Flyception user manual

Dhruv Grover, Takeo Katsuki, and Ralph J. Greenspan

Version 1.0
2016-4-13

Table of contents

Installation

Prerequisites

PC running Microsoft Windows 7/8 64-bit
Microsoft Visual Studio 2013 Professional Edition
CMake
OpenCV library
Point Grey Flycapture SDK 64-bit
Teledyne DALSA Sapera LT SDK
NI-DAQmx
Arduino
MicroManager

Hardware configuration on test setup

Dell Precision T7610 Workstation (Six-core 2.6GHz)
16GB RAM
2x256GB SSD drives, one for Windows OS and second for video recording
Teledyne DALSA XCelera PX4 full frame-grabber PCIE card

National Instruments PCIE-6351 DAQ
Point Grey 2-port USB3 PCIE card
Firewire 800 card
Arduino Mega 2560

Getting Started

Download and install Microsoft Visual Studio 2013. For academic use, free licensed copies of the software are available on dreamspark.com

Download and install CMake v3.3 or above (https://cmake.org/).

Download OpenCV source code (v3.0 or above required) from GitHub (https://github.com/Itseez/opencv). OpenCV for windows provided on opencv.org can also be downloaded as it includes the source code along with a default build, however, the flyception software requires a custom build of OpenCV.

Download and install the latest version of Point Grey Flycapture SDK 64-bit (v2.8 or above) from (https://www.ptgrey.com/support/downloads).

Download and install the latest version of National Instruments DAQmx software from (http://sine.ni.com/nips/cds/view/p/lang/en/nid/10181).

Download and install the Sapera LT SDK from Teledyne DALSA (https://www.teledynedalsa.com/imaging/products/software/sapera/lt/). An account creation step might be required for software download. This software is included with the framegrabber card.

Download and install the latest Arduino IDE environment from (https://www.arduino.cc/en/Main/Software).

Download and install the latest version of MicroManager (https://www.micro-manager.org/).

