入职事项

ftp:

username: smbuser

passwd: introcks

 默认NAT模式，用sudo dhclient eth0更新IP地址后解决ubuntu虚拟机网络问题。

安装openCV：

1）Required Packages

[compiler] sudo apt-get install build-essential //编译器

[required] sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev //依赖包

[optional] sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev //可选

2）下载openCv 源码

https://github.com/opencv/opencv

ex:

cd ~/<my\_working\_directory>

git clone https://github.com/opencv/opencv.git

git clone https://github.com/opencv/opencv\_contrib.git

3)sudo make

4)sudo make install

192.168.2.19 intellif/introcks

git reset --hard HEAD^

git add //提交到暂存区

gti commit //一次性把暂存区的所有修改提交到分支

git checkout -- file//回退文件修改。目前本机git版本好像有个问题，需要先git status查看下，然后再用checkout才生效回退

ssh-keygen -t rsa -C "youremail@example.com" //命令行产生公钥和私钥，可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件，这两个就是SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥。将公钥写到github的SShkey中

要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.git；

关联后，使用命令git push -u origin master第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

git remote add origin git@github.com:yanjingcheng/test.git

git push -u origin master

***//ssh内的rsa名字是id\_rsa,如果非要增加其他名字，可以用以下2条命令***

***ssh-agent bash***

ssh-add ~/.ssh/test\_rsa

https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013752340242354807e192f02a44359908df8a5643103a000

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

切换分支：git checkout <name>

合并某分支到当前分支：git merge <name>

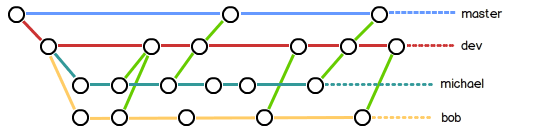
删除分支：git branch -d <name>

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

用git log --graph命令可以看到分支合并图

git merge --no-ff -m "merge with no-ff" dev //禁止快速合并，保留dev分支

团队合作分支，大家在dev分支工作，同时保留各自的开发分支



win7 64bit + VS2013 + Cmake3.11.0 + opencv3.2.0配置、编译生成opencv的lib库以及.dll文件

网上好多教程，只对本次配置的细节问题讲述：

编译好之后跑新建工程这段小程序，始终出问题，主要问题应该是包含的头文件opencv.hpp找不到以及imread()函数找不到。

#include <opencv2/opencv.hpp>

using namespace cv;

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

{

Mat img = imread("Desert.jpg");

imshow("img", img);

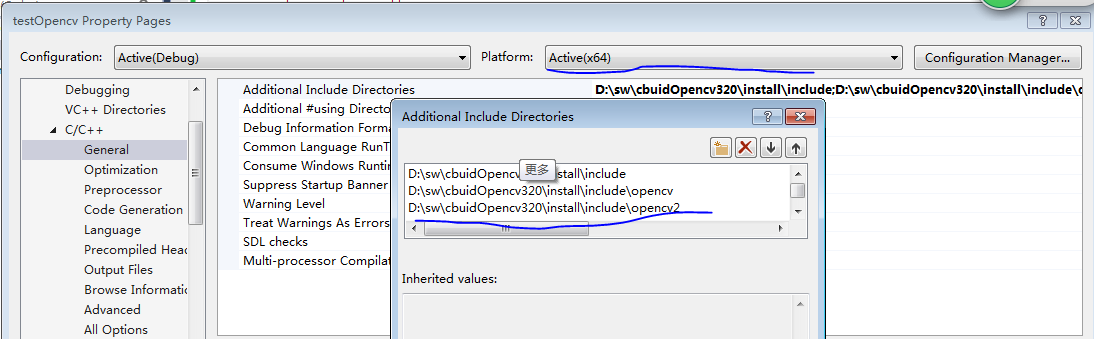
waitKey(0);

return 0;

