容。

生成后内容如下

新加卷 (F:) > gSOAP > gsoaptest_vs2015_centerface

名称	修改日期	类型	大小
face.FaceDetectByBase64.req.xml	2019/12/17 14:32	XML 文档	1 KB
face.FaceDetectByBase64.res.xml	2019/12/17 14:32	XML 文档	1 KB
face.FaceDetectByImgpath.req.xml	2019/12/17 14:32	XML 文档	1 KB
face.FaceDetectByImgpath.res.xml	2019/12/17 14:32	XML 文档	1 KB
face.nsmmap	2019/12/17 14:32	NSMAP 文件	1 KB
face.wsdl	2019/12/17 14:32	Web Service De...	5 KB
ns.xsd	2019/12/17 14:32	XML Schema File	2 KB
soapC.cpp	2019/12/17 14:32	C++ Source	100 KB
soapfaceProxy.cpp	2019/12/17 14:32	C++ Source	10 KB
soapfaceProxy.h	2019/12/17 14:32	C/C++ Header	7 KB
soapH.h	2019/12/17 14:32	C/C++ Header	71 KB
soapReadme.md	2019/12/17 14:32	Markdown File	43 KB
soapStub.h	2019/12/17 14:32	C/C++ Header	18 KB
soapfaceService.cpp	2019/12/16 19:56	C++ Source	11 KB
soapfaceService.h	2019/12/16 19:56	C/C++ Header	6 KB
soapcpp2.exe	2019/12/16 19:56	应用程序	849 KB
stdsoap2.cpp	2019/12/16 19:56	C++ Source	644 KB
stdsoap2.h	2019/12/16 19:56	C/C++ Header	156 KB
base64.cpp	2019/12/16 19:56	C++ Source	4 KB
base64.h	2019/12/16 19:56	C/C++ Header	1 KB
cv_dnn_centerface.cpp	2019/12/16 19:56	C++ Source	7 KB
cv_dnn_centerface.h	2019/12/16 19:56	C/C++ Header	2 KB
face - 副本.h	2019/12/16 19:56	C/C++ Header	1 KB
face.h	2019/12/16 19:56	C/C++ Header	1 KB
gsoap_use_wsdl	2019/12/17 15:33	文件夹	
plugin	2019/12/17 14:46	文件夹	
.git	2019/12/17 9:46	文件夹	
gsoaptest_vs2015	2019/12/16 19:56	文件夹	

四、建立VS2015工程

在 gsoaptest_vs2015_centerface 文件夹中新建 gsoaptest_vs2015 文件夹，在该文件夹建立vs2015工程，分别建立两个项目 gsoapServerTest 和 gsoapClientTest，均设置为 Release X64 模式。

