FLEXIPILOT 1.35

Software installation, configuration and data processing

The document discusses installation of the autopilot support software in various scenarios, autopilot data logging and operating system setup.

Overview

The autopilot uses USB-A connector for communicating with a PC computer. If focuses on integration with existing, in most cases free software. Because the autopilot behaves as a virtual serial port (COM), with any operating system that has Text Mode Serial Console software it is possible to set up the autopilot on the field. As long as USB drivers are installed, any computer you find available for changing flight plans or retrieving the data. Therefore, should you computer fail in the field, you can revert any other machine available; thanks to the use of relative waypoint coordinates you can adjust the flight plane using local paper maps and distances in km.
The software is delivered in loosely coupled form, as you need literally a few independent executable files for managing the setup and processing the results. This way in case of migration to another computer, it is sufficient to carry the software on a removable media.

Data storage

FLEXIPILOT data storage system has the following characteristics:

- Self-tuning IMU, self-calibrating barometric altimeter,
  stores auto-trimmed control surfaces positions based on manual flight
- All variables editable by name with command-line prompt built-in into the autopilot
- Number of **3D waypoints**: More than 1024
  up to 1500, variable, depending on firmware size
- Realtime log (**LOG**): 4.125MB
  Up to 20Hz, selectable frequency, 120 variables
  139minutes @2Hz or 14min@20Hz
- Event log (**TRACE**): 8192 events
  flight time;latitiude; longitude;roll angle; pitch angle; AGL altitude;course; eventID
  size dependent on the max number of waypoints, periodic logging also possible at 20Hz max, recommended rate 0.5-1Hz
- Fast data transfer at 1Mb/s plus compression via USB port

Requirements

Any modern operating system is sufficient for basic connectivity using serial console if the supporting **FTDI232 drivers** are available.
The current list includes according to the manufacturer's website includes:
Windows Vista x64
Windows XP x64
Windows Server 2003 x64
Windows Vista
Windows XP
Windows Server 2003
Windows 2000
Windows ME
Windows 98
Linux
Mac OS X
Mac OS 9
Mac OS 8
Windows CE.NET (Version 4.2 and greater)
The systems listed in bold have been positively tested for compatibility.
The autopilot uses fixed 921600 baud, 8N1 (8 data bits, 1 stop bit, no parity, no flow control).

Enhanced support for Windows operating system is provided with UAVStation simulation and display console, which offers extended functionality to the serial port console, including:
-automatic PnP detection of the autopilot
-additional graphics window with simulation output
-additional graphics window with IMU and diagnostics output
-GoogleEarth integration

Obtaining the software

The software accompanying the autopilot includes:
- **FTDI 232RL drivers**, executable installer and separate ini –based drivers for windows, (recent versions can be downloaded from FTDI website)
- **UAVStation.exe** OpenGL based system console by AerialRobotics
-command-line tools facilitating serial port management:
logread921K.exe, **writeconfig921K.exe** and batch script for mass download
- **ASCII setup file *.txt** contains all the setup variables for your specific installation of the autopilot
- **flexipilot.sci** Scilab data analysis script
- **comdisable.msi** for disabling windows (operating systems antifeature) erroneously detecting serial mouse when using FTDI chip and transmitting GPS data
- **UAVStation - position.kml** and **UAVStation – position-chase.kml** are GoogleEarth files
- **logdecode.exe** automatic data export to Google Earth and other formats

Consider familiarizing yourself with the following software:
- **HyperTerminal** serial port terminal (can be found in Menu
Start/Accessories/Communication menu, except in Windows Vista)
You can copy the executable from another windows version if you have one, or download from the web – this is the most compatible terminal software.
- **TeraTerm** freeware serial port terminal
- **PuTTy** freeware serial port terminal (note mass pasting configuration is not possible because of lacking *line delay =10ms* option)
- GoogleEarth used for mission planning, waypoint display and real-time simulation
- GPicSync for geomapping the images taken with your digital camera for viewing with GoogleEarth application
- Scilab free mathematical analysis and scientific plotting software, only for viewing log plots (could be done with spreadsheet as well)
- **GPSMapEdit** using Garmin MapSource maps for mission simulation and planning
- Google Maps **GPSTracker** for mission simulation
- GeoSetter free geotagging software
- **C_GPS2KML** converter from GPS files to Google Earth
- **WinMerge** freeware file change manager
- **Beyond Compare** commercial file compare for managing a wide set of settings and several autopilots
- **Microsoft ICE Image Composite Editor** free panorama stitching software
- **cygwin** free windows toolset allowing the use of Unix scripting, useful for setting up automated data processing
- **ImageMagick** free image batch processor for resizing, rotation and more
- **ERDAS ER Viewer 11** free image viewer for very large bitmaps, also accepting modified GIS image formats
- MeshLab, CloudCompare, Blender free 3D mesh editing tool, for models generated using web services
- EXIF manipulation tools: **exiv2**, **jhead**, **AddThumbnailHeader**

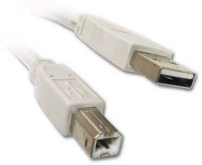
Commercial software useful for aerial mapping:
- **PtGUI Pro** commercial panorama stitching software
- **Microsoft Excel** or **OpenOffice Calc** for plotting the logs and identifying images being shot during atypical conditions
- **AcdSee** very fast commercial image viewer
- **Enso Mosaic** a professional image mosaicking suite, generating georeferenced bitmaps

Installing the software

USB drivers

You may discover the mouse could become unresponsive when using the autopilot. This is due to Windows trying to detect serial mouse by the content of the messages being sent to the computer. The system might also detect Microsoft Ballpoint mouse. This is a common problem for hardware including the GPS when connected to the PC machine. There are several methods to eliminate this problem completely. The easiest is to install the **comdisable.msi** and run it once. The software is available at http://support.microsoft.com/default.aspx?scid=kb;en-us;819036 .
This patch program stops Windows from **probing** already working serial ports for unknown hardware at boot to identify the attached devices.
Contrary to its misleading name it DOES NOT prevent the detection and normal operation of serial ports (either real ones or virtual ones created by USB-to-serial "dongles") themselves, it only prevents the Windows from analyzing the data.

Uncompress the FlexipilotSoftware.zip bundled with the system and run CDM20802_Setup.exe (or newer). You could alternatively skip this step as when the Windows is connected to the internet, plugging the autopilot is sufficient for the system to upload the most recent version.
For finishing the installation, power the autopilot with RC transmitter disconnected
(for general safety) and plug in the USB-A cable (the same your typical printer uses).



Serial port management

If at any moment Windows recognizes the hardware as 'Microsoft Ballpoint'.

As soon as this happens, it is no longer possible to open the relevant COM port. Also, from that moment on the cursor will leap across the screen in random fashion.

First disconnect the GPS module, enabling the mouse to function normally again. In Windows, go to Device Manager. This is easiest done by right-clicking on the My Computer symbol and then selecting Properties from the menu. In the new window, open the Hardware tab, and then click on Device Manager.

The window will show a list of hardware devices inside your computer. Click on Mouse to view all installed devices (usually only one mouse). Now reconnect the GPS module. After a few seconds, a second mouse will appear in the window identified as 'Microsoft Serial Ballpoint'. As soon as that happens, disconnect the module so you can use the real mouse again. Depending on your PC the second mouse will remain visible in the window for a few seconds. Right-click on its symbol, then select Properties. In the screen that pops up, you can opt not to use this device by ticking 'Disable in current profile'. Click on OK and close all other windows.

The next time you connect the module again it may happen that the Serial BallPoint is recognized again, but Windows will not activate the virtual serial mouse and the problems will not manifest themselves anymore.

Uninstalling unused serial port devices (COM)

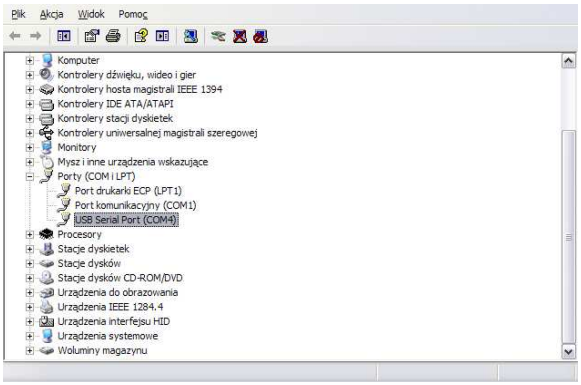
If you have used several different USB to serial port converters in the past, there could be a lot of possibly incorrect installations blocking the assignment of lower COM port number to your device.

When you install a device driver on a Windows XP machine, the operating system loads that driver each time the computer boots regardless of whether the device is present—unless you specifically uninstall the driver. This means that drivers from devices that you have long since removed from your system may be wasting valuable system resources.

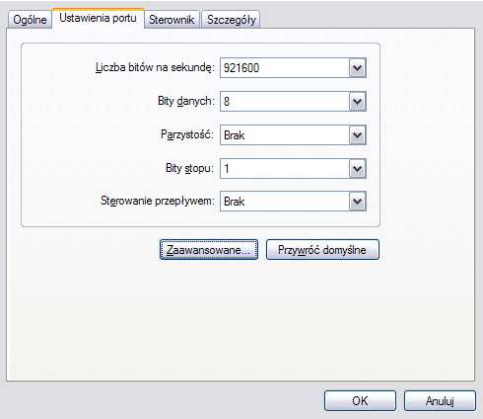
Follow these steps to view and remove these unnecessary device drivers:

- *Press [Windows]+[Break] to bring up the System Properties dialog box.*
- *Select the Advanced tab and click the Environment Variables button.*
- *Click the New button below the System Variables panel.*
- *In the New System Variable dialog box, type* **`devmgr_show_nonpresent_devices`** *in the Variable Name text box and 1 in the Variable Value text box.*
- *Click OK to return to the System Properties dialog box and then click OK again.*
- *Select the Hardware tab and click the Device Manager button.*
- *In Device Manager, go to View | Show Hidden Devices.*
- *Expand the various branches in the device tree and look for the washed out icons, which indicate unused device drivers.*
- *To remove an unused device driver, right-click the icon and select Uninstall.*

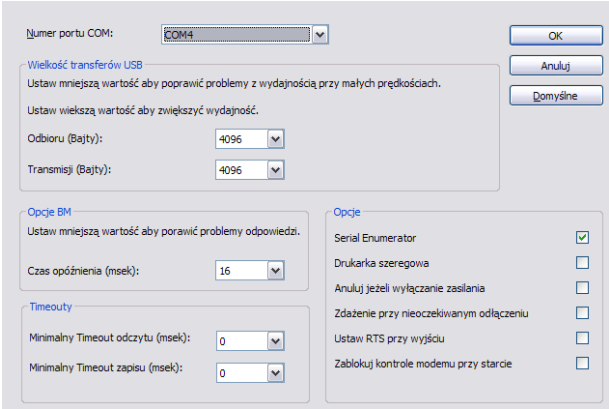
Indentifying the COM port number used by the autopilot
During the first connection, the Windows assigns automatically a fresh COM port number. In the future, every time you reconnect the autopilot, the number will remain the same. Write down this number as it will be useful during data download. Use your device manager in order to find the COM number once the autopilot is connected:



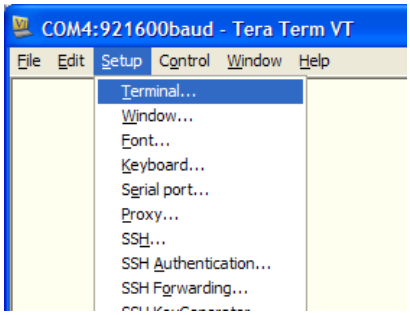
Right-click on the device name and select Properties/Port settings tab in order to change the default port settings (NOTE: no parity, no flow control, 8N1, 921600 baudrate):

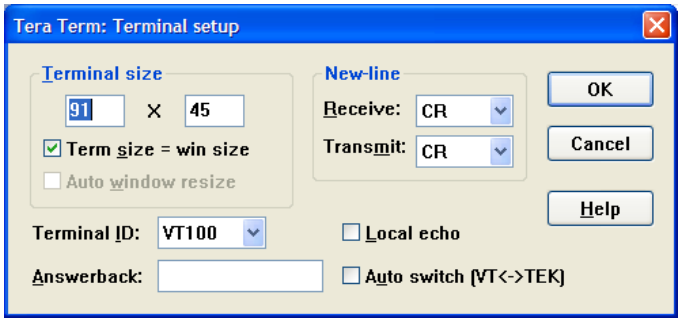


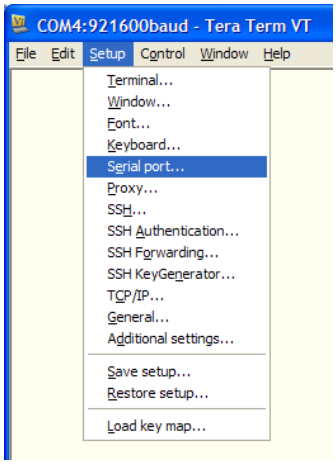
You can change the COM number using Advanced button (then reconnect the autopilot).

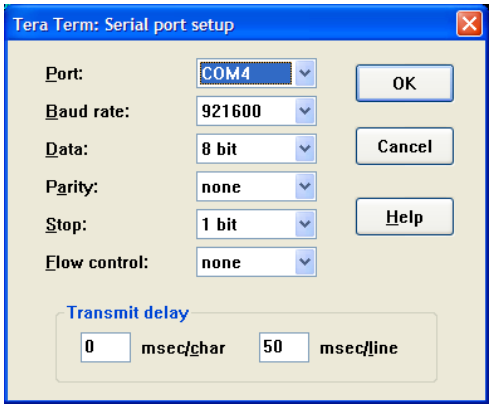


TeraTerm


COM4:921600baud - Tera Term VT
File Edit Setup Control Window Help
Terminal...
Window...
Font...
Keyboard...
Serial port...
Proxy...
SSH...
SSH Authentication...
SSH Forwarding...

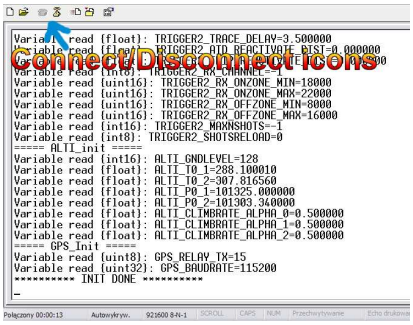

Tera Term: Terminal setup
Terminal size
91
X
45
Term size = win size
Auto window resize
New-line
Receive:
CR
Transmit:
CR
OK
Cancel
Help
Terminal ID:
VT100
Local echo
Answerback:
Auto switch (VT<->TEK)


COM4:921600baud - Tera Term VT
File Edit Setup Control Window Help
Terminal...
Window...
Font...
Keyboard...
Serial port...
Proxy...
SSH...
SSH Authentication...
SSH Forwarding...
SSH KeyGenerator...
TCP/IP...
General...
Additional settings...
Save setup...
Restore setup...
Load key map...


Tera Term: Serial port setup
Port:
COM4
Baud rate:
921600
Data:
8 bit
Parity:
none
Stop:
1 bit
Flow control:
none
OK
Cancel
Help
Transmit delay
0
msec/char
50
msec/line

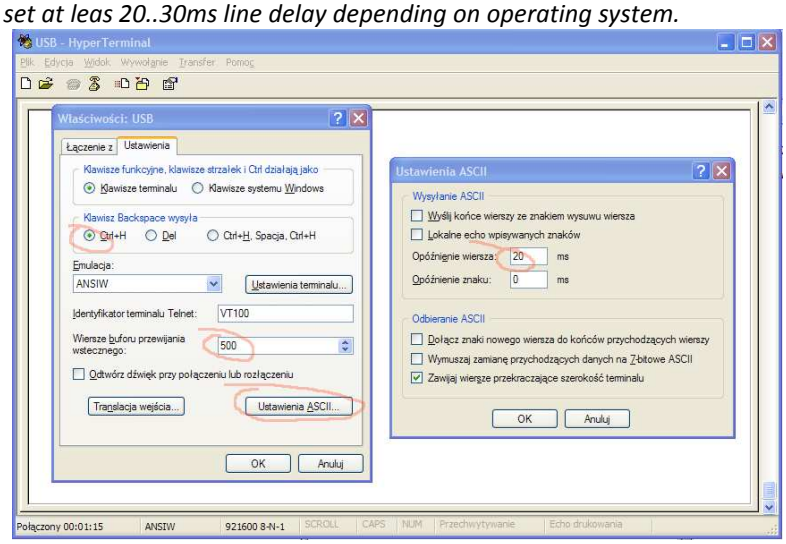
HyperTerminal

*You have to select fixed COM port number found with the procedure mentioned above, select **921600baud, 8 data bits, no parity, 1 stop bit, no flow control**.*



You don't need to close the HyperTerminal before using UAVStation, just use the disconnect buttons. Note that the serial console is not able to handle gracefully USB disconnection: while you can pull out the USB cable at any time, after plugging the autopilot on, you will need to disconnect then connect using the 2 icons on the toolbar. The autopilot must be powered before connecting to the USB, otherwise you will need cycle the connect/disconnected state as described.

Important: need to set at leas 20..30ms line delay depending on operating system.



HyperTerminal and Windows 7

While the software is not available in newest Windows, you can copy its files
from older Windows versions (XP or earlier).

1.
In theory, if you have a valid copy of Windows XP you can find the main HyperTerminal files in:

C:\Program Files\Windows NT\HYPERTRM.EXE
C:\Windows\Help\HYPERTRM.CHM
C:\Windows\Help\HYPERTRM.HLP
C:\Windows\System32\hypertrm.dll

Then create a new directory on your Windows 7 machine (maybe C:\Program Files (x86)\HyperTerminal) and copy all 4 files into that directory. The files don't have to be spread out in different directories like they were on XP.

Make a shortcut to "C:\Program Files (x86)\HyperTerminal\HYPERTRM.EXE"

Use HyperTerminal as you normally would.

2.
Additional steps are required if you would like to save your HyperTerminal session settings so that you can launch HT from a file with a specific configuration.

To be able to save HyperTerminal session settings and open them later you should:

Launch HyperTerminal.

Enter the New Connection info.

After the connection settings have been entered, use the File->Save As menu option to save your session settings as an ".ht" session file to the folder you want to keep the saved session settings in. This will result in a file on your computer named <session name>.ht (Ex: MyTestSessionSettings.ht)

Now that you have the session settings saved inside of a file, you need to tell Windows what an ".ht" file is.

Double-click the .ht file you saved and Windows will complain that it doesn't know which program to use to open the ht file. In the window that pops up that says Windows can't open this file, choose the option button that says, "Select a program from a list of installed programs", and click OK.

Click the Browse button and navigate to the folder where you installed HYPERTRM.EXE (most likely it is in C:\Program Files (x86)\HyperTerminal). Inside of there you should see HYPERTRM.EXE, select it and click the Open button to associate the HyperTerminal application with ".ht" files. Click OK to close the "Open with" window.

You may get an error that says, "Could not read session file", just click OK to ignore it. HyperTerminal may open with a blank session in it, choose File->Exit to close HyperTerminal.

To finish the process you now need to add a new entry

[HKEY_CLASSES_ROOT\ht_auto_file\shell\open\command]
to your Windows Registry.

Be VERY careful when making changes inside of the Registry. If you do not know what you are doing, you should probably not mess with it. If you want to go for it then you should probably make a backup of your Registry and then:

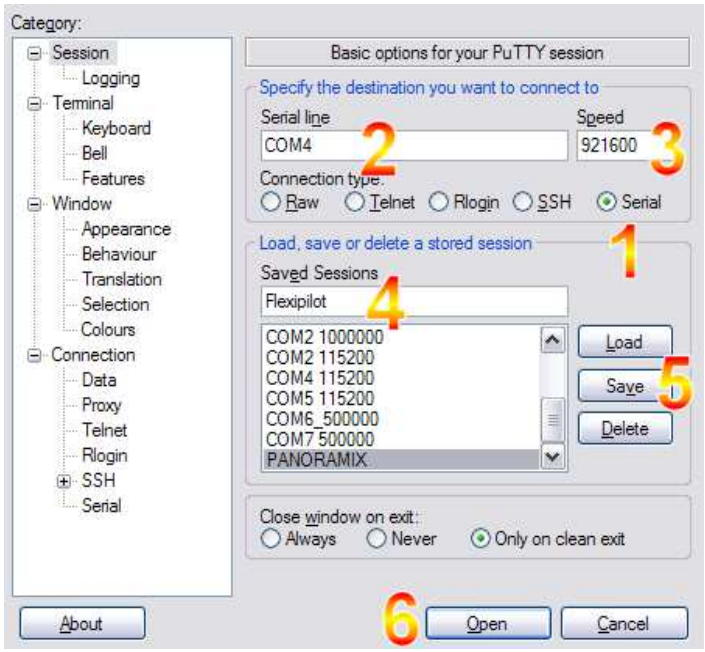
Click the Start button. Inside of the “Search Programs and Files” text box type in:
regedit
and push enter.
User Account Control will ask you if it is ok to run, click Yes, and the Registry Editor will launch.