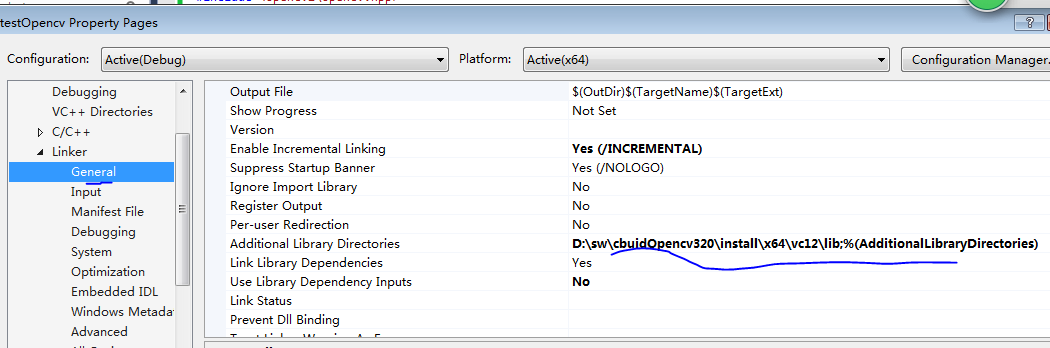
}

主要是VS工程配置问题，需要配置

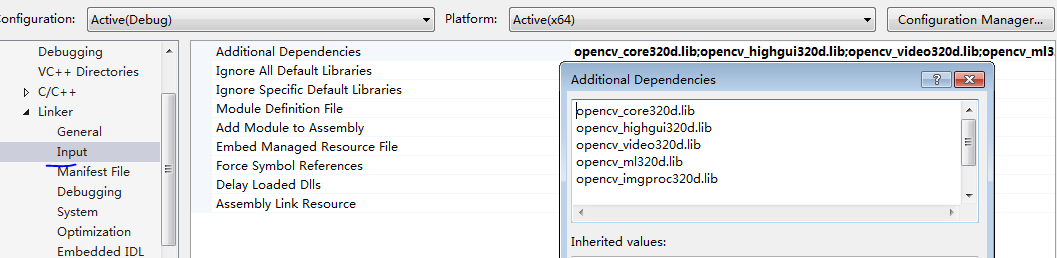
1)vc++路径



2）通用链接库路径



3）链接库input->额外依赖（这个很重要）



主要依赖文件如下：

opencv\_core320d.lib

opencv\_highgui320d.lib

opencv\_video320d.lib

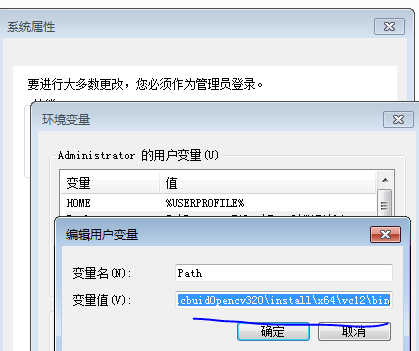
opencv\_ml320d.lib

opencv\_imgproc320d.lib

opencv\_photo320d.lib

opencv\_imgcodecs320d.lib //当时这个文件没有添加 折腾了许久

此外环境变量的设置没有生效，因此将上述.lib对应得opencv\_core320d.dll等需要的动态链接库拷贝到C:/windows/system32下。



后面就OK了。

因需要把YUV数据转换成JPG图像，因此需要增加个JPEG库，

首先是下载源码jpeg-9a.zipz windows版，需要vc编译器，将!include < win32.mak>改为绝对路劲

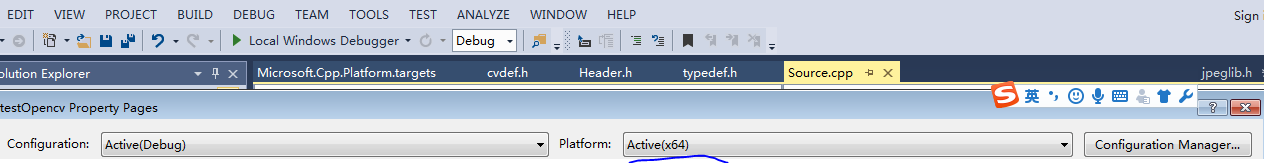
!include <C:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Include\win32.mak>

1)在 jpeg-9a目录下，找到makefile.vc文件，找到前面几行中将

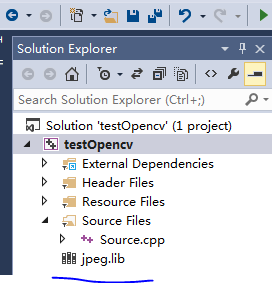
2)找到D:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\Tools\Shortcuts\ VS2013 x64 Native Tools Command Prompt.exe。

然后cd切换到jpeg-9a目录下，执行:nmake /f makefile.vc setup-v10 (具体看makefile.vc有没有setup-v10选项内容)

3）编译好生产sln文件，用VS打开后修改平台为x64，保持跟现有平台一致



4）执行上面opencv 1\2\3步骤，还需要在工程中添加jpeg.lib文件



Build成功！折腾一天了！！fuck

MTCNN:

1.Pyramid

-->Layer分层，大图有scale不断缩放

Linux(readhat ununtu)下的opencv库安装及支持ffmpeg的视频解码

安装

首先安装yasm编译器。安装方法如下：

在http://www.tortall.net/projects/yasm/releases下面找到适合自己平台的yasm版本。然后进行安装。举例如下：

1）下载：wget http://www.tortall.net/projects/yasm/releases/yasm-1.3.0.tar.gz

2）解压：tar zxvf yasm-1.3.0.tar.gz

3）切换路径： cd yasm-1.3.0

4）执行配置： ./configure

5）编译：make

6）安装：make install

1、获得ffmpeg：

下载ffmpeg-2.7.7 (这个版本问题折腾好久，自己尝试的，版本不匹配的话，opencv2.3.14 make时会报错！！！！安装新版本之前，一定要把旧版本卸掉)

2、配置ffmpeg,进入ffmpeg目录，

2.1 ./configure --enable-shared --prefix=/usr

2.2 make (注意：如果编译FFMPEG时，出现了 ffmpeg yasm not found, use --disable-yasm for a crippled build； sudo apt-get install yasm )

2.3 make install

ffmpeg 测试下是否成功安装，如果失败：

此情况是因为ffmpeg未找到相关依赖包，可以通过以下方法解决

先　find / -name libavdevice.so.56 得到该文件的目录地址，

然后 vim /etc/ld.so.conf 将上述目录添加到最后一行并保存退出；

再 ldconfig 使配置生效，即可；

如果执行 ldconfig: command not foun

3、接下来要做的工作就是关系到opencv能否成功将ffmpeg的相关头文件编译进来的问题了：

3.1）、在/usr/include下执行以下命令（root）：

mkdir ffmpeg;

3.2） 在ffmepeg-2.7.7目录下

cp -rf libav\* libsw\* usr/include/ffmpeg; //之所以这么做，是因为opencv configure时检查是否已安装ffmpeg是在这检查的

4.安装opencv2.4.13

4.1 opencv sources/目录下

cmake -D CMAKE\_BUILD\_TYPE=Release -D CMAKE\_INSTALL\_PREFIX=/usr/local .. //注意这里OPECV默认安装在/usr/local，因此lib和include都会在此目录下，

可以根据需要指定自己的目录，特别是当系统的其它用户已经安装了此软件，为了不影响他人的使用，最后重新建一个自己的路径。还好我使用

的机器上虽然已经有其它人编译安装的opencv，但是都在系统默认目录/usr/lib和/usr/include.应该和我的没有交叉。

//如果以上步骤都没错的话，可以看到 opencv的configure结果中ffmpeg为YES，说明我们成功了。

sudo make install

注意：

在编译opencv LINK的时候如下报错：

OpenCV compile error Linking CXX shared library ../../lib/libopencv\_highgui.so /lib/libbz2.so.1:

could not read symbols: File in wrong format

网友答复：

The problem is that cmake finds a 32-bit library rather than the 64-bit one,

which should be found in /lib64/libbz2.so or /usr/lib64/libbz2.so. Sometimes the cmake

get messed up and keep using the cached variable.

So if you remove the cache and cmake again, it should find the correct library、

意思是在使用libbz2.so共享库的过程中， 脚本程序在/usr/lib和/usr/lib64 2处都找到了libbz2.so，CMAKE真正需要的是在lib64下libbz2，结果

在/usr/lib的下libbz2，因此CMAKE懵逼了。

系统默认先在/usr/lib查找，然后在/usr/lib64

解决方法：直接将/usr/lib下的libbz2.so删除，不让CMAKE找到，让它去/usr/lib64找。

最后的方法是在脚本中，调整lib的先后顺序，时间紧迫，没有去CMakeList.txt中慢慢看。

程序需要的是在库路径/LIB64/下的64-bit的工具

5、将/usr/local/lib/pkgconfig下的opencv.pc cp到/usr/lib/pkgconfig下，这一步也可以做，只是一个配置文件，大不了编译时自己指定头文件和库么，其实你打开opencv.pc一看就知道是什么玩意了。

//此前版本存在有lib64，为了新编译的版本不连接到lib64上 而是/usr/local/lib上

REDHAT cp /usr/local/lib/pkgconfig/opencv.pc /usr/lib64/pkgconfig/

6、root下， vim /etc/ld.so.conf, 增加一行， /usr/local/lib (因为opencv的库文件在local,默认情况下local/lib中的库文件是不在系统启动时被load的)

7、ldconfig

//pkg-config能找到ffmpeg的\*.pc文件，通过PKG\_CONFIG\_PATH 来设置pkg-config来

!!!export PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/usr/local/lib/pkgconfig