4.1 gsoapServerTest 项目

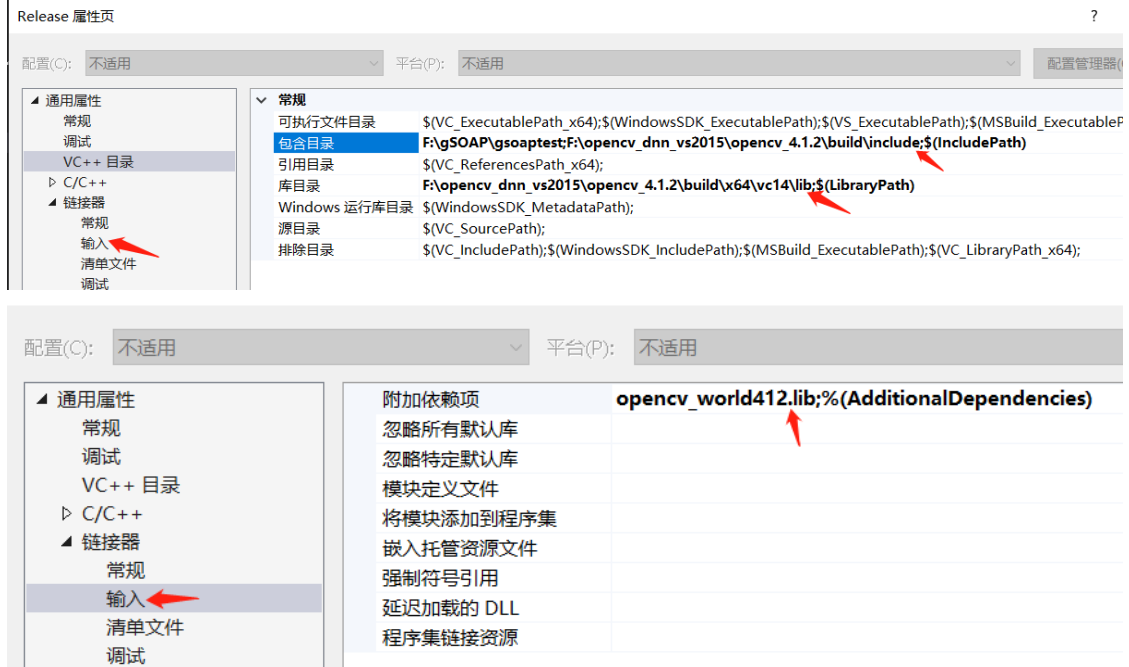
- 添加包含目录、配置opencv路径

在项目的包含目录中添加上述第三步生成文件所在的路径，用于添加相应的头文件及源文件。

包含目录

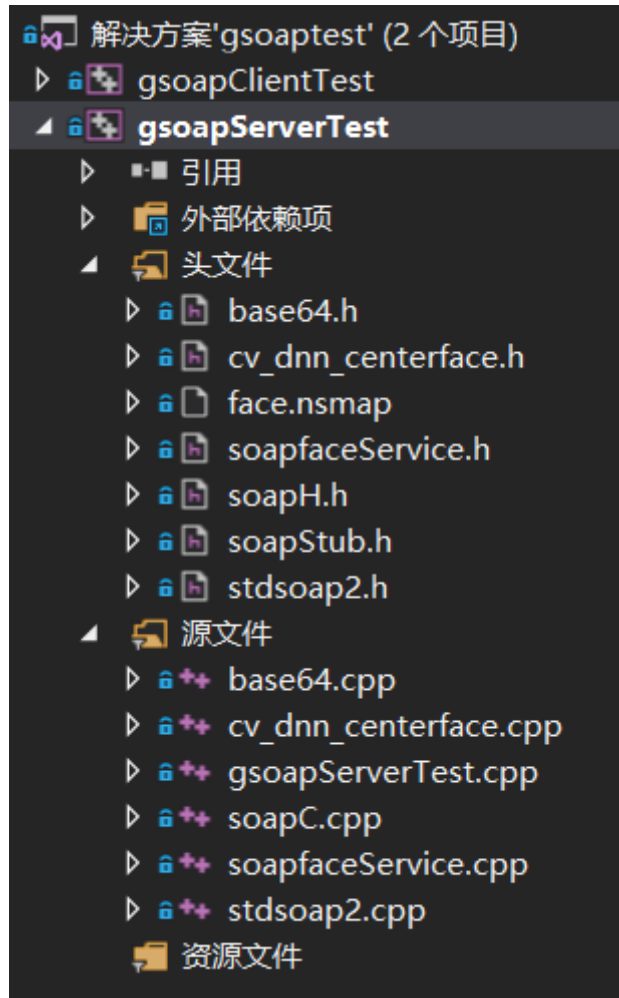
```
F:\gSOAP\gsoaptest_vs2015_centerface  
F:\opencv_dnn_vs2015\opencv_4.1.2\build\include
```

由于在人脸检测时，需要利用 opencv 的 dnn 模块加载 onnx 格式的模型，因此需要按照惯例分别添加 opencv_4.1.2 的包含目录、库目录和附加依赖项。



- 添加头文件、源文件

按照图示内容，添加位于包含目录中的相应头文件和源文件



其中各文件作用如下

- **第三方代码，用于 base64 图片转为 mat**
base64.h
base64.cpp
- **centerface相关代码**
cv_dnn_centerface.h
cv_dnn_centerface.cpp
- **上述第三步通过.h生成的**
face.nsmmap
soapfaceService.h
soapfaceService.cpp
soapH.h
soapStub.h
soapC.cpp
- **第一步提到的gsoap源码**
stdsoap2.h
stdsoap2.cpp