In the left hand side of the Window you should see a node that says “Computer” and under it a yellow folder that says,
“HKEY_CLASSES_ROOT”,
double-click HKEY_CLASSES_ROOT to expand it.

Highlight the phrase, “HKEY_CLASSES_ROOT”, and then RIGHT-click on it and choose, New->Key from the context menu.
A folder will be created that is named, “New Key #1”, type over that name and change it to:
ht_auto_file
Highlight the phrase, “ht_auto_file”, and then RIGHT-click on it and choose, New->Key from the context menu.
A folder will be created that is named, “New Key #1”, type over that name and change it to:
shell
Highlight the phrase, “shell”, and then RIGHT-click on it and choose, New->Key from the context menu.
A folder will be created that is named, “New Key #1”, type over that name and change it to:
open
Highlight the phrase, “open”, and then RIGHT-click on it and choose, New->Key from the context menu.
A folder will be created that is named, “New Key #1”, type over that name and change it to:
command

Highlight the phrase, “command” and on the RIGHT hand side of the screen you should see the single word (Default). Double-click on the word (Default) to open a window that will let you enter a text value inside of it. Inside of the text box labeled, “Value data”, type in:
"C:\Program Files (x86)\HyperTerminal\HYPERTRM.EXE" %1
Exactly like it is shown above (or the path to where your HYPERTRM.EXE is installed).
DO put quotes around "C:\Program Files (x86)\HyperTerminal\HYPERTRM.EXE" and then push the space bar and enter the %1 outside of the quotation marks. Double-check that you have typed (or pasted) this in correctly and then click OK.
Choose File->Exit to close the Registry Editor.
Go back to your file with your saved HyperTerminal settings (MyTestSessionSettings.ht) and Double-Click on it. HyperTerminal should now open with the saved session settings.
If you still get an error that says, “Could not read session file”, open up the Registry Editor like you did above and navigate back to [HKEY_CLASSES_ROOT\ht_auto_file\shell\open\command] ->(Default) and make sure that quote marks did not get placed around the %1. If they did, remove the quotes marks around %1, click OK, close the Registry Editor, and try again.

PuTTy



Note: because PuTTy has no 'line delay when sending' feature, you will inevitably experience an occasional character loss when sending a configuration to the autopilot. This is a limitation of PuTTy and has similar consequences to any other embedded system. As a result you should rather type into this console, instead of pasting large blocks of text. Pasting several lines is OK, but you must verify the information has been parsed correctly. Having said that, this console is very easy to manage sessions and works very well in the field.

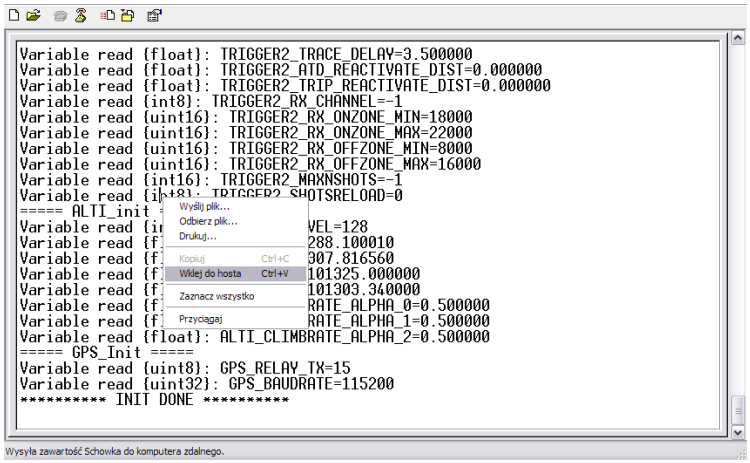
Setup: editing the autopilot variables and issuing commands

ASCII setup file

The variables stored in side the autopilot are doubled in the text files bundled with the autopilot:
@@@WAYPOINT_HOMEMODE=0 //0=Takeoffhome //1=FixedHome //2=RemoteFixedHome
@@@WAYPOINT_AUTOLAND=0
@@@WAYPOINT_HOME_TURNDIR=0 //0=any, 1=left, 2=right
@@@WAYPOINT_HOME_LAT=51.123456
@@@WAYPOINT_HOME_LON=18.123456

Interactive session using HyperTerminal

Those files are in fact commands that can be issued interactively in order to set given variable. Thanks to this feature, you can alter any single variable 'by hand' entering those command s from the keyboard, but also when the HyperTerminal is connected, you can open the file in the editor, CTRL-C, right click in the HyperTerm (the serial port must be connected), select Paste to Host:



The commands can be typed interactively: when you press '@' three times, a command prompt in the form '@@@' appears and you can type phrases and expressions. The backspace is functional in this mode. If you entered the mode accidentally, simply enter invalid, random text and press enter.

A few examples:

@@@anything wrong here?
← observe no answer, unrecognized command
@@@RXOVROFF Blah Blah followed by any comment...
RX override DISABLED... ← recognized valid command: RXOVROFF, comment ignored

@@@GPS_BAUDRATE=115200 my preferred baudrate!
GPS_BAUDRATE=115200 0x0001C200 #332@1199{u32} ← recognized valid variable assignment, comment ignored.
Observe how the value is displayed also in hexadecimal form, the variable ID (332), its location in memory (1199) and data type (u32 is unsigned 32 bit integer).

@@@GPS_BAUDRATE=-1
GPS_BAUDRATE=4294967295 0xFFFFFFFF #332@1199{u32}

you can set any value in the edition mode…

but after reboot, when the variable is updated:
===== GPS_Init =====
INVALID==> 4294967295 Variable read {u32} : GPS_BAUDRATE=115200
you will be notified about incorrect values entered during edition mode.
Additionally, the default fail-safe value is assigned and is displayed at the end of the message.

Note: as long as there is a setup error, all servo outputs are off at autopilot mode. You can hear ESC controller beeping and servos idling after entering the wrong value and resetting. Digital servos will hold their positions on their own.

Note that you can add any text commenting the functionality at the end of the line, however the line length should be kept reasonably short (less than 80…120 characters) in order to be able to mass-update all setting by pasting them to HyperTerminal Console.
In general, every entered string that has been recognized is followed by clear description.
The commands have immediate consequences, while variable assignments require RESET for updating because the inter-variable dependencies might disallow setting one of them in certain cases.
Evaluating the validity during assignment would involve complicated logic dependency structure and is not possible. All validity tests are performed during reboot only.

Another example:
@@@WAYPOINT_BEGIN?
WAYPOINT_BEGIN=0 0x0000 #824@2433{u16}
…is asking for actual variable written to the memory.

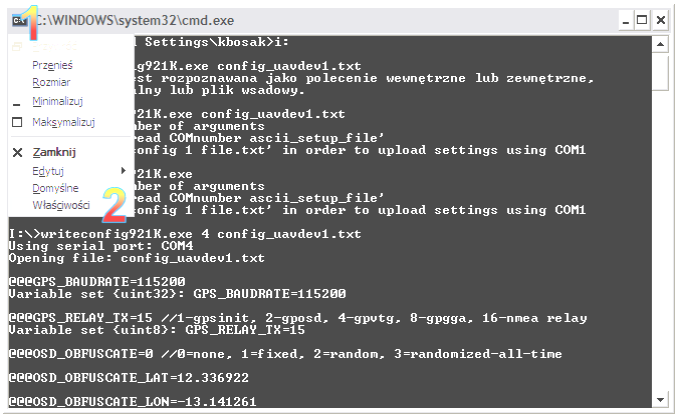
Automated update using WRITECONFIG

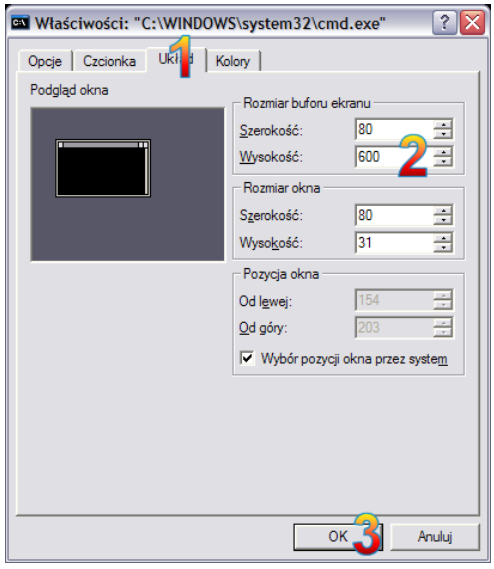
You can use command-line software writeconfig921K.exe
in order to update the setup file. This helps in automating the process.

You can run **cmd** (Windows console/ Command Line interface) and start the following session:

```
I:\>writeconfig921K.exe
Error: wrong number of arguments
use syntax 'logread COMnumber ascii_setup_file'
Example: 'writeconfig 1 file.txt' in order to upload settings using COM1

I:\>writeconfig921K.exe 4 config_flexipilot1.txt
Using serial port: COM4
Opening file: config_flexipilot1.txt

@@@GPS_BAUDRATE=115200
Variable set {uint32}: GPS_BAUDRATE=115200
....
```

Since the text output can be long, you can consider increasing text window buffer length; this is also useful for UAVStation text output window:





Select Window/Properties/Composition tab, increase display buffer length

LOG configuration

LOG is represented by simplified file system: each flight/logging session is a separate file. You can record up to 256 flights, of the total length 40 minutes at 8Hz (this amount varies between autopilot versions depending on main loop speed). You can either increase or decrease logging rate at the expense of logging time. The particularity of LOG system is that it records all the data periodically, therefore (unlike TRACE) is not suitable for logging rare events. If the log is full, it stops the logging. Because of the real-time file system nature, it is not possible to delete a single flight, you have to download all the flights then erase the whole LOG. LOG is functional also during simulation and can be enabled/disabled manually.
Every time you turn the logging off, a session is ended. When starting again, next session ID is attributed.
During log output the autopilot is stopped entirely for maximum efficiency and failsafe watchdog inside the servo processor is passing to remote RC receiver input. **This might spin the motor if the throttle is in wrong position!**

Storing Raw NMEA GPS data

You can log the data using either fixed list of parameters, or you can log the GPS data (however, taking into account 240 characters per message at 5Hz, this is 1200 characters per second. This will fill the LOG in one hour). The total recording length is not shown during directory listing, as it is not possible to evaluate it precisely (GPS message length varies).
Use LOG_DUMP_MODE=0 (raw bytes) and LOG_SOURCE=1 (NMEA) for this scenario.

Using LOG-related commands

Open the console, press **d**, the output should look similar to:

```
LOG_Dir...
FileID  1 Pages:1738  42.4% Bytes:  914188 KBytes:  893 Time:29m6s
FileID  2 Pages: 433  10.6% Bytes:  227758 KBytes:  223 Time:7m16s
     USED Pages:2171  53.0% Bytes: 1141946 KBytes: 1116 Time:36m20s
     FREE Pages:1925  47.0% Bytes: 1012550 KBytes:  989 Time:32m13s
     LOG period: 10 (freq=2.000000Hz, interval=0.500000s)
Done.
TRACE_Dir...
 Used entries: 2603 (periodic trace duration max. 43m23s)
 Free entries: 5589 (periodic trace duration max. 1h33m9s)
 Total entries: 8192 (periodic trace duration max. 2h16m32s)
 Trace period: 20 (freq=1.000000Hz, interval=1.000000s)
Done.
```

You can also type command **@@@LOGDIR** and **@@@TRACEDIR** or press F3 in UAVStation for the same result.

@@@LOGDUMPn prints the file number **n** using actual style from LOG_DUMP_MODE variable. **@@@LOGDUMP** without number prints all files.
Example (with LOG_DUMP_MODE =1):

```
@@@LOGDUMP2
```

```
LOG_Dump...

#ARN2#
#ARF112:0,0,0,0,0,0,0,0,64,128,64,128,4,4,4,3,3,3,3,3,3,3,3,3,3,3,5,0,0,0,0,4,4,
4,64,128,64,128,0,0,0,0,0,0,0,5,5,3,64,128,5,5,5,5,5,5,5,5,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,4,4,4,4,4,0,5,5,5,5,5,5,5,5,4,4,4,4,5,5,5,4,1,1,1,1,4,4,4,4,1,1,1,0,0
,4,25#
========== fileid=2 ==========
#ARi02440439050206C7070108CC09010A350B020C0B0D020E1A0F021041117F124C13421474154616
88174118751AA01B0F1C7220BB210122E224E326E23275330534AA3501381B3A043C043E5B40114101
425B44204501483C493E4C3C4D3E4E3E4F3D50E5513A54F8552A5A6F5B275C785D0A5E785F0A64BA65
3B680A69F76AF46BF670FE71FF742F75027633770290099102920F946495BE9C04A4DAA5FFA60AA85A
AADAABFFAC0AAE5AB013B628B723BAA0BB0FBCA0BD0FBEF0BF07C08BC255C3FAC417C50BC61FC8E2CA
F8CB02CC5CCD05CED8CF04D007D434DE23#
#ARi1C7322E326E34840493C4C404D3C66016AF694EBCED7#
#ARi14761C7520B8482C493E4C2C4D3E4E3A63A064E065BC6AF1900A949895C2CED8#
#ARi1042147820C52E013001480C4C0C4E3250E463C064AF65BB6AEC700E7100763494C095C4B014CE
D9#
#ARi1041147920DA2E00300048C0493D4CC04D3D4E2E63D06498653D660270FE71FF763594DF95C6B0
15#
#ARi06C81042147A18771C7620F348604C604E1A50E3638864ED66056AE67634900994FE95C8B083B1
FFB2FFB3FFCED8#
Done.
```

@@@LOGRECOVER
Is equivalent to @@@LOGDUMP, except that it ignores file table, printing all flash content instead. This allows log recovery even in case of physical damage to the memory chip storing flash usage list. It can take several minutes to transmit all content. No data is being modified in the process.

@@@LOGSTART or '**L'**, **@@@LOGSTOP** or 'l' are forcing stopping and starting the logging at any time, this is the only way to proceed when you want to stop logging without turning FLEXIPILOT off after a flight.

@@@LOGCLEAR erases all the files
'**C**' (**F8** in UAVStation) erases both TRACE and LOG.
Clearing the logs also **stops** actual logging session.

'P'/'p' is toggling printing of all log variables on the console.

Configuration variables

LOG_AUTOSTART
Default: 2
Possible values:
0 - Do not start log automatically
1 - Start new logging session after autopilot reset (will contain pre-takeoff data)
2 - Start logging upon takeoff, but continue after landing

LOG_PERIOD
Default: 4
Possible values:
1-64000
This parameter affects logging frequency when LOG_SOURCE=0 (periodic logging).
As the logger is being evaluated at 20Hz, LOG)
1 is 20Hz logging (at the expense of logging time)
4 is 5Hz
20 is 1Hz and allows around 138min of continuous logging
36000 is 36000/20=1800s=30min logging interval

LOG_SOURCE
Default: 0
Possible values:
0 - Standard log data table
1 - GPS NMEA output ('from GPS')
2 – OSD output and GPS input ('to GPS')
3 – Console input ('what you type')
4 – Console output ('what is being printed')
5 – Feedback tuning loops

LOG_MAXTIME
Default: 0 s (no time limit)
Possible values: 0…1e6 s
The logging will stop automatically after recording at least given amount of seconds.
Useful for logging takeoff only.

TRACE configuration

TRACE is a monolithic nonvolatile data area containing simplified information about plane's location and orientation. This provides optimal data storage for providing the data for image processing. It can also log periodic data but typically at low rate (1-2Hz) because its size is relatively limited (8192 positions). Contrary to LOG, every record TRACE contains a timestamp. TRACE can be only downloaded or erased as a whole.
You only disable or enable the trace, as there are no sessions. Each even is independent, but among events, takeoff is recorded what allows sorting the data per flight by LOGDECODE utility.

Using TRACE-related commands

@@@TRACEDIR prints space usage.

```
TRACE_Dir...
 Used entries: 2603 (periodic trace duration max. 43m23s)
 Free entries: 5589 (periodic trace duration max. 1h33m9s)
 Total entries: 8192 (periodic trace duration max. 2h16m32s)
 Trace period: 20 (freq=1.000000Hz, interval=1.000000s)
Done.
```

@@@TRACEDUMP prints the whole event content.

```
@@@TRACEDUMP
TRACE_Dump...
#ART00000:000391.820,+51.124317,+017.034077,+00000.1,-
023.3,+005.6,037.3,001,2011-03-06,19:35:00.820,00000#
#ART00001:000393.066,+51.124313,+017.034071,-00000.1,-
023.2,+005.6,037.3,001,2011-03-06,19:35:02.066,00000#
Done.
```

@@@TRACESTART or '**T**', **@@@TRACESTOP** or '**t**' control trace logging manually.
@@@TRACECLEAR clears the whole content
'**C**' (**F8** in UAVStation) erases both TRACE and LOG.
Clearing the trace also **stops** the logging in process.

Configuration variables

The layout of TRACE engine is similar to LOG engine as it allows periodic logging, interleaved with special events.

TRACE_AUTOSTART
Default: 2
Possible values:
0 - Do not start log automatically
1 - Start new logging session after autopilot reset (will contain pre-takeoff data)
2 - Start logging upon takeoff, but continue after landing
This controls the periodic logging only.

TRACE_PERIOD
Default: 0
Possible values:
0, 20…64000
This parameter affects TRACE periodic logging frequency.
Contrary to LOG, TRACE PERIOD can be 0 disabling the periodic logging.
In this case, when the trace is enabled, only special events will be logged.
As the logger is being evaluated at 20Hz:
TRACE_PERIOD=0 disables periodic logging
TRACE_PERIOD=20 is 1Hz logging
TRACE_PERIOD=40 is 0.5Hz logging (every 2s)
TRACE_PERIOD=36000 is once per 30min (36000/20Hz=1800s=0.5h)

TRACE_SKIPEVENT
Possible values: 0..255 (default 0)
Filters out events from trace, allowing longer logging time.

1	TRACE_NOTRIG1	No Trigger1 events
2	TRACE_NOTRIG2	No Trigger2 events
4	TRACE_NOWPTHIT	No waypoint hit events
8	TRACE_NOWPTTARGET	No target positions
16	TRACE_NORETHOME	No RETHOME events
32	TRACE_NOFLIGHTNUMBER	No flight number
64	TRACE_AUTO_REDUCE_RATE	Allow automatic logging rate reduction when trace almost full (divide rate by two for every halving of remaining free memory)
128	TRACE_AUTO_STOP_PERIODIC	Allow halting periodic logging when trace almost full (above 90% usage and less than 64 entries left)

Values are additive.
0 means store everything, do not halt periodic logging until trace is full (8192 entries).
The value of 192 is a good choice for a flight of unknown length, when one wants to have periodic trace as long as possible.
Using AUTO_REDUCE_RATE, assuming 8192 trace capacity and 1Hz initial logging rate, you will register 4096 entries with 1Hz period, 2048 entries with 0.5Hz period and so on.
With AUTO_STOP_PERIODIC you will achieve **tripled total logging time**, until the last 64 entries will remain. Those 64 events roughly correspond to number of additional decisions and waypoint switch position during the flight.

