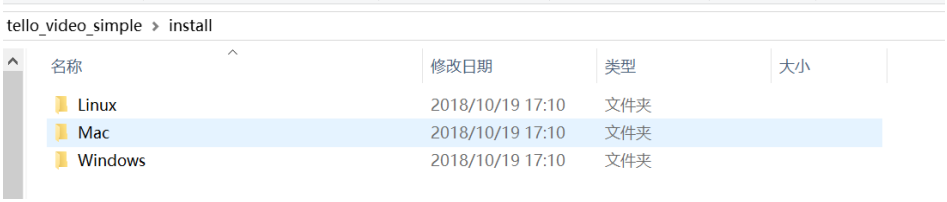
1、 安装各种第三方库：这套程序基于 python2，并依赖包括 boost-python，ffmpeg,numpy,PIL,opencv 等第三方库。为方便安装，编写了支持三种操作系统(windows32/64),Linux,macos 的一键安装第三方库的脚本。如下：



根据你操作系统的版本，选择对应的 install 脚本，将其复制到 main.py 所在的路径下，然后运行该文件，即可完成包括 python，pip 及诸多第三方库的安装。

各个第三方库简要说明:

pip：用于协助安装第三方库，在 macos 中还使用了 homebrew

boost:调用了 boost 库中的 boost-python，用于实现基于 c++的 libh264decoder 与 python 主程序的交互

ffmpeg:用于被 libh264decoder 调用，进行 h264 解码

numpy,:进行图像二维矩阵运算

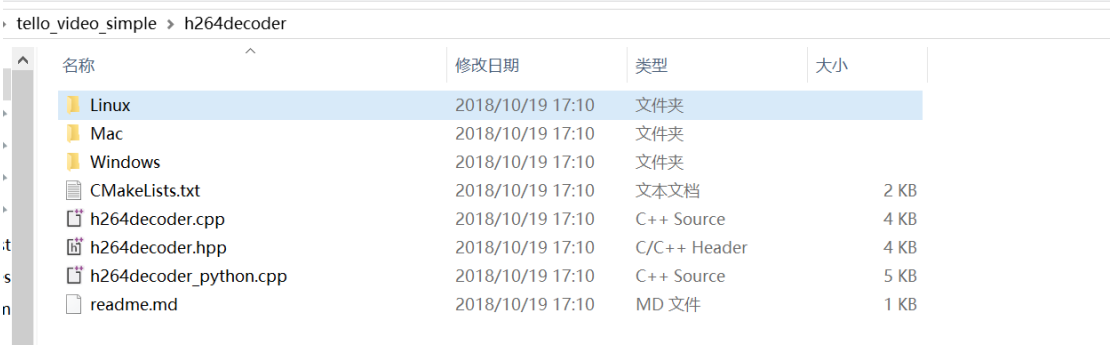
opencv：进行图像处理

PIL,Tkinter：实现 GUI 图像显示和飞机控制面板

cmake：编译 h264decoder 源码，生成动态链接库

2、 运行:连接 tello 的 wifi，用 python 运行 main.py 文件即可。

3、 H264 解码：视频流的解码通过调用动态链接库 libh264decoder 实现，文件夹中包含三种操作系统下编译好的 Libh264decoder，及其源代码，如图：



4、 Libh264 动态链接库的内部实现:如果要运行代码，只需要将对应的动态链接库复制到 main.py 路径下即可（windows 系统则需要将动态链接库移动到 python 文件所在路径下的 site-package 文件夹中）该动态链接库是由 h264decoder.cpp,h264decoder.hpp 和 h264decoder_python.cpp 编译得到。若想要参考 h264 解码的实现，可参考源代码。该源码通过 c++实现，通过 boost-python 实现与 python 的交互。在 linux 和 mac 下，该代码可直接在其路径下编译，若在 windows 系统下，需要配置相关的静态依赖项，可用 visual studio 进行编译。

5、 Python 程序中对 libh264 的调用：

在 tello.py 中，文件中需要进行 import：

```
import numpy as np
import libh264decoder
```

通过 UDP 收听端口号为 11111 的数据流，该数据流即为视频流数据，采用的是 h264 编

码，tello 发送的视频流中，由于一帧图像编码后的大小大于单次 udp 传输的最大载荷，因此每帧图像都进行了分割，以 1460 字节作为单位进行分割，分割的剩余部分小于 1460 字节。

因此，通过检测收听的数据块大小可判定是否该数据块为当前单桢图像的数据末尾，并将前面收听的数据块按顺序进行拼接，即可形成一个完整的单桢的 h264 编码数据。并将该数据传入 libh264decoder 中的 H264Decoder 函数，即可得到一桢图像。

截取 tello.py 中的相关代码如下：

```
54          self.decoder = libh264decoder.H264Decoder()
```

```
120     def _receive_video_thread(self):
121         """
122         Listens for video streaming (raw h264) from the Tello.
123 
124         Runs as a thread, sets self.frame to the most recent frame Tello captured.
125 
126         """
127         packet_data = ""
128         while True:
129             try:
130                 res_string, ip = self.socket_video.recvfrom(2048)
131                 packet_data += res_string
132                 # end of frame
133                 if len(res_string) != 1460:
134                     for frame in self._h264_decode(packet_data):
135                         self.frame = frame
136                     packet_data = ""
137 
138             except socket.error as exc:
139                 print ("Caught exception socket.error : %s" % exc)
140 
```

```
141     def _h264_decode(self, packet_data):
142         """
143         decode raw h264 format data from Tello
144 
145         :param packet_data: raw h264 data array
146 
147         :return: a list of decoded frame
148         """
149         res_frame_list = []
150         frames = self.decoder.decode(packet_data)
151         for framedata in frames:
152             (frame, w, h, ls) = framedata
153             if frame is not None:
154                 # print 'frame size %i bytes, w %i, h %i, linesize %i' % (len(frame), w, h, ls)
155 
156                 frame = np.fromstring(frame, dtype=np.ubyte, count=len(frame), sep='')
157                 frame = (frame.reshape((h, ls / 3, 3)))
158                 frame = frame[:, :w, :]
159                 res_frame_list.append(frame)
160 
161         return res_frame_list
```

6、 对解码后单帧图像的调用方式可参考 tello.py 中函数：

```
def read(self):
    """Return the last frame from camera."""
    if self.is_freeze:
        return self.last_frame
    else:
        return self.frame
```

在 tello_control_ui.py 中被调用：