以上文件所有文件均位于添加的包含目录gsoaptest_vs2015_centerface 中，直接添加现有项即可。

- 自己新建的gsoap服务端生成程序
gsoapServerTest.cpp

在gsoapServerTest.cpp中对之前在头文件中声明的接口函数进行了具体实现

4.2 gsoapClientTest 项目

- 添加包含目录、配置opencv路径

在项目的包含目录中添加上述生成文件所在的路径，用于添加相应的头文件及源文件。

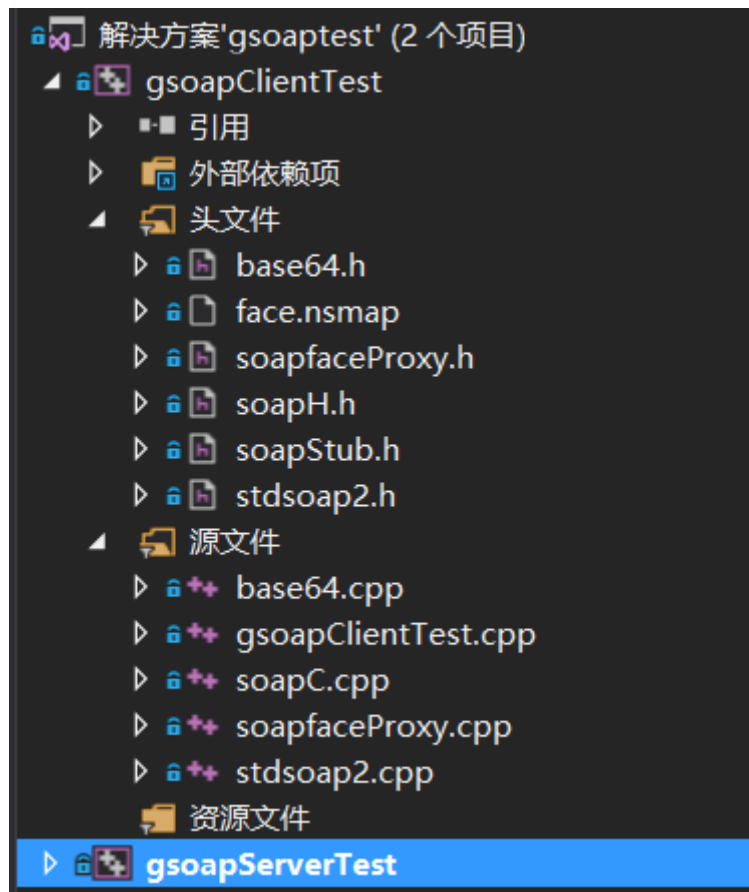
包含目录

```
F:\gSOAP\gsoaptest_vs2015_centerface  
F:\opencv_dnn_vs2015\opencv_4.1.2\build\include
```

在接收到接口函数传回的人脸检测结果后，为了直观，可以将检测结果绘制并显示出来，需要用到opencv来绘制结果，因此在客户端也需要按照惯例分别添加 opencv_4.1.2 的包含目录、库目录和附加依赖项。

- 添加头文件、源文件

按照图示内容，添加位于包含目录中的相应头文件和源文件



其中各文件作用如下

- **第三方代码，用于生成 base64 格式的测试图片**
base64.h
base64.cpp
- **上述第三步通过.h生成的**
face.nsmmap
soapfaceProxy.h
soapfaceProxy.cpp
soapH.h
soapStub.h
soapC.cpp
- **第一步提到的gsoap源码**
stdsoap2.h
stdsoap2.cpp














以上文件所有文件均位于添加的包含目录gsoaptest_vs2015_centerface 中，直接添加现有项即可。

- **自己新建的gsoap客户端生成程序**
gsoapClientTest.cpp

在gsoapClientTest.cpp中接收图片路径或base64格式图片，然后根据要求调用服务端的接口函数，以结构体的形式返回检测结果，然后再将检测结果绘制在原图上。

五、测试

在第四步建立好工程后，可以生成解决方案，将在 x64/Release 目录中生成服务端和客户端的可执行文件

  imgs	2019/12/17 15:05
 models	2019/12/17 9:14
 gsoapClientTest.exe	2019/12/17 16:10
 gsoapClientTest.iobj	2019/12/17 16:10
 gsoapClientTest.ipdb	2019/12/17 16:10
 gsoapClientTest.pdb	2019/12/17 16:10
 gsoapServerTest.exe	2019/12/17 17:00
 gsoapServerTest.iobj	2019/12/17 17:00
 gsoapServerTest.ipdb	2019/12/17 17:00
 gsoapServerTest.pdb	2019/12/17 17:00
 opencv_world412.dll	2019/12/16 19:56
 test.bat	2019/12/16 19:56