AUTO_REDUCE_RATE: trace duration vs frequency with 1Hz initial rate (TRACE_PERIOD=20)

Entries left	Period [s]	Duration [s]	Frequency [Hz]
4096	1	4096	1
2048	2	4096	0.5
1024	4	4096	0.25
512	8	4096	0.125
256	16	4096	0.0625
128	32	4096	0.03125
		24576s=6h48m	

AUTO_REDUCE_RATE: trace duration vs frequency with 0.5Hz initial rate (TRACE_PERIOD=40)

Entries left	Period [s]	Duration [s]	Frequency [Hz]
4096	2	8192	0.5
2048	4	8192	0.25
1024	8	8192	0.125
512	16	8192	0.0625
256	32	8192	0.03125
128	64	8192	0.015625
		49152s=13h39m	

TRACE_ALTMODE

Possible values: 0 … 66

1 - Trigger events use GPS MSL (instead of AGL)

2 - Trigger events use GPS WGS84 (instead of AGL)

32 - Non-trigger events use GPS MSL (instead of AGL)

64 - Non-trigger events use GPS WGS84 (instead of AGL)

Bitwise selection of altitude type to be logged in trace.

Example:

2- trigger events log as WGS84, all others use barometric AGL altitude, common for geodesic applications

33 - all altitudes as MSL

66 - all altitudes as WGS84

Reserved variables

TRACE_OFFSET

Possible values: 0 … TRACE_SIZE-1

Reserved for future use.

TRACE_SIZE

Possible values: 0 … EEPROM_LENGTH/16

Reserved for future use.

Data download using LOGREAD (supplied)

You can use command-line software logread921K.exe
in order to download the data from serial port.
In fact the tool sends the command-line command to the serial port and writes the results on the disk. This tool is not losing any characters and is easier to use than serial port consoles. It can be used for downloading data, reports and debugging messages from the autopilot.

You can run **cmd** (Windows console/ Command Line interface) and start the following session:

```
I:\>logread921K.exe 4 @@@WPTSHOWLIST
Using serial port: COM4
Executing command: @@@WPTSHOWLIST

@@@WPTSHOWLIST
#ARW0:(51.123456,17.123456,100),0.000000,0.220000,100,0,4,0#
#ARW1:(51.123456,17.123456,100),90.000000,0.220000,100,0,4,0#
#ARW2:(51.123456,17.123456,100),180.000000,0.220000,100,0,4,0#
#ARW3:(51.123456,17.123456,100),270.000000,0.220000,100,0,4,0#

I:\>
```

Using LOGDECODE (supplied)

This is the most important tool you will be using after each flight.

Using this universal LOGREAD tool, the following data download scripts have been constructed (**getlog.bat**):

logread921K.exe %COMPORT_FLEXIPILOT_USB% @@@LOGDUMP > log.txt
logread921K.exe %COMPORT_FLEXIPILOT_USB% @@@TRACEDUMP > trace.txt
logread921K.exe %COMPORT_FLEXIPILOT_USB% @@@WPTITERALL > wptiter.txt
logread921K.exe %COMPORT_FLEXIPILOT_USB% @@@PARAZONESHOW > zone_para.txt
logread921K.exe %COMPORT_FLEXIPILOT_USB% @@@RETHOMEZONESHOW > zone_rethome.txt
logread921K.exe %COMPORT_FLEXIPILOT_USB% @@@RESET > reset.txt

Note: ***COMPORT_FLEXIPILOT_USB is the name of environment variable. Learn from the internet about setting Windows andwironment variables. If, for example, FLEXIPILOT installs as COM7, you have to set 7 as environment variable. The value can be different between your laptop and stationary PC at home, so a different value may be needed yet the getlog.bat file stays the same.***

Using LOGDECODE data processing tool:

This universal tool is able to process any autopilot output and extract the data in a useful form. Launch **decodelog.bat** script in order to execute the following commands:

```
logdecode.exe log.txt log
logdecode.exe wpt.txt wpt
logdecode.exe trace.txt trace
```

As a result the content of LOG and TRACE will be split in section and written in uncompressed format on the disk into the files:

log001.txt
log002.txt
…
and
trace001.txt
trace002.txt
…

Also corresponding Google Earth files will be written:

trace001.kml
trace002.kml
…
wpt000.kml

From trace data, for the events corresponding to trigger action, 2 GPS NMEA files containing the data for all flights will be generated:

trace.nmea1.txt
trace.nmea2.tx

logNNN.txt files are ASCII files that are immediately useful with Scilab or any other scientific or plotting software.

If the TRIGGERS were not used, or the TRACE was disabled, or the LOG was empty, no additional files will be written.

Detailed syntax and welcome screen:

```
======================== AerialRobotics LogDecode =========================
Built: Aug  5 2013 23:24:42
Syntax:
logdecode options input-files
--altitude-offset=meters defines absolute altitude of takeoff point
  KML and NMEA output will be absolute instead of relative to ground
--time-offset=days:hours:minutes:seconds adds correction to UTC time
  all values are positive or negative integers, any range
  i.e seconds can be -5000, hours can be +115, milliseconds are always positive
--round-seconds all times will be rounded to the nearest second,
  improving time-based matching of photos with some external software
--roll-stabilized roll value output will be fixed to 0
--pitch-stabilized pitch value output will be fixed to 0
--roll-invert roll value output will be inverted (default: positive turn right)
--pitch-invert pitch value output will be inverted (default: positive nose up)
--heading-offset is added to all heading values, useful for manual tuning
--log-period forces LOG timing divider, 0,1 is 20Hz, 20 is 1Hz, max 65535
--wind-speed=km/h >=0 sets wind speed and disables wind speed estimation
--wind-heading=<0..360) sets wind heading, ignored when speed not set
--wind-lower-alt=m --wind-upper-alt=m sets altitude limits for wind estimation
--use-kml-timestamps KML file will use time selection slider
== Camera geometry ==
 camera1/2 options affect autopilot trigger1/2 position decoding, respectively
--cameraX-aspect-ratio=N (4/3=1.33, 5/4=1.25 etc), X=1 or 2
--cameraX-lens-angle=deg horisontal FOV in degrees
--cameraX-rotation=deg adds rotation offset, clockwise
--cameraX-roll=deg adds offset to correct, positive is right
--cameraX-pitch=deg adds offset to correct, positive is up
--cameraX-trace-delay=s specifies time offset for eliminating GPS lag (use negat
ive)
--cameraX-gps-up-offset=m specifies how higher is GPS relative to camera
--cameraX-gps-fw-offset=m specifies how forward is GPS relative to camera
--cameraX-gps-rt-offset=m specifies how starboard (right sided) is GPS relative
to camera
--cameraX-overlay_opacity=0..255 - 0 is invisible, 255 is opaque
--cameraX-calcamfilename="C:\Enso MOSAIC\Canon.cal"
  a path to camera calibration generated by CalCam for EnsoMosaic
--cameraX-maxoverlap=Ratio skips consecutive photos with given overlap
   if closer than Ratio of their size (1=skip one photo radius, 0=continuous)
=== Two methods to match trigger/shutter press to photo ===
==> 1. Matching  using event number:
--cameraX-filename=IMG_####.JPG a pattern defining photo numbering
  trigger event number+eventoffset will be written replacing # marks
--cameraX-eventoffset=N number of the first photo file,
  will be added to trigger event
  example:
  first file is ABC0023.jpg in subdirectory originals,
  first reasonnable (outdoors, after takeoff) event is 5th line in TRIG.csv,
  eventoffset=18
  filename="originals\ABC####.jpg"
==> 2. Matching  using JPEG EXIF time, must provide:
--cameraX-timefilename=jpegtimes.txt a file with EXIF timestamp
  must be semicolon-delimited, i.e.
  C:\Directory\Photos\DSC_0235.jpg;2010-05-29 07:27:46
  alternatively with additional UTC time:
  5835.JPG;2010-05-29 07:27:46;1275118066
Example:
logdecode --altitude-offset=180 trace.txt
logdecode --altitude-offset=180 wptiter.txt
logdecode log.txt
Parse several files at once in order to merge
GPS data from trace and wind data from log:
logdecode trace.txt log.txt trace.txt
```

As you can see from the options above, additional options help improve GoogleEarth output and can provide a simple preview of image coverage by matching bitmap filenames to trace events with logged positions and orientation.

Contents of LogDecode output

The data in log and trace complement each other: trace contains asynchronous even with GPS timestamp, while log contains periodic data but has no idea about special actions or decisions. LogDecode merges the information, splits the result per-flight, but generates a separate output from each source file.
For example, the following inputs and outputs are related:

log.txt → log.txt.LOG001.txt, log.txt.LOG002.txt, log.txt.LOG003.txt
(the log decompressed and split into columns, the raw data is important mostly for safety and reliability analysis).
Sample content:

```
1023 0 541 230 0 805 644 571  51.123456  17. 123456 28.00 20.30 20.16 +90.0 +22.2 +21.4 +258.3 +258.7 -6.8 -13.7 -13.7 -12.5 -12.5
+126.3 +42.4 +1.06 27 8 8 62 1.12 1.89 2.20  0.026489  0.026489 12808 15014 0 15386 20000 0 10375 +7.60 +6.27 +200.0  0.195297 +0.22
+0.88 -13.25 -14.50 -7.81 -13.25 -13.25 +16.02 +37.13 12808 11835 9622 15386 0 0 9696 15200 0 0 0 0 0 0 17005 0 0.00 0.00 0.00 0.00
0.00 0
1023 0 537 230 0 813 641 576  51.123456 17.123456 28.00 22.95 22.85 +90.0 +22.2 +20.9 +256.6 +257.1 -9.5 -9.3 -9.3 -16.0 -16.0 +126.3
+42.4 +0.53 27 8 8 62 1.12 1.89 2.20  0.136108  0.136108 12773 14726 0 15374 20000 0 10375 +7.60 +6.28 +200.0  0.184937 +0.07 -6.12 -
16.82 -10.03 -6.07 -16.82 -16.82 +14.66 +36.06 12773 11835 9621 15374 0 0 9696 15205 0 0 0 0 0 0 17498 0 0.00 0.00 0.00 0.00 0.00 0
```

log.txt → log.txt.STAT-001.txt, log.txt.STAT-002.txt, log.txt.STAT-003.txt
Flight statistics, this helps to tell quickly which flight was a simulation, which was manual attempt and which was the successful mission. It also tells wind direction, so you can adjust your next flight easily.
Sample content:

```
Log content:
LOG_POINTS=11500
TRACE_POINTS=11500
TRIGGER1_POINTS=0
TRIGGER2_POINTS=0

Wind at cruise altitude:
Wind lower band=0[m]
Wind upper band=20000[m]
Cruise speed=32.3[km/h] 9.0[m/s]
Wind speed=19.7[km/h] 5.5[m/s]
Wind heading=302[deg] (from 122[deg])

Flight statistics:
Max n. of satellites in view=11
Avg n. of satellites in view=8.4
Max altitude above takeoff point AGL=286.0[m]
Max GPS altitude above sea level MSL=383.2[m]
Cumulative climb=286.01[m]
Max distance to takeoff=0.328[km]
Max VLOS distance to takeoff=0.435[km]
Total travel (groundtrip)=10.640[km]
Takeoff: 2011-02-26 13:54:36.746
Last event: 2011-02-26 14:25:16.585
Total log time=1840.00[s]
Total flight time=1831.36[s] 99.5% of log time
Autonomous flight time=1774.08[s] 96.9% of flight time
Average groundspeed=20.9[km/h] 5.8[m/s]
Min temperature=-1.89[C]
Max temperature=5.32[C]
Min outside temperature=-4.13[C]
Max outside temperature=1.00[C]
```

trace.txt + jpgtimes.txt → trace.txt.TRIG2-002.photomatch.csv
This time only flight 2 was using trigger2 (shooting photos), moreover the times in jpgtimes.txt matched and filename list has been generated:

```
Filename;Line number;Frame number;Latitude[decimal];Longitude[decimal];Altitude MSL[m];Altitude AGL[m];Top-heading[deg];Crab angle[deg];Roll[right, deg];Pitch[up, deg];
_orig/00391.JPG;  1;  1;  51.055973;  17.381823;251.6;124.6;244.0; -0.0;  0.0; -1.0;51.055973°N  17.381823°E; 51° 3.3584'N  17°22.9094'E; 51° 3'21.50''N  17°22'.56''E; 51° 3'22''N  17°22'55''E;5103.3584,N,01722.9094,E
```

Note the position is written here using different formats. Also image orientation is available, with heading (fuselage direction) rotated by camera rotation. Note that crab angle (fuselage direction less ground course) is available, allowing image pre-rotation by processing software.

trace.txt → trace.txt.TRIG2-002.csv
This one contains all events when the trigger 2 has been pressed, but not necessarily the camera was operational.

```
Event#;UTC Date;UTC time;UTC time[ms];Log time[s];Latitude[decimal];Longitude[decimal];Altitude ATL[m];Roll[right, deg];Pitch[up, deg];Heading[deg];Course[deg];Groundspeed[km/h];Offline airspeed[km/h]
1;2009-12-27;14:41:41;000;106.000000;51.055973;17.381823;124.600000;0.000000;-1.000000;223.281616;-20.718384;244.000000;32.000000;36.657291
```

trace.txt → trace.txt.TRIG2-002.ensomosaic.gps, trace.txt.TRIG2-002.ensomosaic.trp
If photo matching is possible (jpegtimes.txt supplied the filenames) this file contains input for EnsoMosaic

trace.txt → trace.txt.TRIG2-002.nmea, trace.txt.TRIG2-002.gpx
A synthetic GPS NMEA data for all points where Trigger2 has been used:

```
$GPRMC,144141.000,A,5103.3584,N,01722.9094,E,17.3,244.0,271209,,*31
$GPGGA,144141.000,5103.3584,N,01722.9094,E,1,09,1.0,251.6,M,,,,*0D
$GPGLL,5103.3584,N,01722.9094,E,144141.000,A,*12
$GPVTG,244.0,T,223.3,M,017.3,N,032.0,K*48
$GPGSA,A,1,,,,,,,,,,,,,1.0,1.0,1.0*31
```

Synthetic XML GPS data:

```
<gpx>
<trk>
<trkseg>
<trkpt lat="51.055973" lon="17.381823">
 <time>2009-12-27T14:41:41Z</time>
 <roll>0.0</roll>
 <pitch>-1.0</pitch>
 <compass>223.3</compass>
 <heading>223.3</heading>
 <course>244.0</course>
 <speed>8.9</speed>
 <agl>124.60</agl>
 <ele>251.60</ele>
</trkpt>
```

trace.txt → trace.txt.TRACE-001.kml, trace.txt.TRACE-002.kml
Google Earth files with all flight details (but the flight path may or may not be present depending on trace options). It also contains Image projections as overlay if image matching was used.

trace.txt → trace.txt.TRACE-001.csv, trace.txt.TRACE-002.csv
log.txt → log.txt.TRACE-001.csv, log.txt.TRACE-002.csv
log → log.txt.LOG-001.gpx, log.txt.LOG-002.gpx

log → log.txt.LOG-001.nmea, log.txt.LOG-002. nmea
log → log.txt. TRACE-001.gpx, log.txt.TRACE-002.gpx
log → log.txt. TRACE-001.nmea, log.txt.TRACE-002. nmea
In this case, a **limited** info (TRACE-style) is generated for flight 001 and 002 from log or trace.
All positions noted in the logs above are used, not only those when a shutter has been used.
The difference between TRACE and LOG style output is that the former uses original GPS timestamps and not the autopilot real-time clock, and is therefore more accurate.

log.txt → log.txt.LOG-001.csv, log.txt.LOG-002csv
In this case, **extended** info (LOG-style) is generated for flight 001 and 002 from log.

log.txt → log.txt.WIND-001.csv, log.txt.WIND-002.csv
Wind profile data (also summarized in log.txt.STAT-001.txt and log.txt.STAT-002.txt).

wptiter.txt → wptiter.txt.TRACE-001.kml
Preview of the next mission.

As a summary, LOG style output is richer than TRACE output, therefore log can generate either LOG style or TRACE style output. LOG, however, is not using original GPS timing, therefore in absence of TRACE the processed results have time reference starting since takeoff (hour 00:00).
WPTITER output is useful only for validating the NEXT flight and uses SIM_LATITUDE/LONGITUDE for extrapolating the waypoint evaluation.

Advanced usage example, matching images using shutter event ID:

```
logdecode.exe wptiter.txt trace.txt log.txt trace.txt --camera1-aspect-ratio=1.333333 --camera1-lens-angle=50 --camera1-rotation=180 --camera1-roll=0 --camera1-pitch=0 --camera1-overlay-opacity=200 --camera1-id=711 --altitude-offset=174  --camera1-filename=DSC#####.JPG
```

In this case, trigger1 will be used for generating image Overlays in Google Earth.
The JPG files collected are starting with DSC00711.JPG, hence the pattern and camera1-id.
From the Google Earth one can read with a mouse the MSL elevation in meters of the takeoff point: this is entered by --altitude-offset=174 and makes logs look 'straight' when flying over uneven area in Google Earth (which can display only absolute altitude or relative to ground altitude, while the autopilot measures altitude above takeoff point; the altitude measured by GPS during takeoff is rarely accurate). This photo matching method is possible only when the number of button presses is equal to the number of JPG files, whose numbering must be continuous.
You can edit trace.txt manually if there are more presses than the actual photos, but you must find what events to delete.

Preparing input for EnsoMosaic, matching images using EXIF timestamp:

```
logdecode.exe wptiter.txt trace.txt log.txt trace.txt --round-seconds --time-offset=0:+1:-1:+123
--camera1-calcamfilename="C:\EnsoMOSAIC_trial\Camera\Canon_PowerShot-S45.cal" --camera1-rotation=90 --
altitude-offset=180 --camera1-timefilename=jpgtimes.txt
```

In this case, trigger1 will be used to match with images using their digitized time.
You need to create jpgtimes.txt beforehand, f.ex using cygwin script and exif tool:

```
(
for i in *.JPG
do
TIMEVAR=`exif --ifd=EXIF -t 0x9004 $i | grep -w Value:`
UTCTIMEVAR=`date -u +%s -d "$ TIMEVAR "`
echo "$i;$ TIMEVAR;$UTCTIMEVAR"
done
) > jpgtimes.txt
```

The final file should look like:

```
_orig/00391.JPG;2009-12-27 14:41:40;1261924900
_orig/00392.JPG;2009-12-27 14:41:45;1261924905
_orig/00393.JPG;2009-12-27 14:41:49;1261924909
```

the last column contains UTC time in us, is optional, the following is also accepted:

```
_orig/00391.JPG;2009-12-27 14:41:40
_orig/00392.JPG;2009-12-27 14:41:45
_orig/00393.JPG;2009-12-27 14:41:49
```

this is also accepted:

```
_orig/00391.JPG;2009-12-27 14:41:40.123
_orig/00392.JPG;2009-12-27 14:41:45.456
_orig/00393.JPG;2009-12-27 14:41:49.999
```

You can see the option --time-offset=0:+1:-3:+123, this means the GPS times from autopilot trace has been offset to match generated jpgtimes.txt. This way we keep original time set by the camera. One hour has been added, three minutes subtracted and 123 seconds added; use any combination that is natural when examining logs and EXIF data visually. For determining time skew, it is best to synchronize your camera with GPS before the flight (using 'G' key and autopilot console, then resetting time of the camera as the new minute rolls over). After the flight, verify the timing is looking right at *trace.txt.TRIGn-001.csv* and your *jpgtimes.txt*.
Using --round-seconds increases matching precision as EXIF data rarely contains millisecond time.
The camera itself has its own clock, which can progressively drift away from autopilot (GPS) clock, regardless of initial agreement (this can be as high as 1s per day!). You will need to guess the new time offset per day, usually incrementing in one direction.

Observe that CalCam filename is provided, this string will appear in *trp* file (but it could be added manually later and you can simply omit that option).
Camera rotation: –camera1-rotation is counted clockwise 0-360deg for logdecode (it is translated to *EnsoMosaic* direction internally).
Altitude offset is 180m added from takeoff point to all altitude data, is also used for *Approximate_terrain_altitude* in *trp* file. Usually, use MSL altitude (above mean sea level using WGS84 earth model) displayed by Google Earth at takeoff location.
LogDecode will perform automatic line grouping for *EnsoMosaic* which could be adjusted in *trp* file manually.
Also, you can put full or relative path to image files in jpgtimes.txt, this will be appropriately structured in *trp* file.
The projection string is fixed and can be changed manually in *trp* file.

