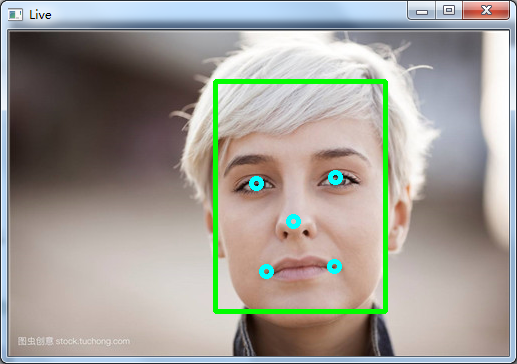
# MTCNN人脸检测框架的配置与测试

MTCNN是一个高效的人脸检测框架，因为在初次配置过程中，遇到了很多问题，最后配置并测试成功，记录下配置过程，以供以后参考。



## 1.下载并编译Caffe

下载Windows版本Caffe，地址：https://github.com/BVLC/caffe/tree/windows

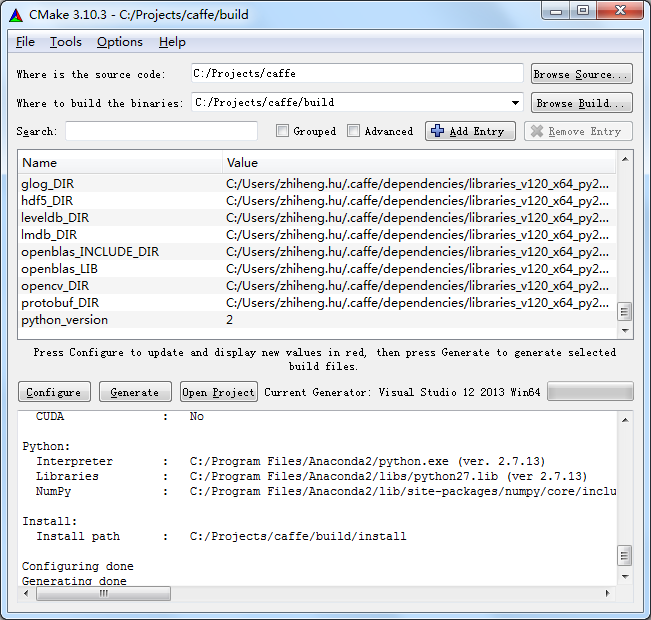
用CMake编译Caffe，编译器我使用的是Visual Studio 2013 x64，首次编译时提示下载libraries\_v120\_x64\_py27\_1.1.0.tar.bz2文件，并很可能失败，可以自己去以下网址下载：https://github.com/willyd/caffe-builder/releases，并拷贝到C:\Users\zhiheng.hu\.caffe\dependencies\download目录下。

再次编译，可能还会遇到Altas错误，把BLAS选项改成Open。

勾选CPU\_ONLY，并去掉与CUDA相关的勾选标记。

再次编译，即可成功，若还不行，重启CMake再编译。

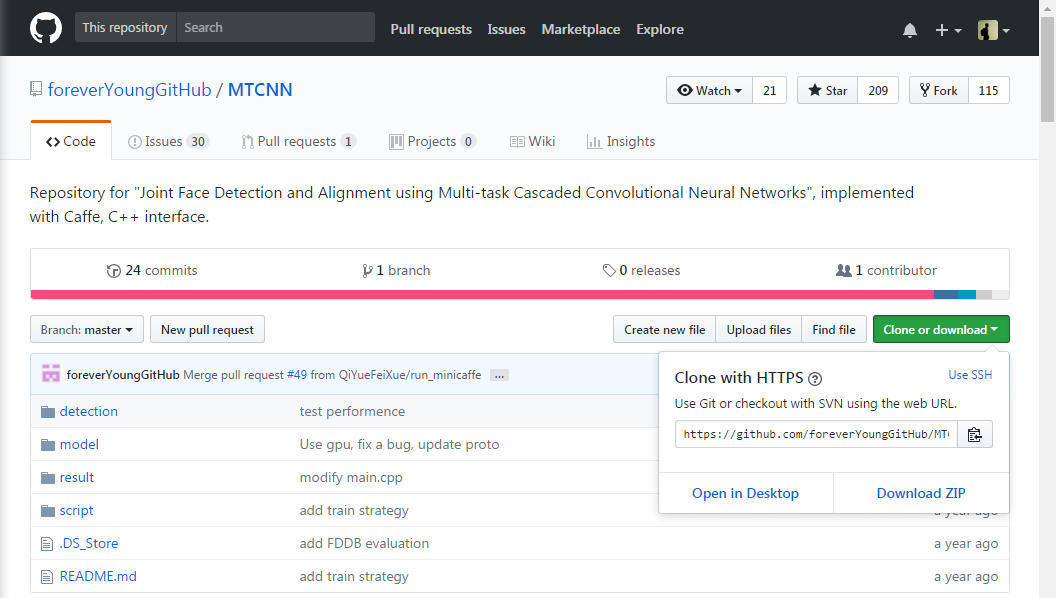
使用Visual Studio 2013打开build目录下Caffe.sln，对INSTALL重新生成即可。