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:usr/lib/

Linux中error while loading shared libraries错误解决办法

　　默认情况下，编译器只会使用/lib和/usr/lib这两个目录下的库文件，通常通过源码包进行安装时，如果不指定--prefix，会将库安装在/usr/local/lib目录下；当运行程序需要链接动态库时，提示找不到相关的.so库，会报错。也就是说，/usr/local/lib目录不在系统默认的库搜索目录中，需要将目录加进去。

　　1、首先打开/etc/ld.so.conf文件

　　2、加入动态库文件所在的目录：执行vi /etc/ld.so.conf，在"include ld.so.conf.d/\*.conf"下方增加"/usr/local/lib"。

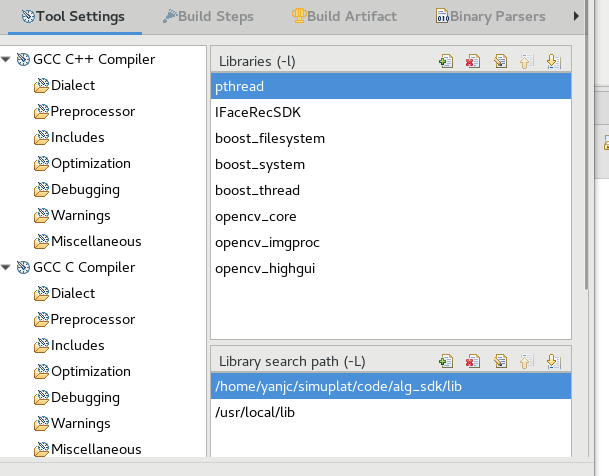
　　3、保存后，在命令行终端执行：/sbin/ldconfig -v；其作用是将文件/etc/ld.so.conf列出的路径下的库文件缓存到/etc/ld.so.cache以供使用，因此当安装完一些库文件，或者修改/etc/ld.so.conf增加了库的新搜索路径，需要运行一下ldconfig，使所有的库文件都被缓存到文件/etc/ld.so.cache中，如果没做，可能会找不到刚安装的库。

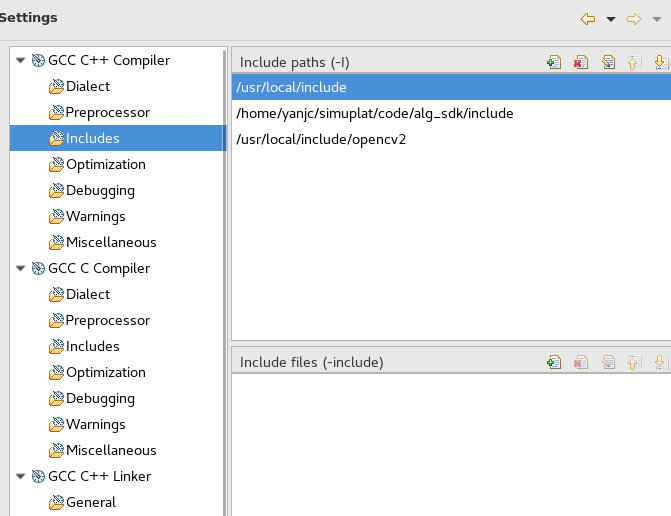
　　经过以上三个步骤，"error while loading shared libraries"的问题通常情况下就可以解决了。

　　如果运行应用程序时，还是提示以上错误，那就得确认一下是不是当前用户在库目录下是不是没有可读的权限。像我遇到的问题就是，从别的机子拷贝了一些.so动态库，然后用root权限放到了/usr/local/lib中（普通用户没有对该目录的写权限），然后切换用户运行程序时，始终提示找不到.so库，一直以为是我配置有问题，结果是因为权限原因，那些我用root权限增加到/usr/local/lib中的.so文件对于普通用户而言，是没有访问权限的，所以以普通用户运行程序，当需要链接.so库时，在/usr/local/lib中是查找不到的。

　　其实，对于由普通用户自己编译生成的.so库文件，比较好做好是将这些.so库文件的路径用export指令加入到~/.bash\_profile中的LD\_LIBRARY\_PATH变量中,LD\_LIBRARY\_PATH是程序运行需要链接.so库时会去查找的一个目录，~/.bash\_profile是登陆或打开shell时会读取的文件，这样，每次用户登录时，都会把这些.so库文件的路径写入LD\_LIBRARY\_PATH，这样就可以正常地使用这些.so库文件了。

DEBUG版本配置：





！