Example of generated ensomosaic.trp file:

```
Projection: KKJ 3 INT24 WGS84ED50
Calibration_file: C:\EnsoMOSAIC_trial\Camera\Canon_PowerShot-S45_456_Trigger_110209_C.cal
Ground_control_point_file: no.gcp
Approximate_terrain_altitude: 128.0
Camera_rotation: 0.0
Map_files: 0
    1     1 00277.JPG _orig\
    1     2 00278.JPG
    1     3 00279.JPG
    1     4 00280.JPG
    1     5 00281.JPG
```

Example of generated ensomosaic.gps file:

```
    1     1   17.123456 51.123456 249.9 212.0 Dec 20 10:51:25 2009 220.9   3.0  -5.0 ypr
    1     2   17.123456 51.123456 249.5 209.0 Dec 20 10:51:30 2009 217.7   3.0  -9.0 ypr
    1     3   17.123456 51.123456 249.0 212.0 Dec 20 10:51:34 2009 220.3   2.0   3.0 ypr
    1     4   17.123456 51.123456 249.0 210.0 Dec 20 10:51:39 2009 218.4   3.0  -5.0 ypr
    1     5   17.123456 51.123456 248.6 219.0 Dec 20 10:51:43 2009 225.8   2.0  -1.0 ypr
```

With the minimum options provided, those files enable immediate processing with *EnsoMosaic* provided the flight parameters generated acceptable quality of pictures.

Using LogDecode for planning the mission:

In order to verify the flight plan, issue @@@WPTITERALL command on the autopilot and capture the output using serial console. You can do it automatically using log download tool:

```
logread921K.exe %COMPORT_FLEXIPILOT_USB% @@@WPTITERALL > wptiter.txt
```

(Note the system variable which value is 5, 12 or anything depicting your serial port number). You can write this command into .bat file and just double-click it.

At this point you can decode wptiter.txt which typically looks like

```
#AREVE51.153114 22.427326 0#
#ARW111:(51.123456,22. 123456,250),0.000000,0.000000,250,0,7,0#
#AREVE51. 123456m22. 1234560#
#ARW112:(51.123456,22.123456,250),3090.000000,0.300000,250,0,6,0#
```

Decoding it using the simplest syntax

```
logdecode.exe wptiter.txt
```

is generating the kml file that you can view in Google Earth.

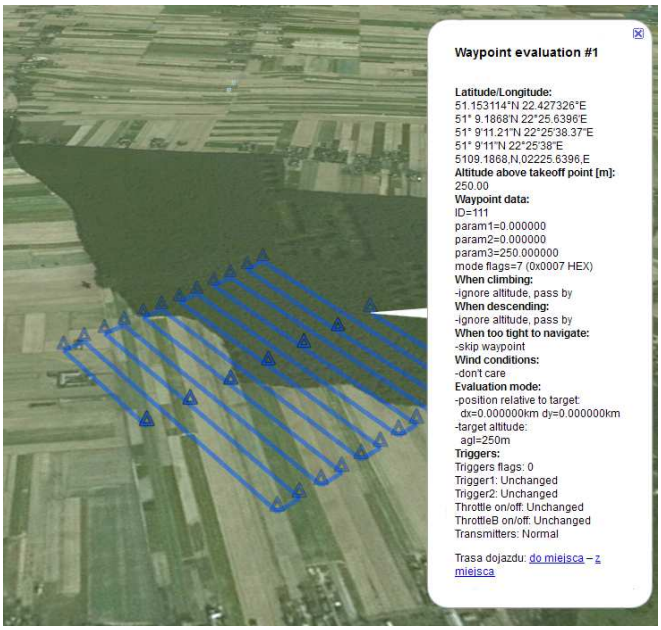
If you adjust your display for planned takeoff altitude…

```
logdecode.exe wptiter.txt  --altitude-offset=582
```

…and put everything into one bat file

```
logread921K.exe %COMPORT_FLEXIPILOT_USB% @@@WPTITERALL > wptiter.txt
logdecode.exe wptiter.txt --altitude-offset=582
```

You have a solution that view the next planned mission in a few seconds.
It is particularly useful as a part or pre-flight checking.



A mission preview in Google Earth takes just a few seconds

TRACE content

#ART00000:000391.820,+51.123456,+017. 123456,+00000.1,-023.3,+005.6,037.3,001,2011-03-06,19:35:00.820,00000#

Each trace event contains the following information:
-event number
-time of the day in s
-latitude in degrees
-longitude in degrees
-altitude above takeoff point in meters
-roll (positive is right)
-pitch (positive is up)
-course
-UTC date
-UTC time
-event type ID

Note: roll and pitch angles are concerning the airplane.
If roll or pitch stabilized video head is present, the data roll and pitch still contain the fuselage orientation.
In red: Critical errors.

Event type ID might be one of the following:

0	Periodically saved data
1	Trigger one activated
2	Trigger two activated
3	Waypoint switch position
4	Waypoint target position
6	Automatic takeoff terminated
7	Disabling triggers
8	RETHOME: Disabling MIN AGL testing
9	RETHOME: Enabling MIN AGL testing
10	RETHOME: MIN AGL detected
11	RETHOME: MAX AGL detected
12	RETHOME: MAXDIST detected
13	RETHOME: MAXLOSDIST detected
14	RETHOME: MAXTIME detected
15	RETHOME: RX MIN detected
16	RETHOME: RX MAX detected
17	RETHOME: MINLOSANGLE detected
18	RETHOME: EXCLZONE detected
19	RETHOME: ALLOWZONE detected
20	RETHOME: GPSCONN detected
21	RETHOME: GPSLOCK detected
22	RETHOME: MAXTIME2BASE detected
23	RETHOME: TIMERESERVE2BASE detected
24	RETHOME: MAXTRIP detected, too long flight measured over ground
25	RETHOME: next waypoint outside distance limits, too far
26	RETHOME: VOLTMIN1 voltage too low
27	RETHOME: VOLTMIN2 voltage too low
28	RETHOME: HEARTBEAT TIMEOUT (ping not received)
29	RETHOME: ZONE (outside allowed zone)
30	RESET DETECTED (autopilot reset, critical anomaly)
31	WDRF DETECTED (watchdog: autopilot reset, critical anomaly)
32	Entering low power mode (servo actions reduced)
33	Engine low power mode
34	Engine cutoff before parachute deployment
35	PARACHUTE: armed MINALT testing (above minimal altitude)
36	PARACHUTE: decided to open because of MINALT
37	PARACHUTE: deployed at waypoint
38	PARACHUTE: armed MINALT testing waypoint
39	PARACHUTE: deployed after descent (below min alt for opening)
40	PARACHUTE: deployed by RC
41	PARACHUTE: deployed by console/modem
42	Throttle enabled
43	Throttle disabled
44	ThrottleB enabled
45	ThrottleB disabled
46	Takeoff Voltage too low, aborting

47	PARACHUTE: logic reset by RC, latch closed
48	PARACHUTE: disarmed at waypoint
49	PARACHUTE: deployed because of failed GPS
50	PARACHUTE: deployed because of max distance
51	PARACHUTE: deployed because of leaving ZONE list
52	RETHOME: issued command by console or modem
54	TAKEOFF reference point here
55	LOITER reference point here
56	PARACHUTE: deployed using @@@PARA
57	RETHOME: AHCONSUMED1 (battery empty, using AMP1)
58	RETHOME: AHCONSUMED2 (battery empty, using AMP2)
59	RETHOME: RPM1 below minimum
60	RETHOME: RPM1 above maximum
61	Radio silence enabled: modem off until next waypoint
62	Radio silence disabled: modem will remain on
63	RXOVR: RC override enabled (manual control is possible)
64	RXOVR: RC override disabled
65	RETHOME: AMP1MAX exceeded
66	RETHOME: AMP2MAX exceeded
67	RETHOME: WATT1MAX exceeded
68	RETHOME: WATT2MAX exceeded
69	MODE: MANUAL
70	MODE: AUTO
71	NAVIMODE: MISSION
72	NAVIMODE: LOITER
73	NAVIMODE: BACK2TAKEOFF
74	RETHOME: AHRESERVE1 (battery reserve critical, using AMP1)
75	RETHOME: AHRESERVE2 (battery reserve critical, using AMP2)
76	RETHOME: WHCONSUMED1 (battery empty, using AMP1)
77	RETHOME: WHCONSUMED2 (battery empty, using AMP2)
78	RETHOME: WHRESERVE1 (battery reserve critical, using AMP1)
79	RETHOME: WHRESERVE2 (battery reserve critical, using AMP2)
80	RETHOME: MAX MSL detected
81	RETHOME: MAX FL detected
82	RETHOME: TARGET MAX AGL detected
83	RETHOME: TARGET MAX MSL detected
84	RETHOME: TARGET MAX FL detected
85	RETHOME: KEEP CLIMB ABOVE AGL violation
86	MISSION SELECT
87	RETHOME: MIN AGL detected, disabling climbing (NOCLIMB)
101…356	takeoff for flight number N, where N=ID-100

LOG content

The values in **bold** are of particular importance for flight analysis. Note that the time is not saved, but the maximum rate is 20Hz/LOG_PERIOD what allows calculating relative time. logNNN.txt file contains the following information, stored in floating point or decimal format:

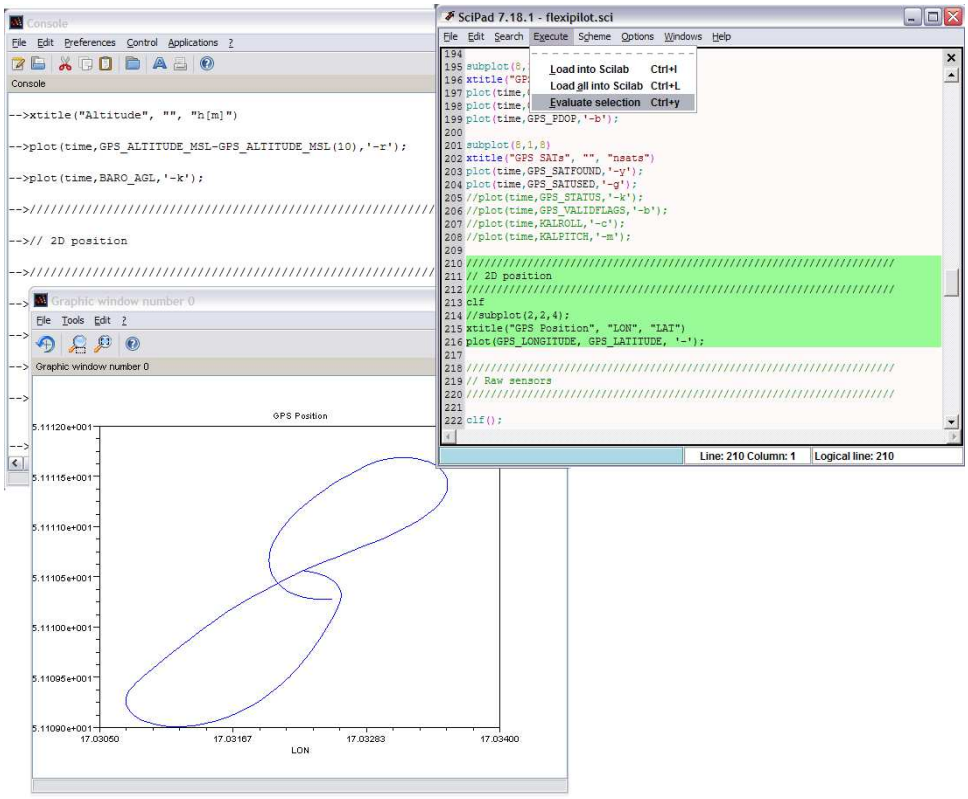
Field number	Typical value	Field name	Units and description
1	391	ADC0	
2	64	ADC1	
3	570	ADC2	
4	460	ADC3	
5	466	ADC4	
6	622	ADC5	
7	556	ADC6	
8	542	ADC7	
9	51.123456	**GPS_LATITUDE**	In decimal format
10	17.123456	**GPS_LONGITUDE**	In decimal format
11	19.26	GPS_SPEED	km/h
12	19.26	NAVI_SPEED3D_SMOOTH	km/h
13	12.61	NAVI_GROUNDSPEED_SMOOTH	km/h
14	350.2	**DIRECTION TARGET** (turnrate or rollangle)	degrees/s or degrees
15	354.9	GPS_COURSE	degrees
16	354.9	**NAVI_COURSE_SMOOTH**	degrees
17	110	IMU_GYRO_YAWANGLE	degrees
18	110.1	IMU_GYRO_YAWANGLE_SMOOTH	degrees
19	-20	**TARGET_TURNRATE**	degrees/s
20	-22.5	ALTITUDE_TARGET (Climbrate or Pitch)	degrees or m/s
21	-22.5	**NAVI_TURNRATE**	degrees/s
22	5.4	**NAVI_TURNRATE_SMOOTH**	degrees/s
23	5.4	IMU_YAWRATE	degrees/s
24	129.3	**GPS_ALTITUDE_MSL**	m
25	170.4	GPS_ALTITUDE_WGS	m
26	0.09	GPS_CLIMBRATE	m/s
27	27	GPS_STATUS	flags
28	9	**GPS_SATUSED**	Number from GPGGA
29	9	GPS_SATFOUND	Number from GPGSA
30	62	GPS_VALIDFLAGS	flags
31	1.09	**GPS_HDOP**	m
32	1.92	**GPS_VDOP**	m
33	2.21	**GPS_PDOP**	m
34	0.064697	GPS_AGE	s age of last received GPS message
35	0.064697	GPS_AGE_LASTPARSE	s age of last position update
36	15690	PWMOUT1	PWM pulse width , 10000=1ms=LOW
37	18196	PWMOUT2	PWM pulse width , 10000=1ms=LOW
38	0	PWMOUT3	PWM pulse width , 10000=1ms=LOW
39	14453	PWMOUT4	PWM pulse width , 10000=1ms=LOW
40	19997	PWMOUT5	PWM pulse width , 10000=1ms=LOW
41	19997	PWMOUT6	PWM pulse width , 10000=1ms=LOW
42	9979	BARO_PRES	Pa
43	32.62	**BARO_TEMP**	C temp of internal barometric sensor and board
44	31.64	BARO_PRES_RAW	Pa
45	128.9	**TARGET_AGL**	m
46	1.42417	**BARO_AGL**	m
47	0.37	**BARO_CLIMBRATE**	m/s
48	3.84	ACCELROLL	degrees from accelerometer, positive is right
49	-1.98	**ROLL**	degrees, positive is right
50	-24.1	NAVI_GPS_ROLLANGLE	degrees, positive is right

51	-8.03	NAVI_GPSSPD_ROLLANGLE	degrees, positive is right
52	-4.21	IMU_YAWGYRO_ROLLANGLE	degrees, positive is right
53	-2.45	IMU_YAWGYROSPD_ROLLANGLE	degrees, positive is right
54	29.91	ACCELPITCH	degrees from accelerometer, positive is up
55	27.65	**PITCH**	degrees, positive is up
56	15698	**SCAP0**	PWM pulse width , 10000=1ms=LOW
57	18206	**SCAP1**	PWM pulse width , 10000=1ms=LOW
58	9652	**SCAP2**	PWM pulse width , 10000=1ms=LOW
59	14461	**SCAP3**	PWM pulse width , 10000=1ms=LOW
60	0	**SCAP4**	PWM pulse width , 10000=1ms=LOW
61	0	**SCAP5**	PWM pulse width , 10000=1ms=LOW
62	0	**SCAP6**	PWM pulse width , 10000=1ms=LOW
63	15175	**SCAP7**	PWM pulse width , 10000=1ms=LOW
64	20834	**SCAP8**	PWM pulse width , 10000=1ms=LOW
65	0	**SCAP9**	PWM pulse width , 10000=1ms=LOW
66	0	**SCAP10/RPM1 scaled**	PWM pulse width or scaled RPM1
67	0	**SCAP11**	PWM pulse width - AP enable
68	0	**SCAP12**	reserved
69	0	**SCAP13**	reserved
70	1104	**SCAP14**	live timer 0-65535
71	0	**SCAP15**	reserved
72	10.61	AIRSPEED_KMH from Pitot	km/h
73	12.99	**AIRSPEED_KMH_SMOOTH** from Pitot	km/h
74	3.85	**NAVI_GROUNDTRIP**	km
75	0	**WAYPOINT_GROUNDTRIP**	km
76	0	**WAYPOINT_GROUNDTRIP_PENDING**	km
77	0	TIMING_PROBLEM	Subcomponent error flags
78	0.01	ACCELX_STATIC	G's
79	0.00	ACCELY_STATIC	G's
80	1.02	ACCELZ_STATIC	G's
81	0.02	ACCELX_CORR	G's minus centrifugal accel
82	0.05	ACCELY_CORR	G's minus centrifugal accel
83	0.96	ACCELZ_CORR	G's minus centrifugal accel
84	67	ROLLRATE	deg/s
85	12	PITCHRATE	deg/s
86	13	YAWRATE	deg/s
87	279	WIND_WHDG	deg
88	12.2	WIND_WSPEED	km/h
89	49.7	WIND_CSPEED	km/h groudnspeed minus wind speed avg
90	56.7	WIND_AIRSPEED2D	km/h groundspeed minus wind speed, flat
91	17.8	HUMI_TEMP	C temp from humidity sensor
92	-2.12	HUMI_DEWPOINT	C
93	-3.14	HUMI_FROSTPOINT	C
94	47	HUMI_RH	0-100% relative humidity
95	250	HUMI_ALTI_STD	m AMSL Baro altitude ISA model
96	262	HUMI_ALTI_DENS	m AMSL density altitude vs ISA model
97	3456	HUMI_ALTI_CLOUDBASE	m AMSL cloud base altitude
98	4561	HUMI_ALTI_FROSTALT	m AMSL frost altitude
99	11.23	VOLTAGE1	V input #1
100	0.55	VOLTAGE2	V input #2
101	0	AMP1	A input #1 Volts scaled to Amperes
102	3.12	AMP2	A input #2 Volts scaled to Amperes
103	-13	MAGX	Magnetometer X RAW reading
104	123	MAGY	Magnetometer Y RAW reading
105	1234	MAGZ	Magnetometer Z RAW reading
106	3120	MAHCONSUMED1	mAh consumed by battery 1
107	0	MAHCONSUMED2	mAh consumed by battery 2
108	34	WIND_HDG	Airplane heading from wind estimator
109	1234.5	TOTEN	Total energy/mass
110	1200	TOTENAVG	Total energy/mass averaged

111	25023.17	GPS_UTC	s time of day
112	124.67	CLOCK	s autopilot clock time since booting
113	0.11	EXTREMES_GX	max G factor along wings
114	0.21	EXTREMES_GY	max G factor along fuselage
115	1.32	EXTREMES_GZ	max vertical G factor
116	40.2	EXTREMES_GYROXRATE	deg/s max rollrate
117	14.5	EXTREMES_GYROYRATE	deg/s max pitchrate
118	20.2	EXTREMES_GYROZRATE	deg/s max yawrate
119	11.2	EXTREMES_MINVOLT1	Min recorded voltage since last iteration
120	11.1	EXTREMES_MINVOLT2	Min recorded voltage since last iteration
121	0.0	EXTREMES_MAXAMP1	Max recorded amperage since last iteration
122	3.16	EXTREMES_MAXAMP2	Max recorded amperage since last iteration

SCILAB

Scilab is a free and very powerful data analysis tool. However, the basic operations of reading the data from files and plotting them are very simple. You can open **flexipilot.sci** file in Scilab and execute it section by section, analyzing different aspect of the flight.