## 2.编译MTCNN工程

因为不熟悉工程配置，我直接使用Caffe自带的工程修改成MTCNN工程。

下载MTCNN仓库，地址：https://github.com/foreverYoungGitHub/MTCNN

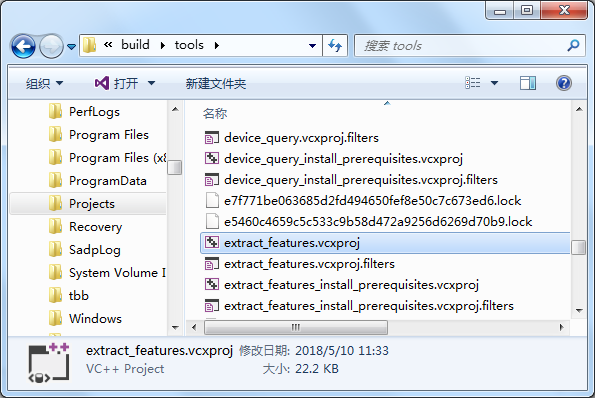


解压得到：

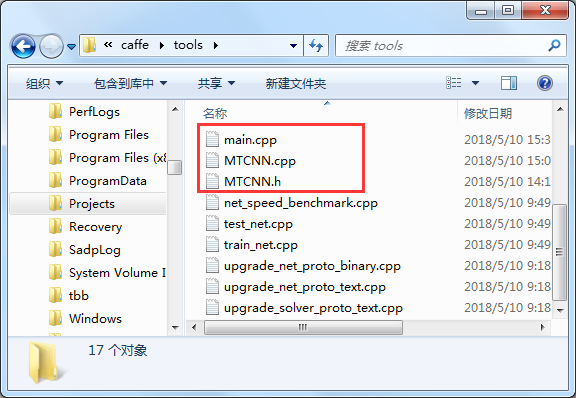


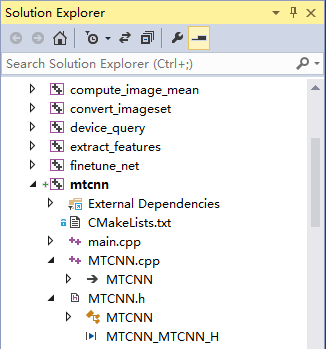
代码在detection目录中，模型在model目录中。

我复制了一份extract\_feature.vcxproj文件，并重命名为mtcnn.vcxproj，添加到工程中，添加到工程后，工程名称还是extract\_feature，将其重命名为mtcnn。



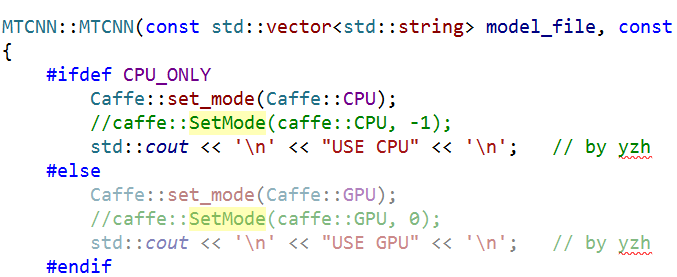
将MTCNN的代码拷贝到caffe/tools目录中，并添加到mtcnn工程中：



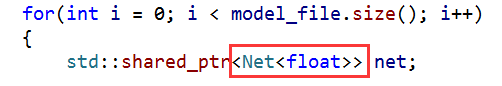


编译该工程，会出现很多错误，几个关键要修改的地方如下：

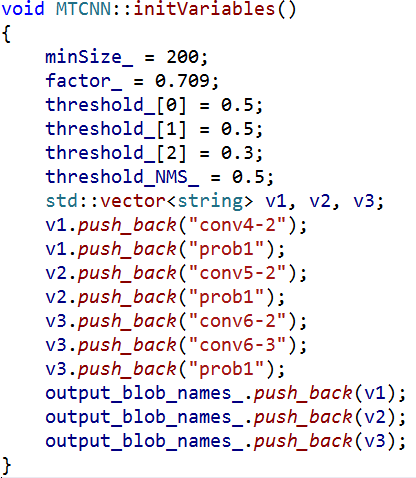
1) 工程中caffe::set\_mode(caffe::CPU, -1)该成Caffe::set\_mode(Caffe::CPU)