先点击 gsoapServerTest.exe ,出现以下界面

```
F:\gSOAP\gsoaptest\gsoaptest_vs2015\gsoaptest\x64\Release\gsoapServerTest.exe
Socket connection successful : master socket = 908
```

然后双击 test.bat 文件, 其内容为

```
gsoapClientTest.exe imgs/21.jpg
gsoapClientTest.exe imgs/19.jpg
gsoapClientTest.exe imgs/1.jpg

pause
```

运行结果

```
C:\WINDOWS\system32\cmd.exe

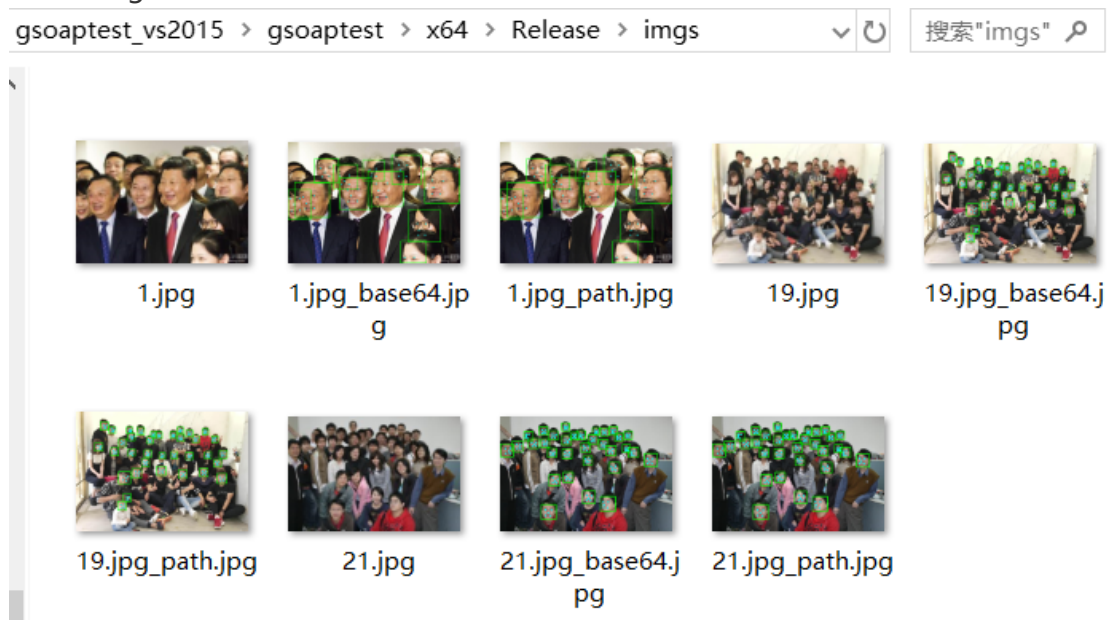
F:\gSOAP\gsoaptest_vs2015_centerface\gsoaptest_vs2015\gsoaptest\x64\Release>gsoapClientTest.exe imgs/21.jpg
The Client is runing...
FaceDetectByImgpath time is: 0.217s
FaceDetectByImgpath face_res.face_count = 32
FaceDetectByBase64 time is: 0.233s
FaceDetectByBase64 face_res.face_count = 32

F:\gSOAP\gsoaptest_vs2015_centerface\gsoaptest_vs2015\gsoaptest\x64\Release>gsoapClientTest.exe imgs/19.jpg
The Client is runing...
FaceDetectByImgpath time is: 0.341s
FaceDetectByImgpath face_res.face_count = 35
FaceDetectByBase64 time is: 0.314s
FaceDetectByBase64 face_res.face_count = 35

F:\gSOAP\gsoaptest_vs2015_centerface\gsoaptest_vs2015\gsoaptest\x64\Release>gsoapClientTest.exe imgs/1.jpg
The Client is runing...
FaceDetectByImgpath time is: 0.312s
FaceDetectByImgpath face_res.face_count = 11
FaceDetectByBase64 time is: 0.266s
FaceDetectByBase64 face_res.face_count = 11

F:\gSOAP\gsoaptest_vs2015_centerface\gsoaptest_vs2015\gsoaptest\x64\Release>pause
请按任意键继续. . .
```