Scilab allows partial execution of the code

Before using the script, you must adjust 2 things.
As the log file does not contain time, you must provide correct value of **LOG_PERIOD**.
Depending on your autopilot version, use the correct loop update rate **MAINLOOP_FREQUENCY**, typically 20Hz.
You also have to modify the input file path: `logdata=read('D:\Logs\.txt.LOG001.txt', -1, 120);`
The frequency is shown in log.txt as the last position after #ARF

The first part of the script is essentially preparing the named vectors with correct data labels.

```
//////////////////////////////////////////////////////////////////////////
clear
logdata=read('D:\Logs\log.txt.LOG001.txt', -1, 108);
LOG_PERIOD=4;
MAINLOOP_FREQUENCY=25;

time=logdata(:,1);
time(1)=0;
for  i = 2:length(time),
        time(i)=time(i-1)+LOG_PERIOD/MAINLOOP_FREQUENCY;
end;

ADC0=logdata(:,1);
ADC1=logdata(:,2);
ADC2=logdata(:,3);
ADC3=logdata(:,4);
```

```
ADC4=logdata(:,5);
ADC5=logdata(:,6);
ADC6=logdata(:,7);
ADC7=logdata(:,8);
ADC_ACCELX=ADC6;
ADC_ACCELY=ADC7;
ADC_ACCELZ=ADC5;

GPS_LATITUDE=logdata(:,9);
GPS_LONGITUDE=logdata(:,10);

GPS_LATITUDE_NOTZERO=0;
for  i = 1:length(GPS_LATITUDE)
        if(GPS_LATITUDE(i)<>0) then
                GPS_LATITUDE_NOTZERO=GPS_LATITUDE(i);
                break;
        end
end
for  i = 1:length(GPS_LATITUDE)
        if(GPS_LATITUDE(i)<>0) then
                break;
        else
                GPS_LATITUDE(i)=GPS_LATITUDE_NOTZERO;
        end
end

GPS_LONGITUDE_NOTZERO=0;
for  i = 1:length(GPS_LONGITUDE)
        if(GPS_LONGITUDE(i)<>0) then
                GPS_LONGITUDE_NOTZERO=GPS_LONGITUDE(i);
                break;
        end
end
for  i = 1:length(GPS_LONGITUDE)
        if(GPS_LONGITUDE(i)<>0) then
                break;
        else
                GPS_LONGITUDE(i)=GPS_LONGITUDE_NOTZERO;
        end
end

GPS_SPEED=logdata(:,11);
NAVI_GROUNDSPEED3D_SMOOTH=logdata(:,12);
NAVI_GROUNDSPEED_SMOOTH=logdata(:,13);

TARGET_COURSE=logdata(:,14);
NAVI_COURSE=logdata(:,15);
NAVI_COURSE_SMOOTH=logdata(:,16);
IMU_GYRO_YAWANGLE=logdata(:,17);
IMU_GYRO_YAWANGLE_SMOOTH=logdata(:,18);

TARGET_TURNRATE=logdata(:,19);
NAVI_TURNRATE=logdata(:,20);
NAVI_TURNRATE_SMOOTH=logdata(:,21);
IMU_GYRO_YAWRATE=logdata(:,22);
IMU_GYRO_YAWRATE_SMOOTH=logdata(:,23);

GPS_ALTITUDE_MSL=logdata(:,24);
ZERO_GPS_ALTITUDE_MSL=GPS_ALTITUDE_MSL-GPS_ALTITUDE_MSL(2);
GPS_ALTITUDE_WGS=logdata(:,25);
GPS_CLIMBRATE=logdata(:,26);

GPS_STATUS=logdata(:,27);
GPS_SATUSED=logdata(:,28);
GPS_SATFOUND=logdata(:,29);
GPS_VALIDFLAGS=logdata(:,30);
GPS_HDOP=logdata(:,31);
GPS_VDOP=logdata(:,32);
GPS_PDOP=logdata(:,33);
GPS_AGE=logdata(:,34);
GPS_AGE_LASTPARSE=logdata(:,35);

PWMOUT0=logdata(:,36);
```

```
PWMOUT1=logdata(:,37);
PWMOUT2=logdata(:,38);
PWMOUT3=logdata(:,39);
PWMOUT4=logdata(:,40);
PWMOUT5=logdata(:,41);

BARO_PRES=logdata(:,42);
BARO_TEMP=logdata(:,43);
AIRTEMP=logdata(:,44);

TARGET_AGL=logdata(:,45);
BARO_AGL=logdata(:,46);
BARO_CLIMBRATE=logdata(:,47);
//detrend(BARO_AGL);

GRAVROLL=logdata(:,48);
KALROLL=logdata(:,49);
NAVI_GPS_ROLLANGLE=logdata(:,50);
NAVI_GPSSPD_ROLLANGLE=logdata(:,51);
IMU_YAWGYRO_ROLLANGLE=logdata(:,52);
IMU_YAWGYROSPD_ROLLANGLE=logdata(:,53);

GRAVPITCH=logdata(:,54);
KALPITCH=logdata(:,55);

SCAP0=logdata(:,56);
SCAP1=logdata(:,57);
SCAP2=logdata(:,58);
SCAP3=logdata(:,59);
SCAP4=logdata(:,60);
SCAP5=logdata(:,61);
SCAP6=logdata(:,62);
SCAP7=logdata(:,63);
SCAP8=logdata(:,64);
SCAP9=logdata(:,65);
SCAP10=logdata(:,66);
SCAP11=logdata(:,67);
SCAP12=logdata(:,68);
SCAP13=logdata(:,69);
SCAP14=logdata(:,70);
SCAP15=logdata(:,71);

AIRSPEED_KMH=logdata(:,72);
AIRSPEED_KMH_SMOOTH=logdata(:,73);
NAVI_GROUNDTRIP=logdata(:,74);
WAYPOINT_GROUNDTRIP=logdata(:,75);
WAYPOINT_GROUNDTRIP_PEND=logdata(:,76);

TIMING_PROBLEM=logdata(:,77);

ACCELX_STATIC=logdata(:,78);
ACCELY_STATIC=logdata(:,79);
ACCELZ_STATIC=logdata(:,80);
ACCELX_CORR=logdata(:,81);
ACCELY_CORR=logdata(:,82);
ACCELZ_CORR=logdata(:,83);
ROLLRATE=logdata(:,84);
PITCHRATE=logdata(:,85);
YAWRATE=logdata(:,86);

WIND_WHDG=logdata(:,87);
WIND_WSPEED=logdata(:,88);
WIND_CSPEED=logdata(:,89);
WIND_AIRSPEED2D=logdata(:,90);

HUMI_TEMP=logdata(:,91);
HUMI_DEWPOINT=logdata(:,92);
HUMI_FROSTPOINT=logdata(:,93);
HUMI_RH=logdata(:,94);
HUMI_ALTI_STD=logdata(:,95);
HUMI_ALTI_DENS=logdata(:,96);
HUMI_ALTI_CLOUDBASE=logdata(:,97);
HUMI_ALTI_FROSTALT=logdata(:,98);
```

```
VOLTAGE1=logdata(:,99);
VOLTAGE2=logdata(:,100);
AMP1=logdata(:,101);
AMP2=logdata(:,102);

MAGX=logdata(:,103);
MAGY=logdata(:,104);
MAGZ=logdata(:,105);

AHCONSUMED1=logdata(:,106)/1000;
AHCONSUMED2=logdata(:,107)/1000;

WIND_HDG=logdata(:,108);
```

This part saves a few selected data in a file for display with external software:

```
////////////////////////////////
//Save altitude file

unix('rm -f \alti.txt');
[fd,err]=mopen('\alti.txt', 'wt');
mfprintf(fd,'%f,%f,%f,%f,%f\n',[time,TARGET_AGL,BARO_AGL,ZERO_GPS_ALTITUDE_MSL,AIRTEMP]);
err=mclose([fd]);
```

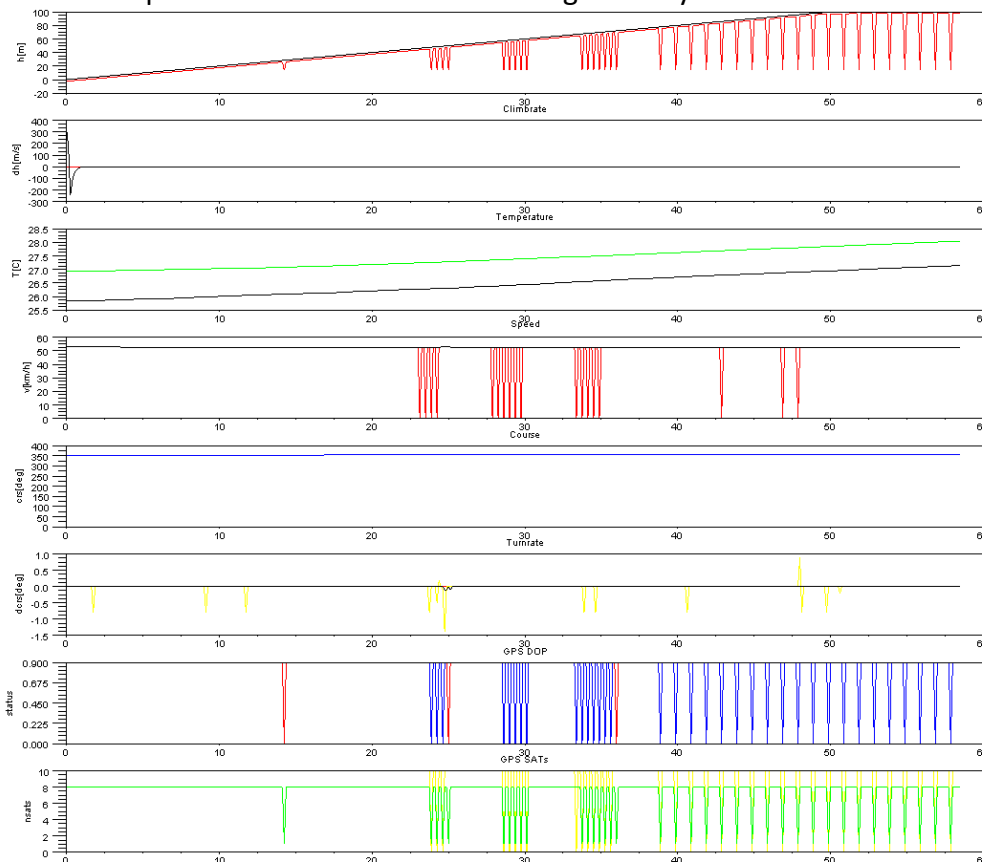
This part writes a Google Earth file:

```
////////////////////////////////
//Save Google Earth file

BARO_AGL(1)=BARO_AGL(3);
BARO_AGL(2)=BARO_AGL(3);
GPS_LATITUDE(1)=GPS_LATITUDE(3);
GPS_LATITUDE(2)=GPS_LATITUDE(3);
GPS_LONGITUDE(1)=GPS_LONGITUDE(3);
GPS_LONGITUDE(2)=GPS_LONGITUDE(3);

unix('rm -f \TestFlight.kml');
[fd,err]=mopen('\TestFlight.kml', 'wt');

mfprintf(fd,'<kml>\n');
mfprintf(fd,'<Document>\n');
mfprintf(fd,'<Placemark>\n');
mfprintf(fd,'<name>Test flight</name>\n');
mfprintf(fd,'<LineString>\n');
mfprintf(fd,'<altitudeMode>relativeToGround</altitudeMode>\n');
mfprintf(fd,'<coordinates>\n');

mfprintf(fd,'%f,%f,%f\n',[GPS_LONGITUDE,GPS_LATITUDE,BARO_AGL]);

mfprintf(fd,'</coordinates>\n');
mfprintf(fd,'</LineString>\n');
mfprintf(fd,'</Placemark>\n');
mfprintf(fd,'</Document>\n');
mfprintf(fd,'</kml>\n');

err=mclose([fd]);
```

This example displays 2D position map within Scilab:

```
////////////////////////////////////////////////////////////////////
// 2D position
////////////////////////////////////////////////////////////////////
clf
xtitle("GPS Position", "LON", "LAT")
plot(GPS_LONGITUDE, GPS_LATITUDE, '-');
```

The most advanced parts allow simultaneous viewing of many variables:



Feel free to experiment with the data you need.

You can also execute the script by dragging-dropping the *.sci file on the console window.

Pix4D

You can process the data easily using either offline of online cloud processing service.
The following steps are necessary:

1. Download the logs from the autopilot (getlog.bat, supplied)
2. Download the jpg files from the camera
3. Inspect the files with image viewer, remove blurred photos
4. Run logdecode, note the flight number (there might be several flights per log), note the MSL altitude of the takeoff point

```
logdecode.exe wptiter.txt trace.txt log.txt trace.txt
```

5. **(skip if time offset known)** Make a list of jpeg times by running the following script under CYGWIN environment (exifdatconv is supplied):

```
(
for i in *.JPG
do
TIMEVAR=`exif --ifd=EXIF -t 0x9004 $i | grep -w Value:`
TIMEVARFILTERED=`exifdatconv.exe "$TIMEVAR"`
UTCTIMEVAR=`date -u +%s -d "$TIMEVARFILTERED"`
echo "$i;$TIMEVARFILTERED;$UTCTIMEVAR"
done
) > jpgtimes.txt
```

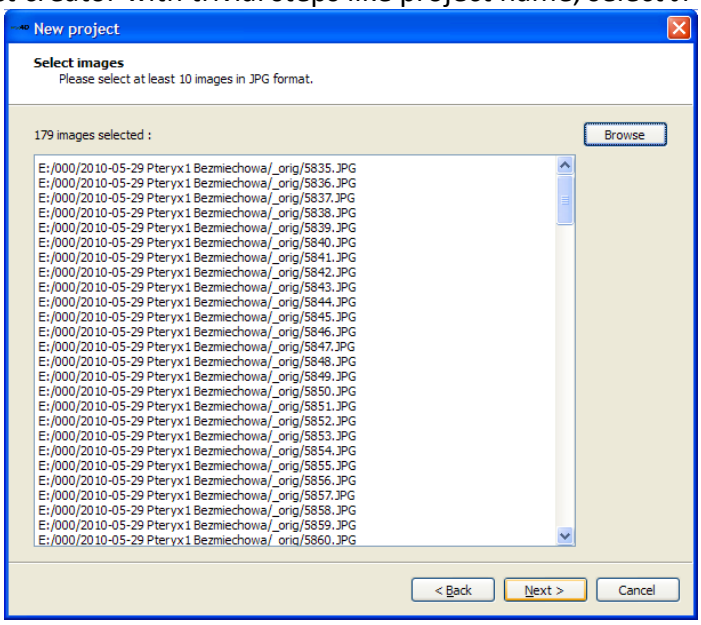
6. **(skip if time offset known)** Find time offset between your camera time and GPS UTC time found in `trace.txt.TRIGn-001.csv` file (this file contains the moments when camera shutter was activated)
7. Run logdecode again with camera rotation and time, example for Pteryx UAV with Canon S90:

```
logdecode.exe --round-seconds --time-offset=0:0:59:1 wptiter.txt
trace.txt log.txt trace.txt --wind-lower-alt=50 --camera1-aspect-
ratio=0.75 --camera1-lens-angle=39 --camera1-rotation=180 --camera1-
roll=0 --camera1-pitch=-9 --camera1-overlay-opacity=200 --altitude-
offset=138
--time-offset=0:0:59:1
Adjust the values until in trace.txt.TRIGn-001.csv you get same times as
in jpgtimes.txt.
--altitude-offset=138
Look at Google Earth, find takeoff position, adjust MSL takeoff altitude
--wind-lower-alt=50
Cut off wind estimation data below this altitude.
```

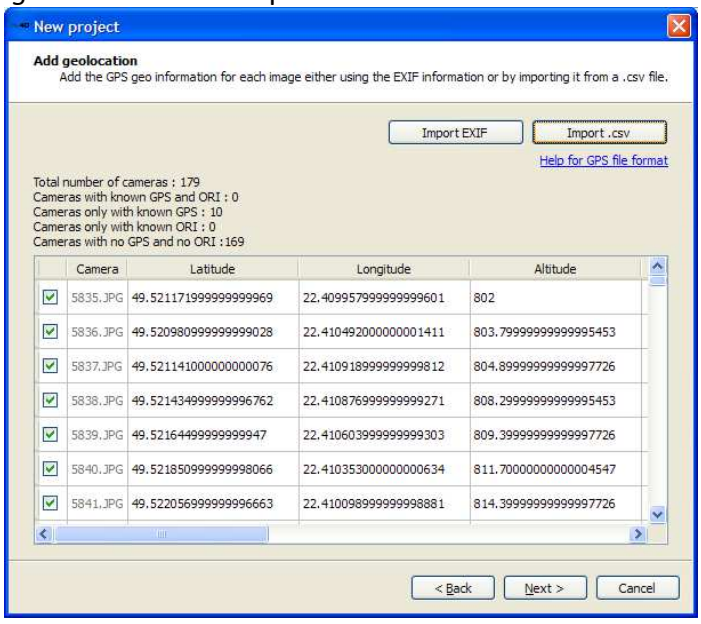
8. Run logdecode again with corrected camera rotation and time
   Note: another method to find time offset post factum is by experimenting with GeoSetter. Also one can reset the camera to GPS time before flight, this way the correction necessary will be know to be less than one minute, typically a few seconds what makes synchronizing even easier.

9. Run Pix4D launcher:

Follow NewProject creator with trivial steps like project name, select JPG files



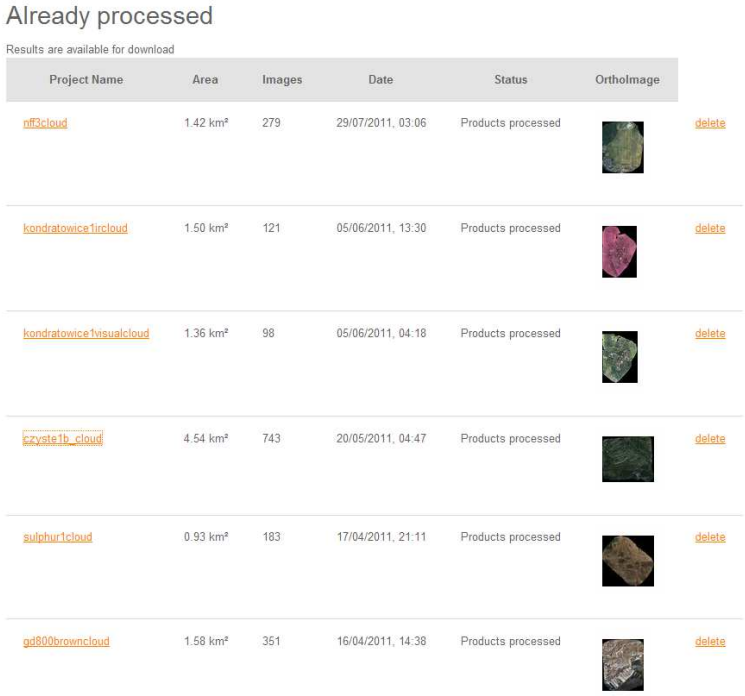
Then at stage *Add geolocation* click Import csv and select trace.txt.TRIG1-001.pix4d.dat



A synchronized list of position should display. Click next then Upload project, that's all for the manual part you need.

Note: you can also use GeoSetter for importing GPS positions into EXIF data that is written inside jpg files. This way you can completely ignore pix4d.dat file and click Import EXIF.