2）所有设计到Net，Blob类的定义都加上<float>以实例化这些类，不知道作者为何没有实例化。



3）因为工程中的变量的一些定义和初始化，在Visual Studio中不符合规范，增加一个初始化变量的函数，并在构造函数中调用：

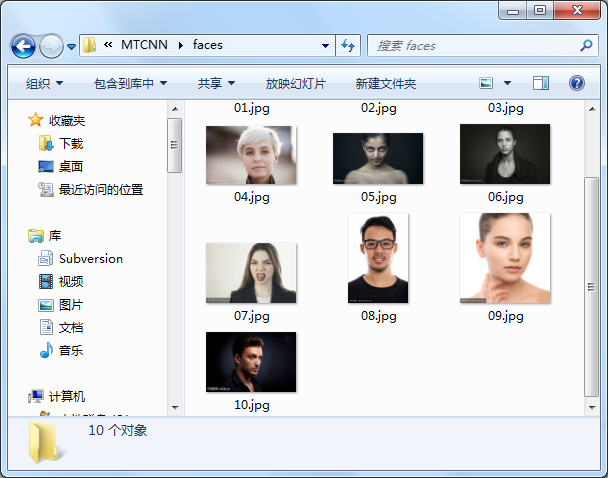


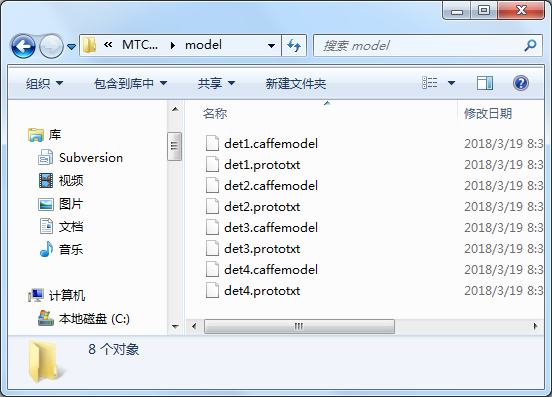
4) 因为工程中使用了drient.h头文件完成系统的文件目录操作，由于drient.h在windows下不存在，所以我改成opencv的glob函数来完成类似的功能，我将main函数修改如下：

|  |
| --- |
| #include <iostream>  #include "MTCNN.h"  #include "opencv2/opencv.hpp"  #include <time.h>  #include <sys/types.h>  #include <cstring>  using namespace std;  using namespace cv;  int main() {  vector<string> model\_file = {  "E:/MTCNN/model/det1.prototxt",  "E:/MTCNN/model/det2.prototxt",  "E:/MTCNN/model/det3.prototxt"  };  vector<string> trained\_file = {  "E:/MTCNN/model/det1.caffemodel",  "E:/MTCNN/model/det2.caffemodel",  "E:/MTCNN/model/det3.caffemodel"  };  MTCNN mtcnn(model\_file, trained\_file);  vector<String> files;  glob("E:/MTCNN/faces/\*.jpg", files);  for (int i = 0; i < files.size(); i++)  {  Mat img = imread(files[i]);  vector<Rect> rectangles;  vector<float> confidences;  std::vector<std::vector<cv::Point>> alignment;  mtcnn.detection(img, rectangles, confidences, alignment);  for (int i = 0; i < rectangles.*size*(); i++)  {  int green = confidences[i] \* 255;  int red = (1 - confidences[i]) \* 255;  rectangle(img, rectangles[i], cv::Scalar(0, green, red), 3);  for (int j = 0; j < alignment[i].*size*(); j++)  {  cv::circle(img, alignment[i][j], 5, cv::Scalar(255, 255, 0), 3);  }  }  imshow("Live", img);  waitKey(0);  }  return 0;  } |

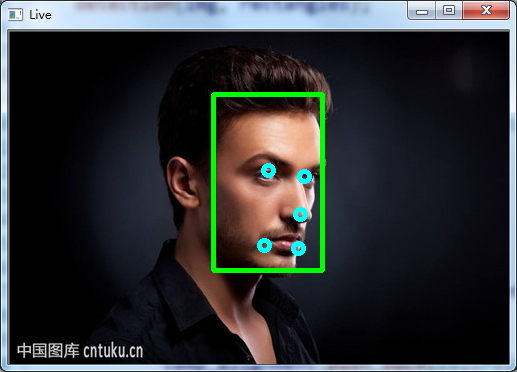
### 3.测试

准备图像和模型：





运行：



## 3.完整的MTCNN代码

见https://github.com/hy17003/MTCNN