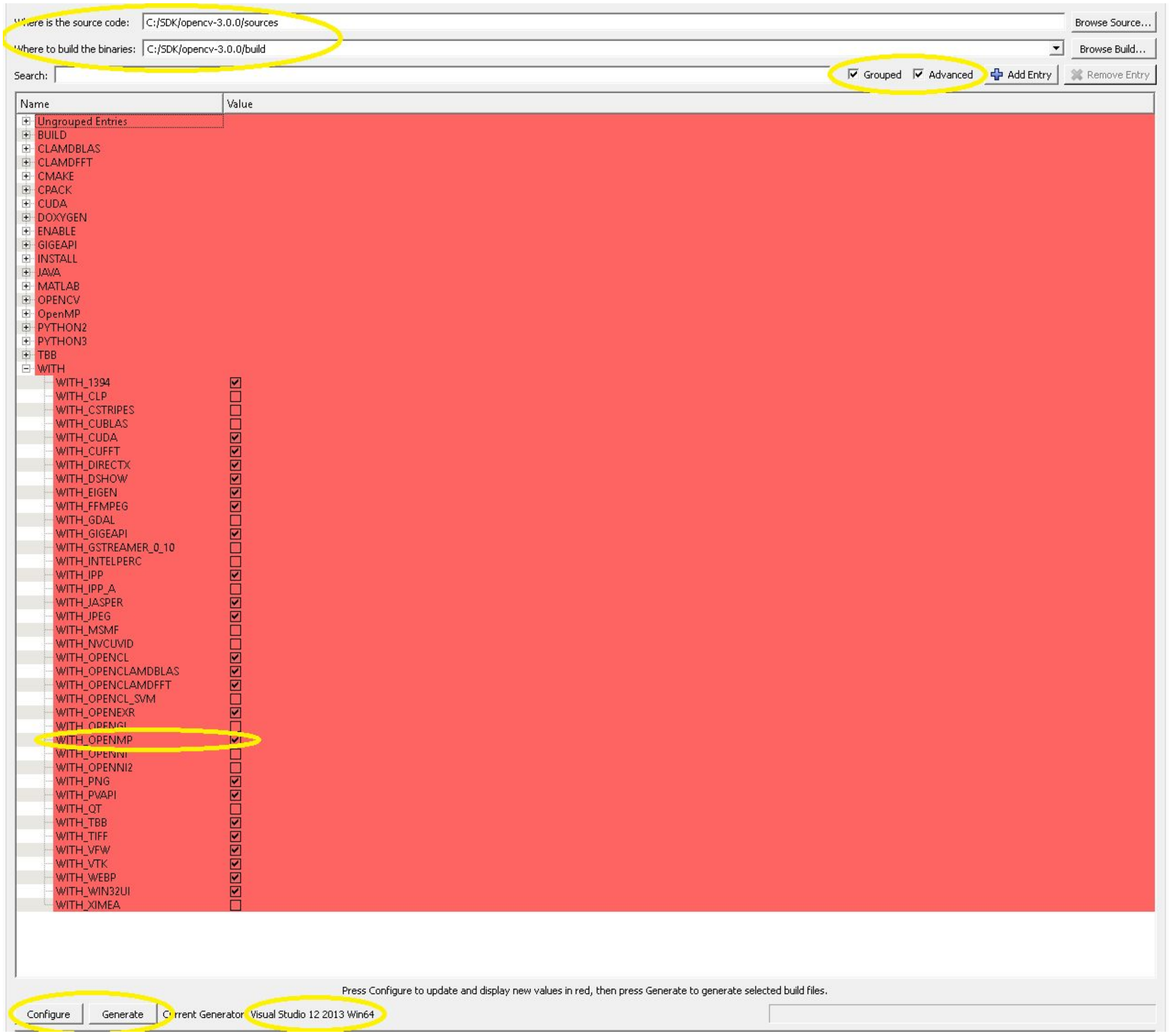
Compiling

Custom building OpenCV with CMake

The following instructions are similar to those provided by OpenCV at
(http://docs.opencv.org/3.0-beta/doc/tutorials/introduction/windows_install/windows_install.html)

Unzip the OpenCV source code to C:\SDK\opencv-3.0.0\sources\ folder
Create a folder C:\SDK\opencv-3.0.0\build\ to hold the custom built binaries
Run the cmake-gui program
Enter the full paths for the OpenCV source code and binaries
Select the grouped and advanced options and press configure
Select Visual Studio 12.0 (2013) Win64 as the compiler of choice
Uncheck BUILD->DOCS
Check WITH->WITH_OPENMP
Press configure again and verify that OpenMP module has been loaded
Press configure again to include OpenMP and then press generate

The source code for the OpenCV library with OpenMP support for Visual Studio 2013 will now be saved to the binaries folder indicated in CMake.



Where is the source code: C:/SDK/opencv-3.0.0/sources

Where to build the binaries: C:/SDK/opencv-3.0.0/build

Search:

☑ Grouped ☑ Advanced

Name	Value
Ungrouped Entries	
BUILD	
CLAMDBLAS	
CLAMDFFT	
CMAKE	
CPACK	
CUDA	
DOXYGEN	
ENABLE	
GIGEAPI	
INSTALL	
JAVA	
MATLAB	
OPENCV	
OpenMP	
PYTHON2	
PYTHON3	
TBB	
WITH	
WITH_1394	☑
WITH_CLP	☐
WITH_CSTRIPES	☐
WITH_CUBLAS	☐
WITH_CUDA	☑
WITH_CUFFT	☑
WITH_DIRECTX	☑
WITH_DSHOW	☑
WITH_EIGEN	☑
WITH_FFMPEG	☑
WITH_GDAL	☐
WITH_GIGEAPI	☑
WITH_GSTREAMER_0_10	☐
WITH_INTELPERC	☐
WITH_IPP	☑
WITH_IPP_A	☐
WITH_JASPER	☑
WITH_JPEG	☑
WITH_MSMF	☐
WITH_NVCUVID	☐
WITH_OPENCL	☑
WITH_OPENCLAMDBLAS	☑
WITH_OPENCLAMDFFT	☑
WITH_OPENCL_SVM	☐
WITH_OPENEXR	☑
WITH_OPENGL	☐
WITH_OPENMP	☑
WITH_OPENNI	☐
WITH_OPENNI2	☐
WITH_PNG	☑
WITH_PVAPI	☑
WITH_QT	☐
WITH_TBB	☑
WITH_TIFF	☑
WITH_VFW	☑
WITH_VTK	☑
WITH_WEBP	☑
WITH_WIN32UI	☑
WITH_XIMEA	☐