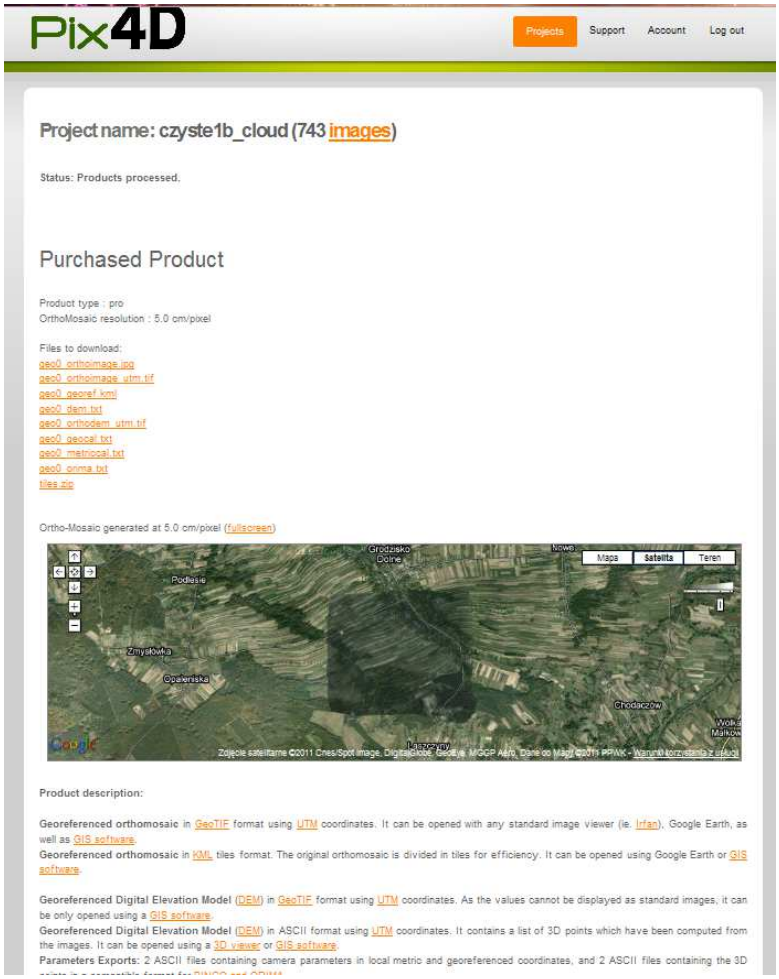
10. Log into your Pix4D account to select rendering resolution and view project status, download the results with web browser.



Already processed

Results are available for download

Project Name	Area	Images	Date	Status	OrthoImage	
nff3cloud	1.42 km²	279	29/07/2011, 03:06	Products processed		delete
kondratowice1ircloud	1.50 km²	121	05/06/2011, 13:30	Products processed		delete
kondratowice1visualcloud	1.36 km²	98	05/06/2011, 04:18	Products processed		delete
czyste1b_cloud	4.54 km²	743	20/05/2011, 04:47	Products processed		delete
sulphur1cloud	0.93 km²	183	17/04/2011, 21:11	Products processed		delete
gd800browncloud	1.58 km²	351	16/04/2011, 14:38	Products processed		delete



Pix4D

Projects Support Account Log out

Project name: czyste1b_cloud (743 images)

Status: Products processed.

Purchased Product

Product type : pro
OrthoMosaic resolution : 5.0 cm/pixel

Files to download:
geo0_orthoImage.jpg
geo0_orthoImage_utm.tif
geo0_georef.kml
geo0_dsm.txt
geo0_orthodem_utm.tif
geo0_geocal.txt
geo0_metricCal.txt
geo0_onma.txt
tiles.zip

Ortho-Mosaic generated at 5.0 cm/pixel (fullscreen)

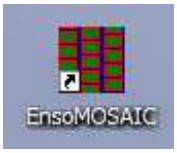
Product description:

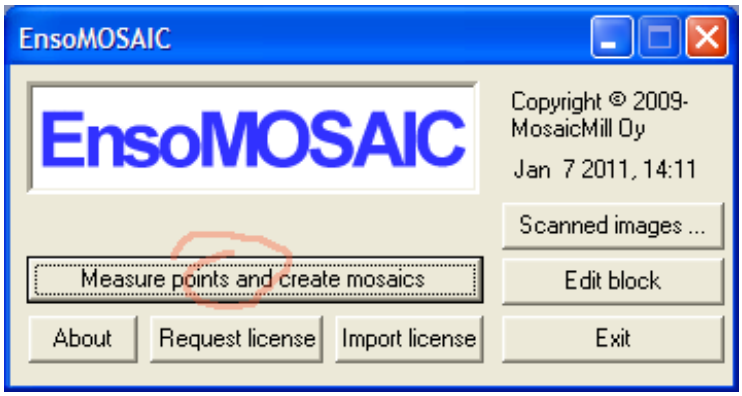
Georeferenced orthomosaic in GeoTIF format using UTM coordinates. It can be opened with any standard image viewer (ie. Irfan), Google Earth, as well as GIS software.
Georeferenced orthomosaic in KML tiles format. The original orthomosaic is divided in tiles for efficiency. It can be opened using Google Earth or GIS software.

Georeferenced Digital Elevation Model (DEM) in GeoTIF format using UTM coordinates. As the values cannot be displayed as standard images, it can be only opened using a GIS software.
Georeferenced Digital Elevation Model (DEM) in ASCII format using UTM coordinates. It contains a list of 3D points which have been computed from the images. It can be opened using a 3D viewer or GIS software.
Parameters Exports: 2 ASCII files containing camera parameters in local metric and georeferenced coordinates, and 2 ASCII files containing the 3D

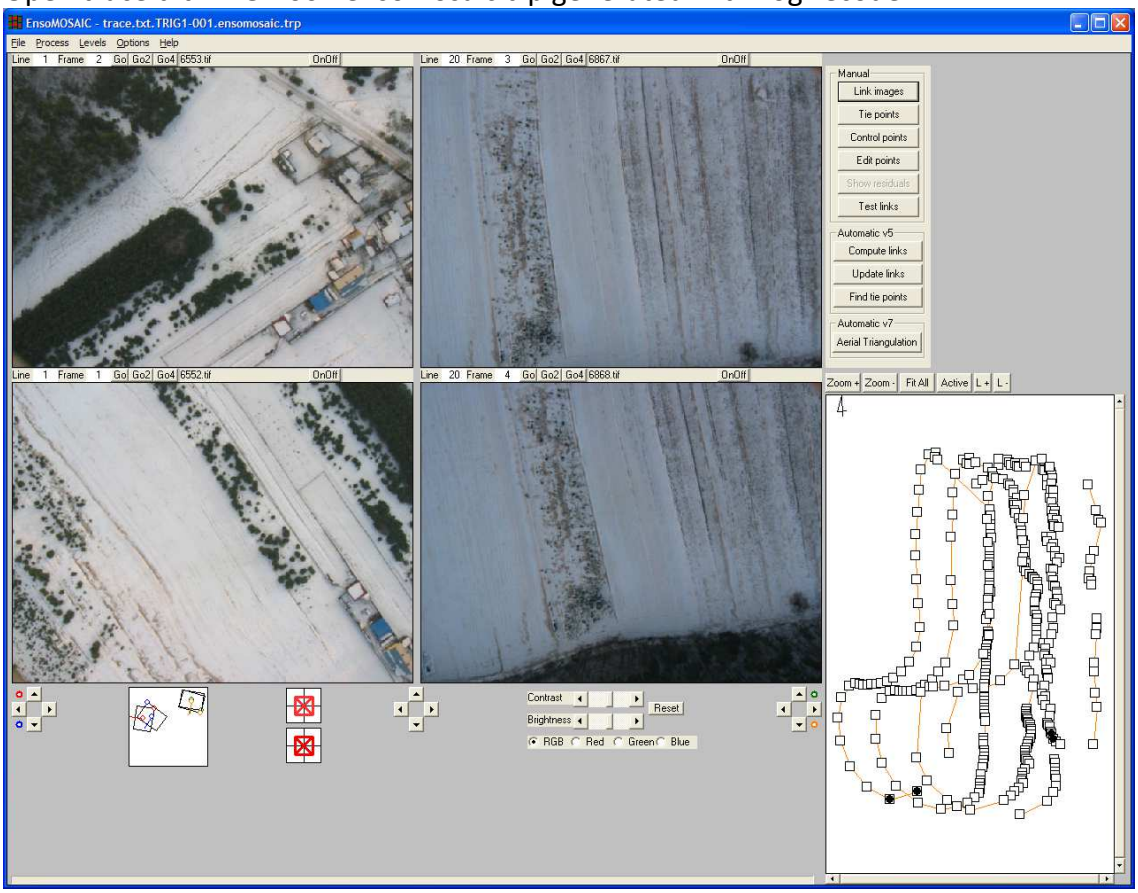
Enso Mosaic

Run Enso Mosaic

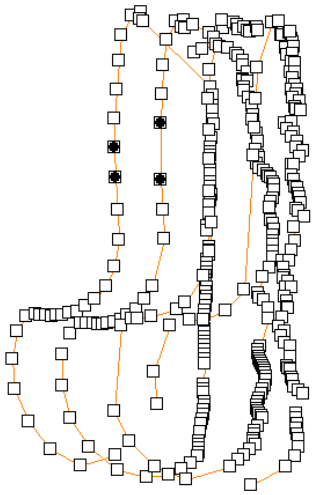




Open trace.txt.TRIG1-001.ensomosaic.trp generated with LogDecode:

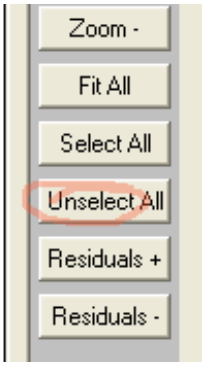


Use options:
Process/Check Image Status
Convert the jpg files into ‘pyramid TIFF’ using Process/Create pyramid Images
Verify that ‘flight lines’ have been defined correctly by LogDecode, if not, use Excel and rename lines inside trp file

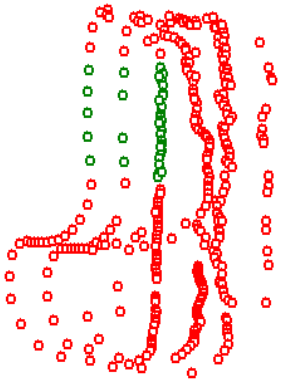


Click Aerial Triangulation v7

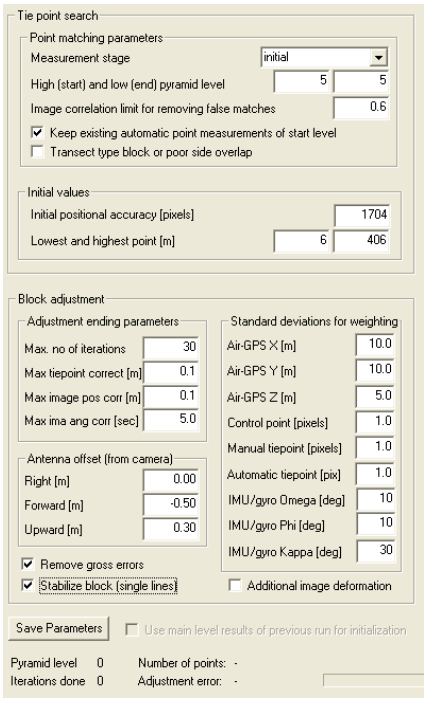
Unselect all mages



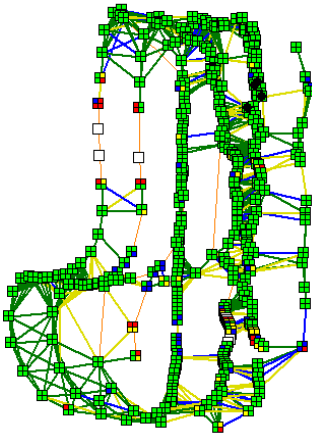
Select a sub region of points that looks most regular:



Enter most relaxed fitting parameters for initial fitting:
Select *Measurement stage: initial*
High-low pyramid level: 4…4
Select *Keep existing point measurements*
Increase IMU fitting precision to relaxed values:
Omega=10deg
Phi=10deg
Kappa=20deg (or more in strong winds when crab angles are high)
Enter antenna offset from camera, i.e.
Forward -0.5m
Upward 0.30m
Select *Stabilize block*

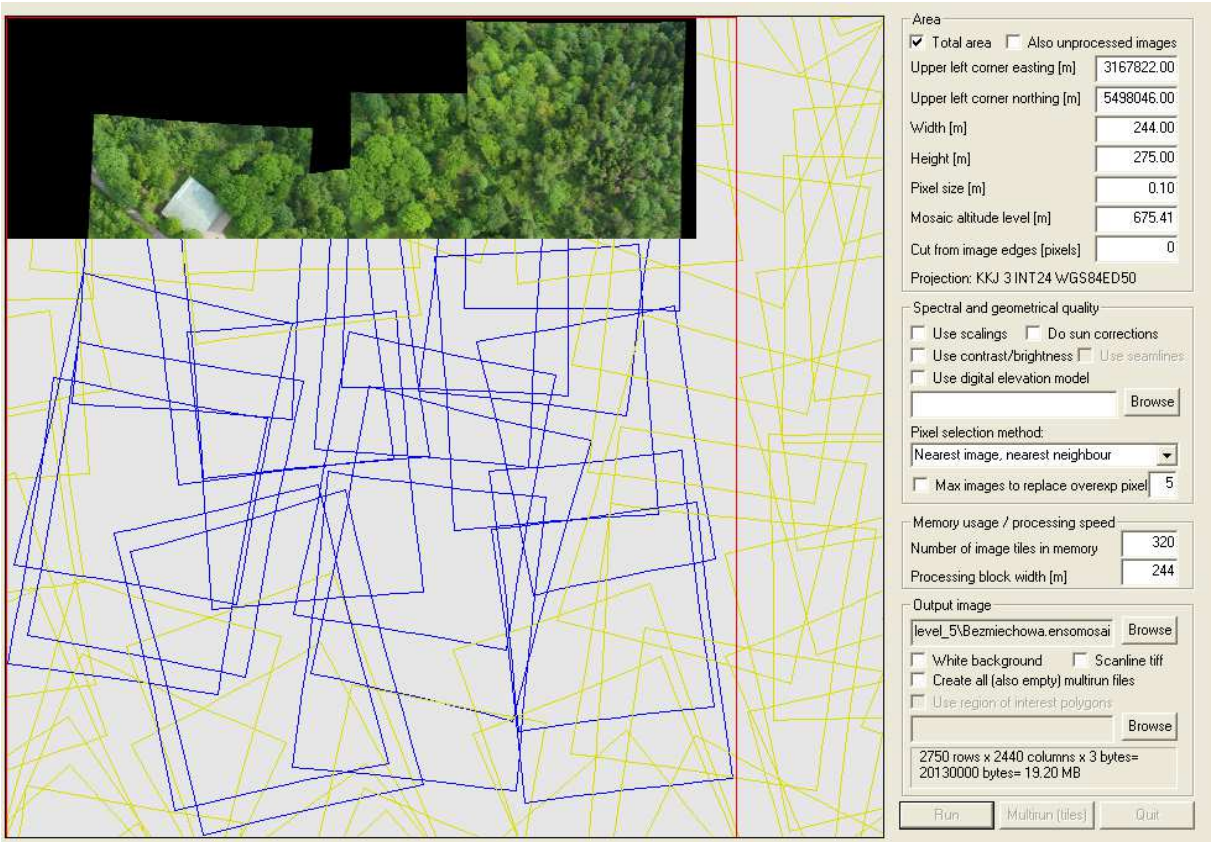


Click *Run* and observe the convergence
Retry several times until you got reasonable number of photos matched.
The final convergence might look like this:



At this point you are ready to call Process/Block Adjustment. This will move all photos collectively to correct GPS positions.

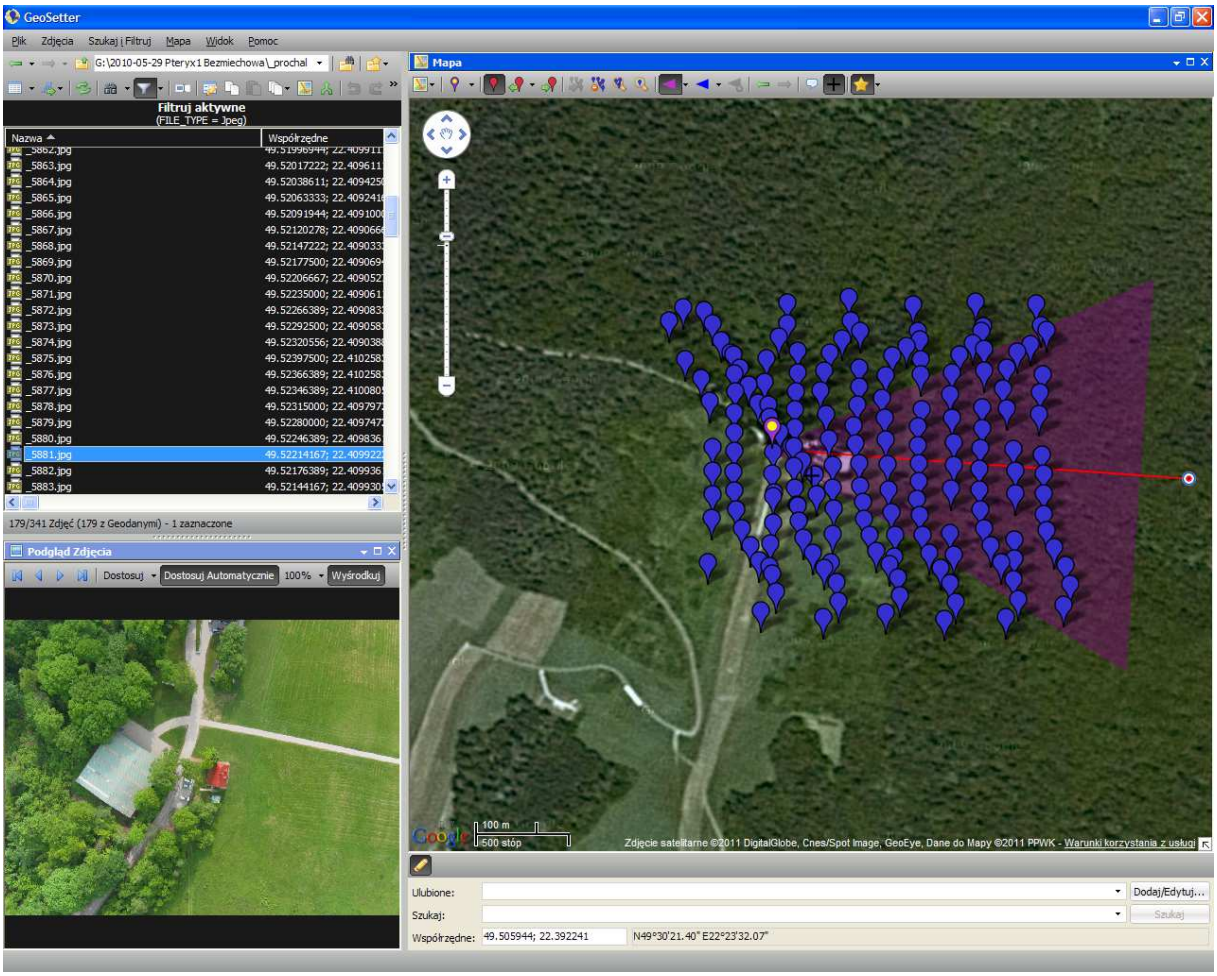
At this point you can Run Process/Create mosaic from original images.



The procedure is likely to involve several retries at triangulation stage.

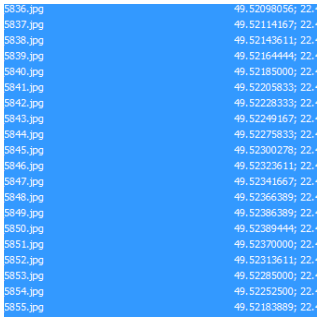
GeoSetter

This free application allows adding EXIF positional data to jpg files. The most important functionality is to keep the jpeg date/time unchanged, while synchronizing the image with gpx files generated by LogDecode.