同时在imgs文件夹, 将生成绘制的结果图片



六、通过 wsdl 文件建立测试客户端

上述第五步的客户端代码是基于 face.h 生成的，下面将基于 wsdl 生成客户端代码进行测试。

在上述第三步中，通过 .h 生成相关文件时，会同时生成 face.wsdl 文件，这是一个通用的 webservice 格式文件，不论什么语言，只要有了这个文件就能建立相应的客户端测试程序，调用远程服务端的接口函数，得到结果。

下面我们基于这个 wsdl 文件，通过 gsoap 来创建 cpp 的客户端测试程序。

6.1 通过 wsdl2h.exe 生成 .h 文件

建立一个文件夹 gsoap_use_wsdl，将上述的 face.wsdl 以及 wsdl2h.exe、soapcpp2.exe、stdsoap2.cpp、stdsoap2.h 复制过去

在 cmd 窗口移动到该目录，然后执行以下命令

```
wsdl2h -o face.h face.wsdl
```

意思是通过 face.wsdl 文件生成一个 face.h 头文件。

接下来使用这个生成的 face.h 头文件来创建相应的客户端依赖文件，其实跟上面第三、四步差不多，不同的是，要修改 gsoapClientTest.cpp 文件。

6.2 通过 .h 文件生成客户端需要的相关必要文件

执行以下命令

```
soapcpp2 -j -r -CL face.h
```

-j：表示生成一个 c++ 代理类

-r：表示生成一个报告

-CL：表示只生成客户端的代码，而不会生成用不着的 lib 文件

出现 compilation successful 表明生成成功。

6.3 建立 gsoap_client_cpptest 工程

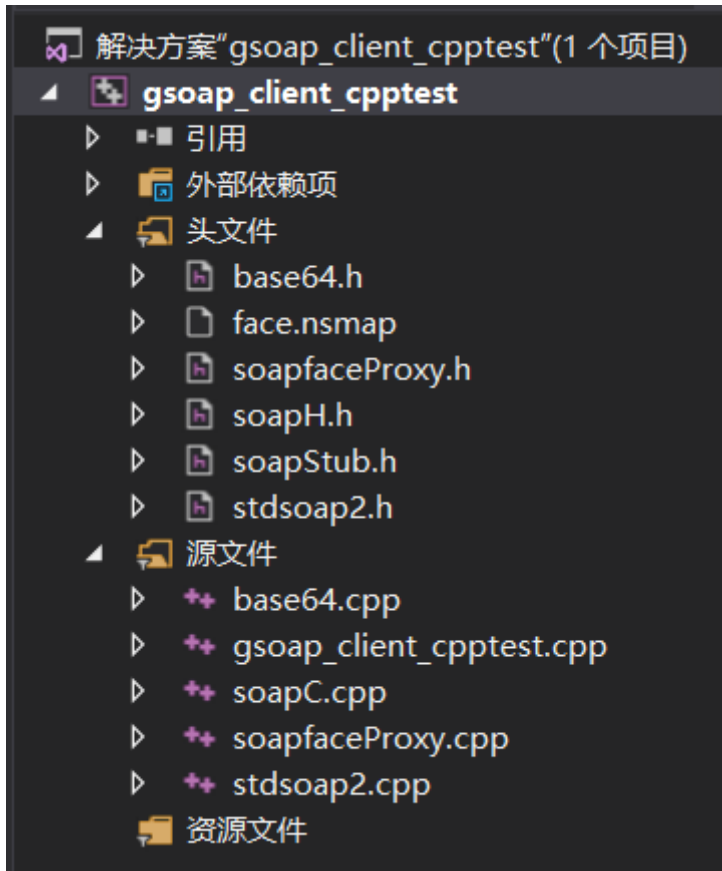
- 添加包含目录、配置 opencv 路径

在项目的包含目录中添加上述生成文件所在的路径 gsoap_use_wsdl，用于添加相应的头文件及源文件。

在接收到接口函数传回的人脸检测结果后，为了直观，可以将检测结果绘制并显示出来，需要用到 opencv 来绘制结果，因此在客户端也需要按照惯例分别添加 opencv_4.1.2 的包含目录、库目录和附加依赖项。