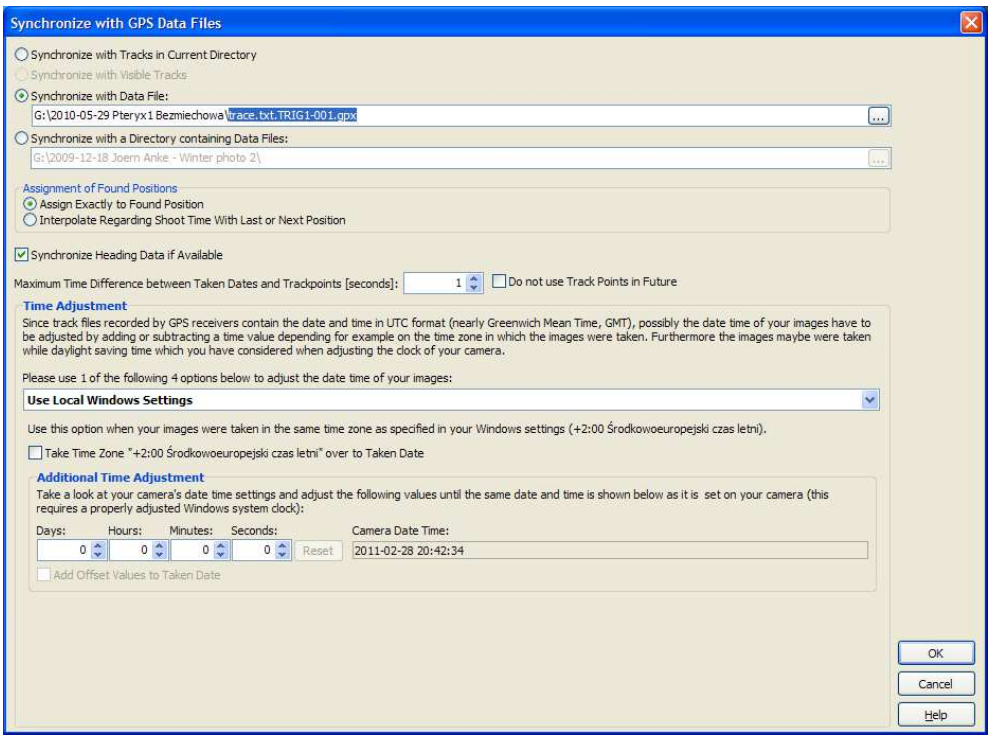
Using the preview, you can also verify that logdecode options have been correctly entered, in particular camera-rotation.

In order to perform geotagging,
Select the files



Select Images/Synchronize with GPS data files

Select the file with trigger data, for example
trace.txt.TRIG1-001.gpx



Note that the time zone added to jpeg time can destroy matching. Click OK and observe results:

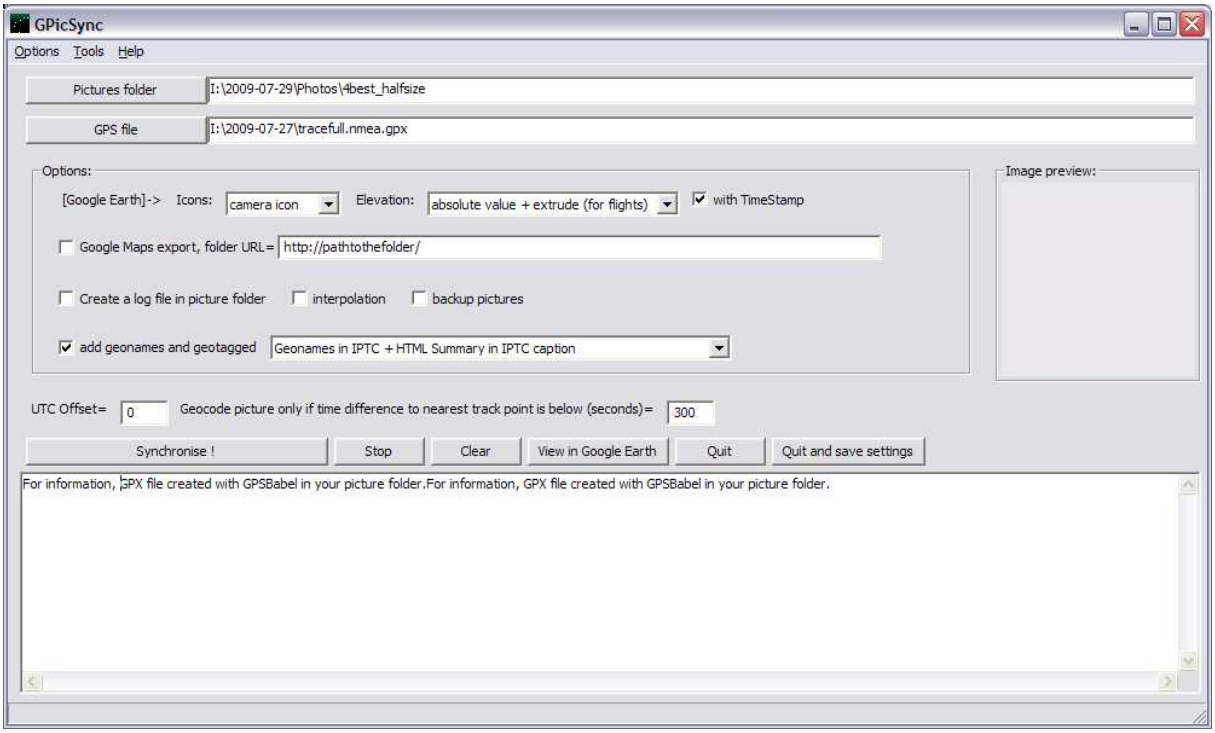
This means a failure, either time-offset parameter used for logdecode is completely wrong, or you need to adjust +1 or 2 hours using Additional Time Adjustment.
Click YES to try again.

This time it worked.

By clicking YES the locations will be updated.

Images/Edit Data is the next step.
Images/Save Images will save all changes.

GPicSync



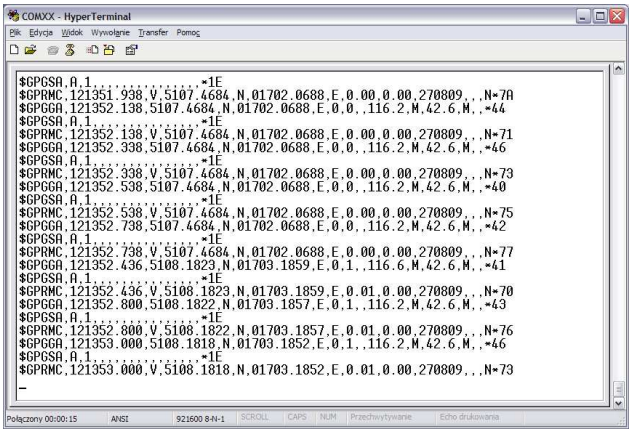
In order to match the pictures shot with your digital camera, you can use this free application. The presence of NMEA text files generated by LOGDECODE is crucial for this task. The screenshot represents the application set-up.

Clicking **GPS file** button/selector, you have to change to NMEA input file and select triggerN.nmea.txt. The GPX file is created immediately by GPicSync.

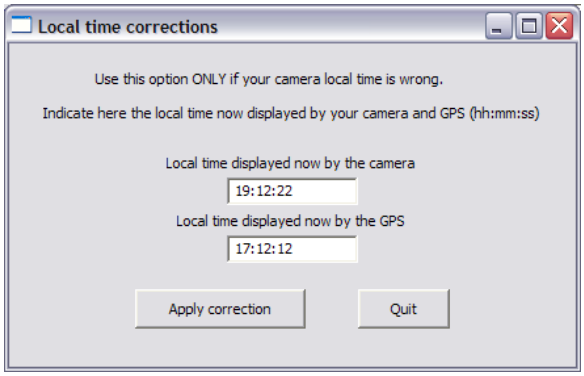
The files must be reasonably small, or the KML file will be excessively large. The processing could take several minutes. It is practical to reduce photo size automatically, before starting conversion. Consider using **ImageMagick Win2K Binaries, mogrify.exe** which is a perfect application for mass image resizing. Example syntax: **mogrify -resize 25% *.jpg**.
Warning: this command replaces original files.

Note: it is easy to additionally modify original jpeg timestamps. Start working on copies.
It is much better to use time-offset by LogDecode, as this way original data is preserved both for logs as for images.

Position matching algorithm uses the time written by your digital camera inside JPEG pictures. The camera clock and GPS clock are different by as much as several hours (different time zones) plus a few second inaccuracies. In order to know the precise clocks, open HyperTerminal and press ‘G’. This will output actual GPS clock. Make a photo with your digital camera and note the GPS time displayed at this moment (you can do this after the flight as well).



Using advanced picture viewer, look at the timestamp recorded inside the JPEG file that tells the camera’s time at the moment of making the photo. Enter both values in Options/Local time corrections window in GPicSync application:



Local time correction is necessary for accurate position matching

Photo mapping results as shown in Google Earth

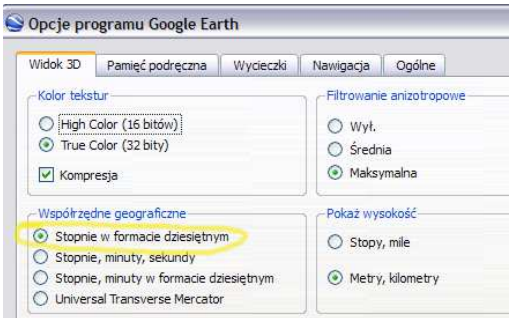
Note: since not all so-called geotagging software has intuitive time-adjustment options, you can instead adjust all log times using LogDecode tool supplied with Flexipilot. Also, you can round the autopilot times to nearest second what will make time-based matching slightly more accurate, particularly when there is small time spacing between consecutive photos.

Related free tools worth trying: Copiks PhotoMapper and GeoSetter. The latter allows embedding photo direction based on gpx file generated by LogDecode.

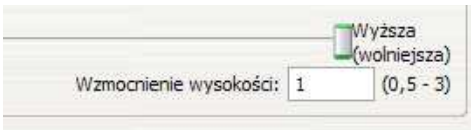
Accelerate generating geotagged names by subscribing to:
www.geonames.org

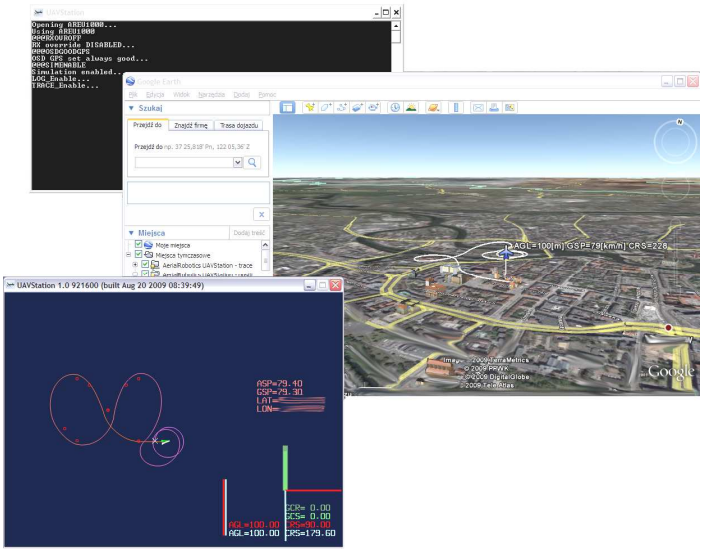
GoogleEarth

Follow the standard installation procedures. If you will scan the area of your flights before disconnecting from the internet and going to the filed, the application will remember the maps in its cache. However, for installation the Internet connection is necessary.
This free application is useful for both mission simulation and for viewing the flight logs parsed by LOGDECODE. Just drag-n-drop generated KML file on the application window.
Defining waypoints using maps is traditionally done with absolute coordinates. You can use Google Earth for this purpose. As the autopilot uses decimal latitude and longitude format, make sure to set decimal coordinate format in Google Earth menu Tools/Options/3D View tab.



In menu Tools/ Options…/3D View tab you can select altitude amplification to 1 in order to improve the perception of the track geometry.

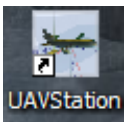


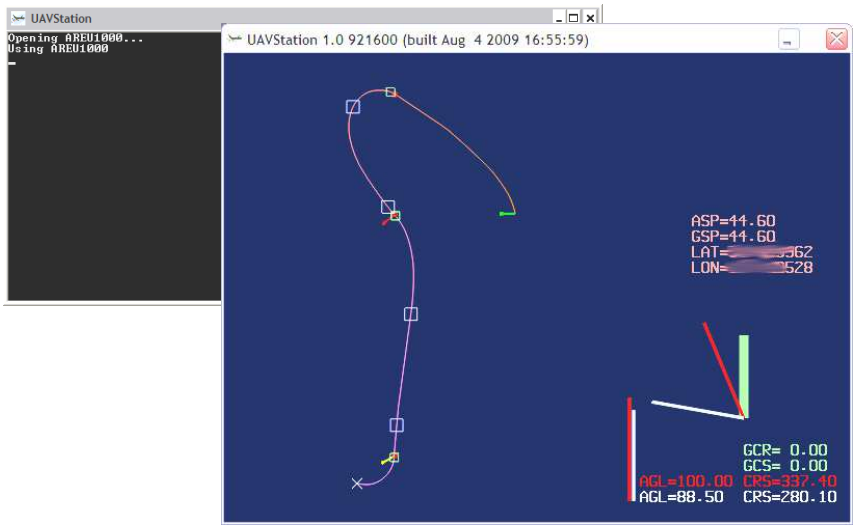


Typical simulation result with GoogleEarth

UAVStation

Close the HyperTerminal and launch the **UAVStation** console. This application is detecting the autopilot automatically, no need to know the COM port number. However, when the application is open, you must close the serial port terminal. The application is accepting typed text as long as the graphics window is active. You can also copy-paste text from notepad into the application.





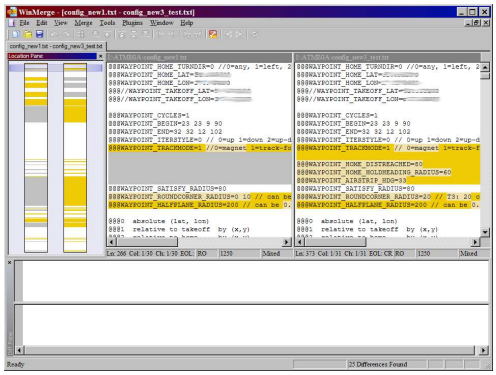
UAVStation provides the following key shortcuts:

F1 @@@ (without CRLF)
F2 @@@RXOVROFF
F3 @@@LOGDIR
F4 @@@SIMENABLE
F8 @@@LOGCLEAR and @@@TRACECLEAR
F12 @@@RESET
Esc Exit application

In general, UAVStation parses all data it can recognize (waypoint update during simulation, IMU angles etc.), while printing only not-recognized data (remaining statistics and error messages). This allows for cleaner view when using mission simulation. However, for downloading the log files, UAVStation must be closed. You can keep the UAVStation open for operations in the field; it will reconnect itself to the autopilot about 2s after plugging the FLEXIPILOT to USB port. The serial port terminals, on the other hand, must be disconnected (HyperTerm) or closed (Putty) at that time - even if they use different transmission channels on Windows Driver layer.

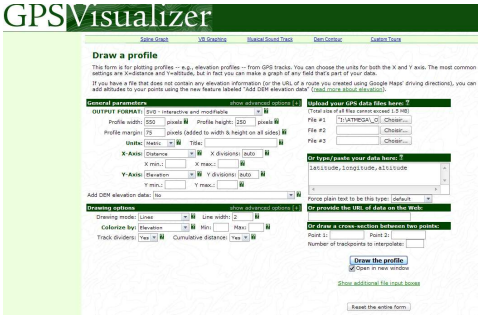
WinMerge

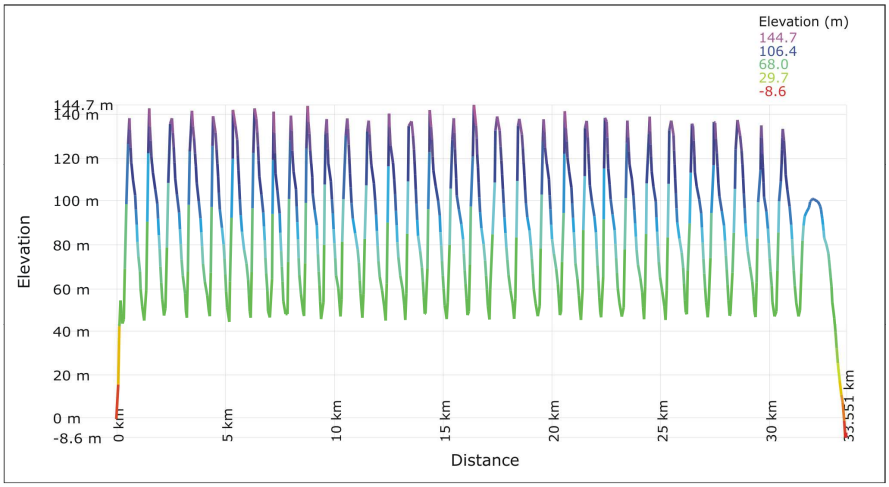
WinMerge is free, easy-to-use application for comparing ASCII files.
Because normally you are supposed to keep several versions of configuration files, you can synchronize them selectively using this intuitive application (using ALT + arrow keys).



gpsvisualizer.com

Consider using free NMEA converter in order to easily plot data using NMEA files created from logs by Logdecode. Note that NMEA files created from TRACE are naturally not containing speed data.

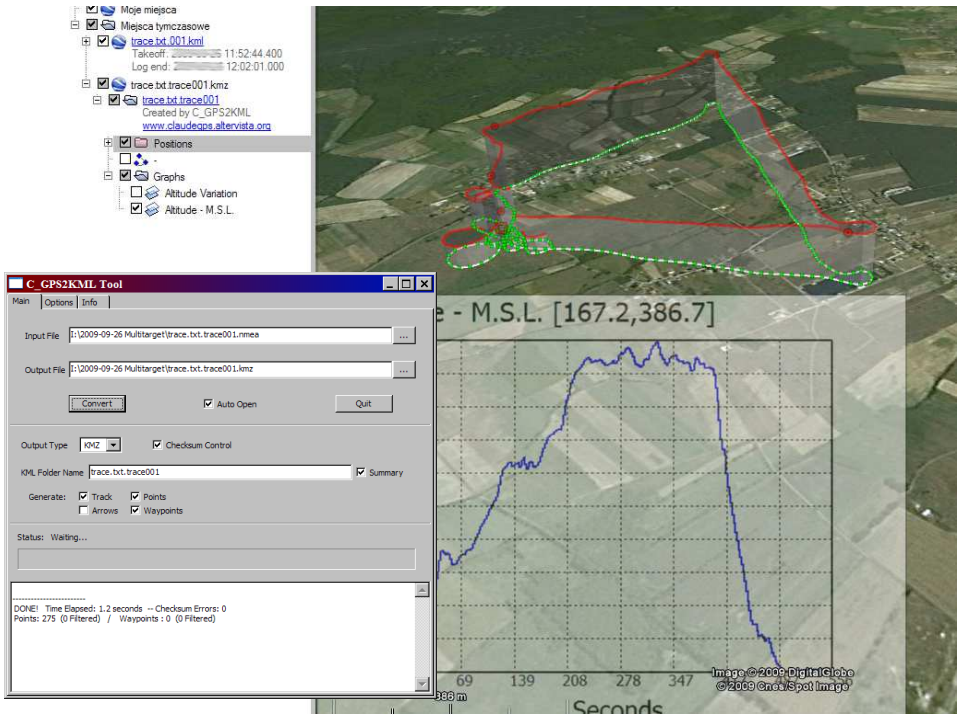




Example: altitude vs distance flown for record flight of EasyUAV

C_GPS2KML

Since GoogleEarth allows simultaneous display of several kml files, it is possible to generate additional data with any kml generator you can find. In this case, a track generated by the converter bundled with Flexipilot is overlaid by altitude plot from C_GPS2KML, which in turn used NMEA synthesized from flight logs.

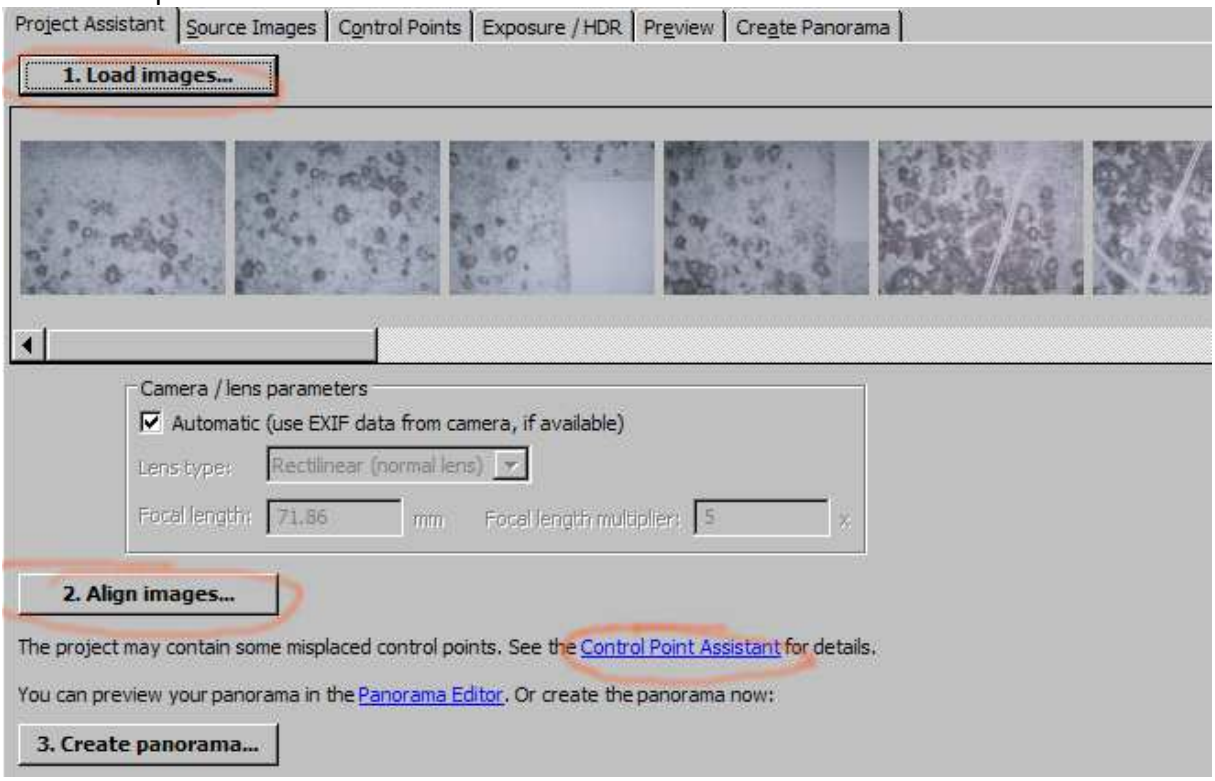


PTGUI PRO

Use commercial Panorama Stitching software in order to create maps of your area.

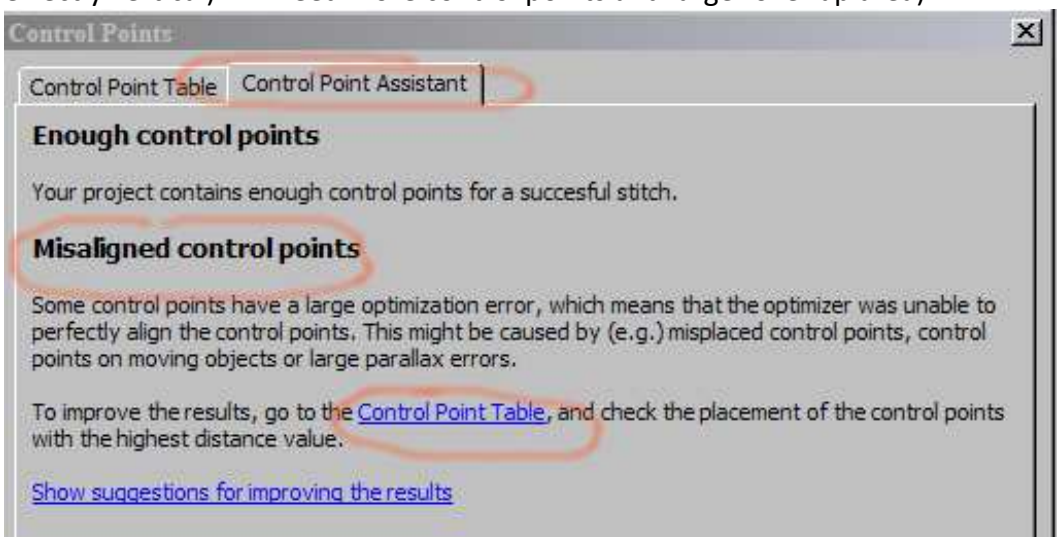
1. Resize your photos to reasonably small size, f.ex. using ImageMagick command:
mogrify -resize 50% *.jpg

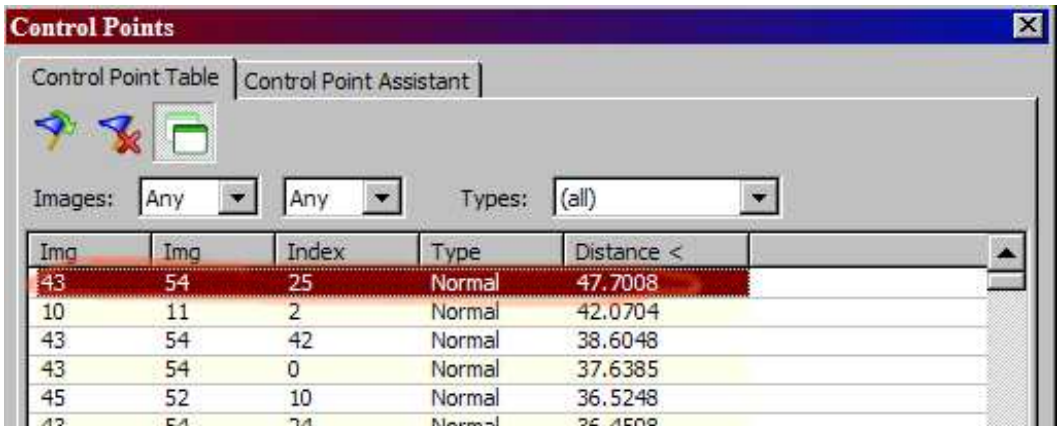
2. Load the photos into PTGUI:



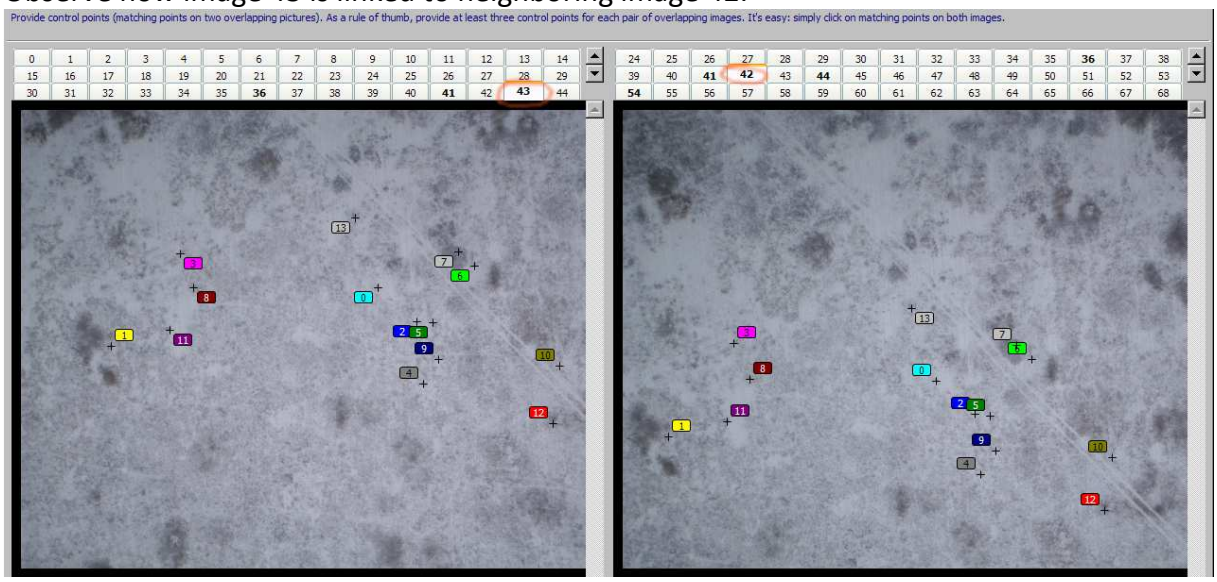
3. Click on align images, the software will try to guess the most obvious reference points between the images. The key concept is that you can create many points of reference between any of the overlapping images. PTGUI will typically detect overlaps between consecutive images (in a row), you will have to provide manually points of reference between the columns (not necessarily for all photos). Discard photos taken during significant bank or pitch (over 10deg).

4. Click on ControlPointAssistant and follow the guidelines. The photos that are distorted (not perfectly vertical) will need more control points and larger overlap area)

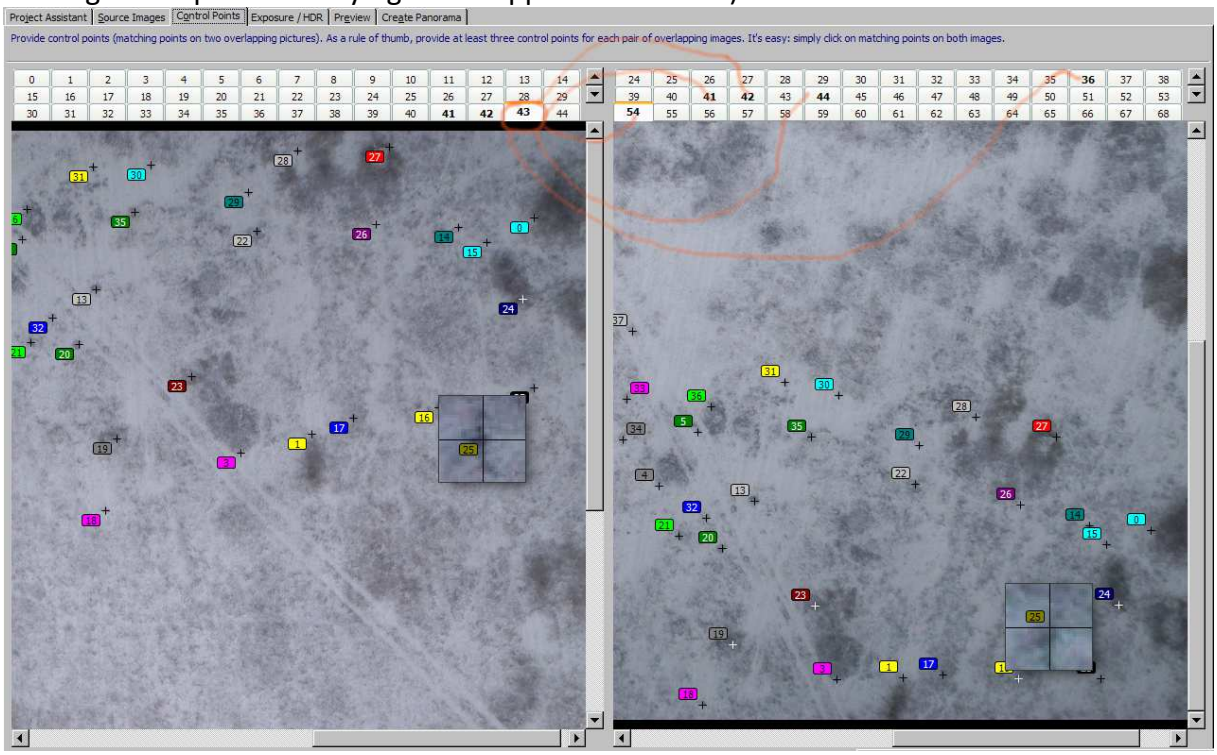




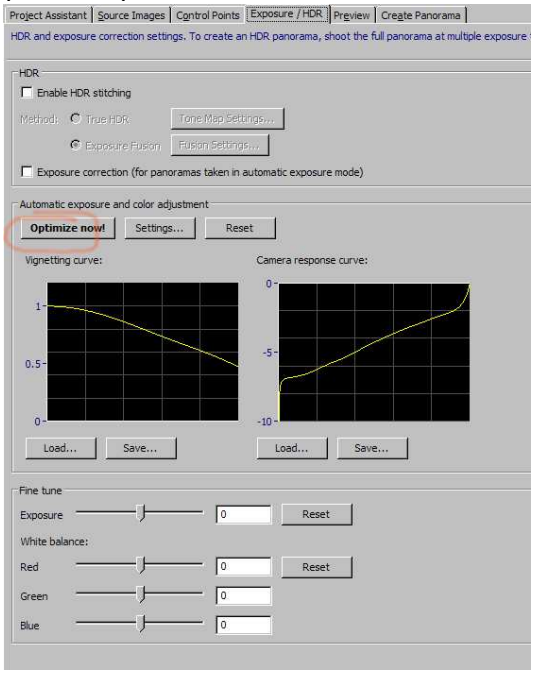
5. Add control points for specific cases:
Observe how image 43 is linked to neighboring image 42:



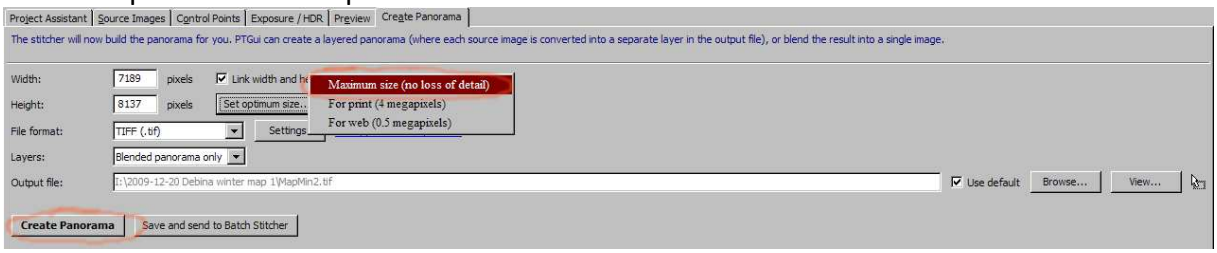
Note that image 43 is also potentially overlapping with images 41, 42 and 44 (what is not surprising), but also to the images 36 and 54 belonging to the different column (also rotated 180deg as the plane was flying in the opposite direction)



6. Use automatic image exposure optimizer



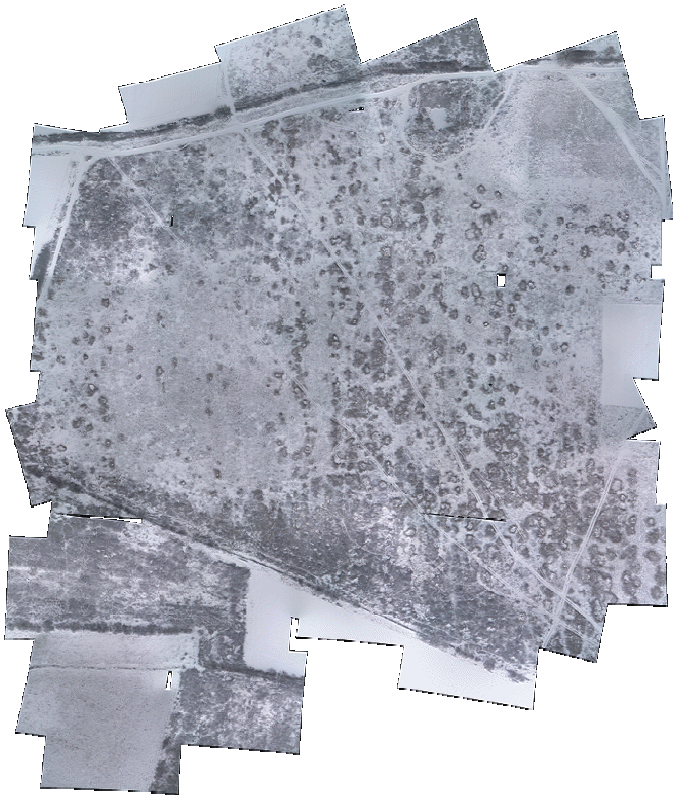
7. Create panorama bitmap.



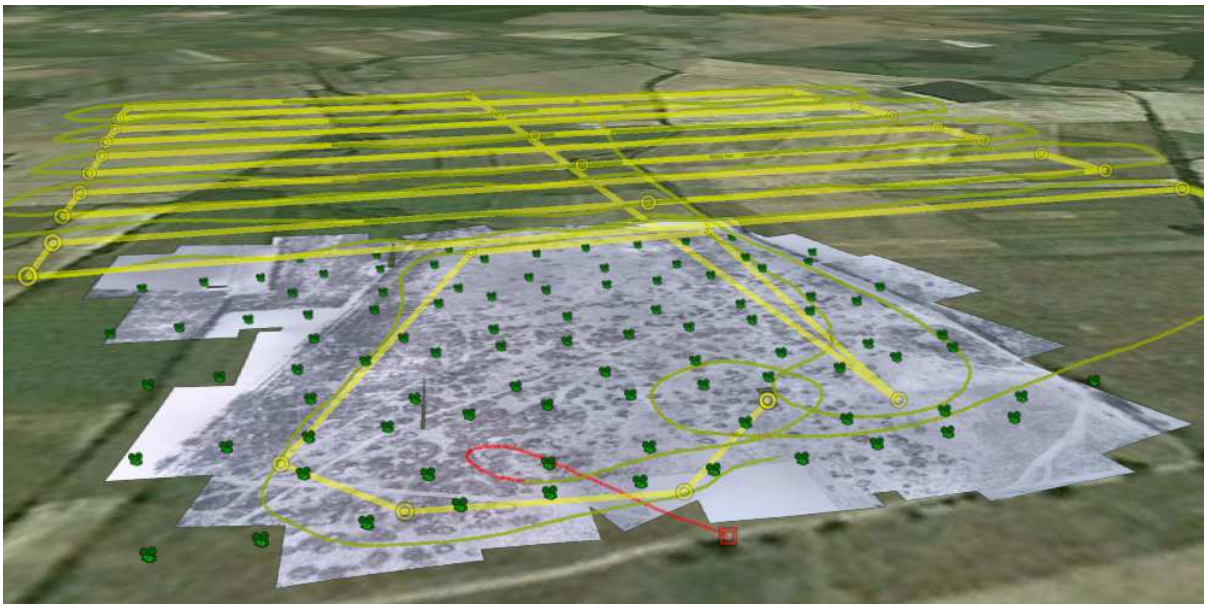
When using large file size, consider using TIFF with packbits compression which is fast to decode in viewers.

8. Import the panorama into Google Earth using 'Insert Image Overlay' then resize manually to match existing maps. Save as kmz compressed file, embedding the image in it.

Sample result: a bitmap taken with 5mpix camera shoting at 4.5s intervals from 121m altitude. Around 74 photos taken, 8 photos were discarded. A few minor regions not covered because the airspeed was too high or camera repetition rate too slow. 5mpix corresponded to 2600x1950 resolution. Image size on the ground was roughly 105x78m, therefore the pixel size from 121m altitude for this specific camera was 4cm. After halving the image size, the pixel size used for panorama stitch was 8cm.



Google Earth allows simultaneous display of imported overlay image and the kml files generated automatically from flight logs by FLEXIPILOT LogDecode software:



Web services and other tools

Free 3D model generation:
www.my3dscanner.com
(300MB zip file size limit, good quality colored point cloud only)
www.photosynth.net
(colored point cloud output only, suggested max 300 photos)
www.arc3d.be
(meshed model, textured)
www.hypr3d.com
(meshed model, max 200 photos)
www.areoscan.com
(no typo here! free preview, point cloud only, slower, partially commercial)

Geotagged photo hosting:
www.panoramio.com
www.locr.com

Mapping place names to geographic positions:
www.geonames.org

Flight path documentation:
www.gpsvisualizer.com
www.maddyhome.com/igc2kml/convert.do

Recording a flight with GPS-equipped cell phones:
www.gpslogbooks.com
www.droneos.com
www.google.com/mobile/latitude

EXIF data manipulation tools:
jhead www.sentex.net/~mwandel/jhead
exiftool www.sno.phy.queensu.ca/~phil/exiftool/
exiv2 www.exiv2.org
libexif exif ftp.iitm.ac.in/cygwin/release/exif

Georeferencing service and application:
www.georeferencer.org
mapanalyst.cartography.ch

Predicting sun position:
http://www.sunearthtools.com/dp/tools/pos_sun.php

Predicting GPS reception:
http://www.sokkia.com/Products/Detail/Planning_Software.aspx

Compatible professional map generation services

Pix4D
DroneMapper
ICAROS
GISCAT
Photoscan
EnsoMosaic