- 添加头文件、源文件

按照以下内容添加相应的头文件和源文件



其中各文件作用如下

- **第三方代码，用于生成 base64 格式的测试图片**
base64.h
base64.cpp
- **上述第三步通过.h生成的**
face.nsmap
soapfaceProxy.h
soapfaceProxy.cpp
soapH.h
soapStub.h
soapC.cpp
- **第一步提到的gsoap源码**
stdsoap2.h
stdsoap2.cpp

以上文件所有文件均位于添加的包含目录 gsoap_use_wsdl 中，直接添加现有项即可。

- **自己新建的gsoap客户端生成程序**
gsoap_client_cpptest.cpp

6.4 测试

创建好工程后，直接生成，将在 x64/Release 路径下生成可执行文件。
由于本次测试仅是创建客户端工程，因此需要事先开启相应的服务，即先点击上述第五步中的 gsoapServerTest.exe，服务正常开启后，再点击本工程中的test.bat文件，其内容如下

```
gsoap_client_cpptest.exe imgs/21.jpg
gsoap_client_cpptest.exe imgs/19.jpg
gsoap_client_cpptest.exe imgs/1.jpg

pause
```

电脑 > 新加卷 (F:) > gSOAP > gsoap_use_wsdl > gsoap_client_cpptest > x64 > Release				
<input type="checkbox"/>	名称	修改日期	类型	大小
	imgs	2019/12/16 19:38	文件夹	
	gsoap_client_cpptest.exe	2019/12/16 19:38	应用程序	175 KB
	gsoap_client_cpptest.iobj	2019/12/16 19:38	IOBJ 文件	1,709 KB
	gsoap_client_cpptest.ipdb	2019/12/16 19:38	IPDB 文件	1,478 KB
	gsoap_client_cpptest.pdb	2019/12/16 19:38	Program Debug ...	2,732 KB
	opencv_world412.dll	2019/10/10 7:58	应用程序扩展	57,854 KB
	test.bat	2019/12/16 13:28	Windows 批处理...	1 KB

将显示以下结果

```
F:\gSOAP\gsoap_use_wsdl\gsoap_client_cpptest\x64\Release>gsoap_client_cpptest.exe imgs/21.jpg
The Client is runing...
FaceDetectByImgpath time is: 0.179s
FaceDetectByImgpath face_res.face_count = 32
FaceDetectByBase64 time is: 0.175s
FaceDetectByBase64 face_res.face_count = 32

F:\gSOAP\gsoap_use_wsdl\gsoap_client_cpptest\x64\Release>gsoap_client_cpptest.exe imgs/19.jpg
The Client is runing...
FaceDetectByImgpath time is: 0.24s
FaceDetectByImgpath face_res.face_count = 35
FaceDetectByBase64 time is: 0.234s
FaceDetectByBase64 face_res.face_count = 35

F:\gSOAP\gsoap_use_wsdl\gsoap_client_cpptest\x64\Release>gsoap_client_cpptest.exe imgs/1.jpg
The Client is runing...
FaceDetectByImgpath time is: 0.265s
FaceDetectByImgpath face_res.face_count = 11
FaceDetectByBase64 time is: 0.259s
FaceDetectByBase64 face_res.face_count = 11

F:\gSOAP\gsoap_use_wsdl\gsoap_client_cpptest\x64\Release>pause
请按任意键继续. . .
```

同时也将在imgs目录下生成相应的结果图片



七、在服务器部署服务程序，在局域网的其他电脑作为客户端测试

上述第一步至第六步中，服务端和客户端程序都部署在同一台电脑上。但这其实不是gsoap的初衷，gsoap还可以在功能强大的服务器上部署服务端程序，然后在局域网的其他电脑上进行客户端调用。

步骤如下：

- 1、按照上述第一至第五步，在 **服务器** 上部署服务端代码；
- 2、按照上述第一至第五步，在 **同一局域网的其他电脑上** 部署客户端程序，仅仅需要将客户端测试代码 gsoapClientTest.cpp 中的 server 地址由 localhost 改为 **服务器的IP地址**，如

```
//const char server[] = "http://localhost:8080";  
const char server[] = "http://192.168.0.35:8080";
```

- 3、在服务器开启服务；
- 4、在客户端电脑开启测试程序进行测试；

鸣谢：

[基于gsoap开发WebService服务返回结构体数组](#)

[gsoap传递数组方法](#)

[gsoap创建webservice服务简单教程](#)