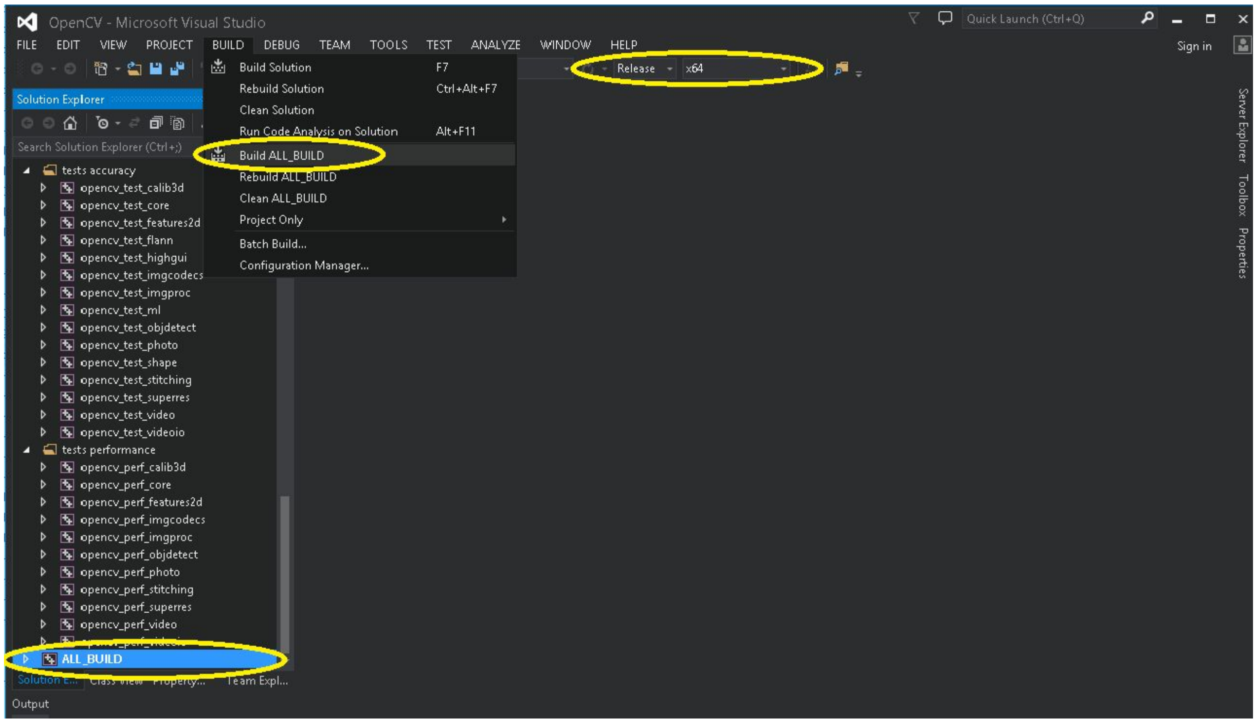
Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 12 2013 Win64

Compiling custom OpenCV source in Visual Studio

Navigate to C:\SDK\opencv-3.0.0\build and run the OpenCV.sln visual studio solution file

Once loaded, select the ALL_BUILD project and use the RELEASE solution configuration, and x64 solution platform. Select the BUILD menu item and then build ALL_BUILD



OpenCV should now begin compiling. This process may take a few minutes.

Once compiled, select the INSTALL project and then the BUILD menu item followed by build INSTALL. This installs the OpenCV binaries built to the folder C:\SDK\opencv-3.0.0\build\install\

Custom binaries of OpenCV with OpenMP support are now built and are saved to the following folders. In the next section, we will be including these folder paths when compiling the flyception source code

INCLUDE PATH - C:\SDK\opencv-3.0.0\build\install\include
LIB PATH - C:\SDK\opencv-3.0.0\build\install\x64\vc12\lib
BIN PATH - C:\SDK\opencv-3.0.0\build\install\x64\vc12\bin

Note: Path to the OpenCV bin folder needs to be included in the SYSTEM Path. This can be achieved by navigating to Control Panel -> System -> Advanced System Settings -> Environment Variables -> System Variables -> Path. Add the aforementioned BIN PATH to the path string. A restart of Windows may be required for the system path to be refreshed.

Setting up Flyception

Download the latest version of the flyception source code from
https://github.com/dgrover/flyception.

The flyception source code has the following folder hierarchy:

- Flyception - root folder
- Flyception\Arduino - includes code to be installed on the arduino for camera triggering and odor delivery
- Flyception\flybam - source code for the tracking program
- Flyception\calibration - scripts to calibrate the arena-view camera to convert pixel coordinates to real-world coordinates
- Flyception\ccf - camera link configuration file for Point Grey Gazelle camera (fly-view)

Compile and upload the arduino sketch multi_cam_trigger_mega to the first arduino. This generates PWM signals to trigger the arena-view, fly-view, fluo-view cameras and the flash using synchronized arduino timers.

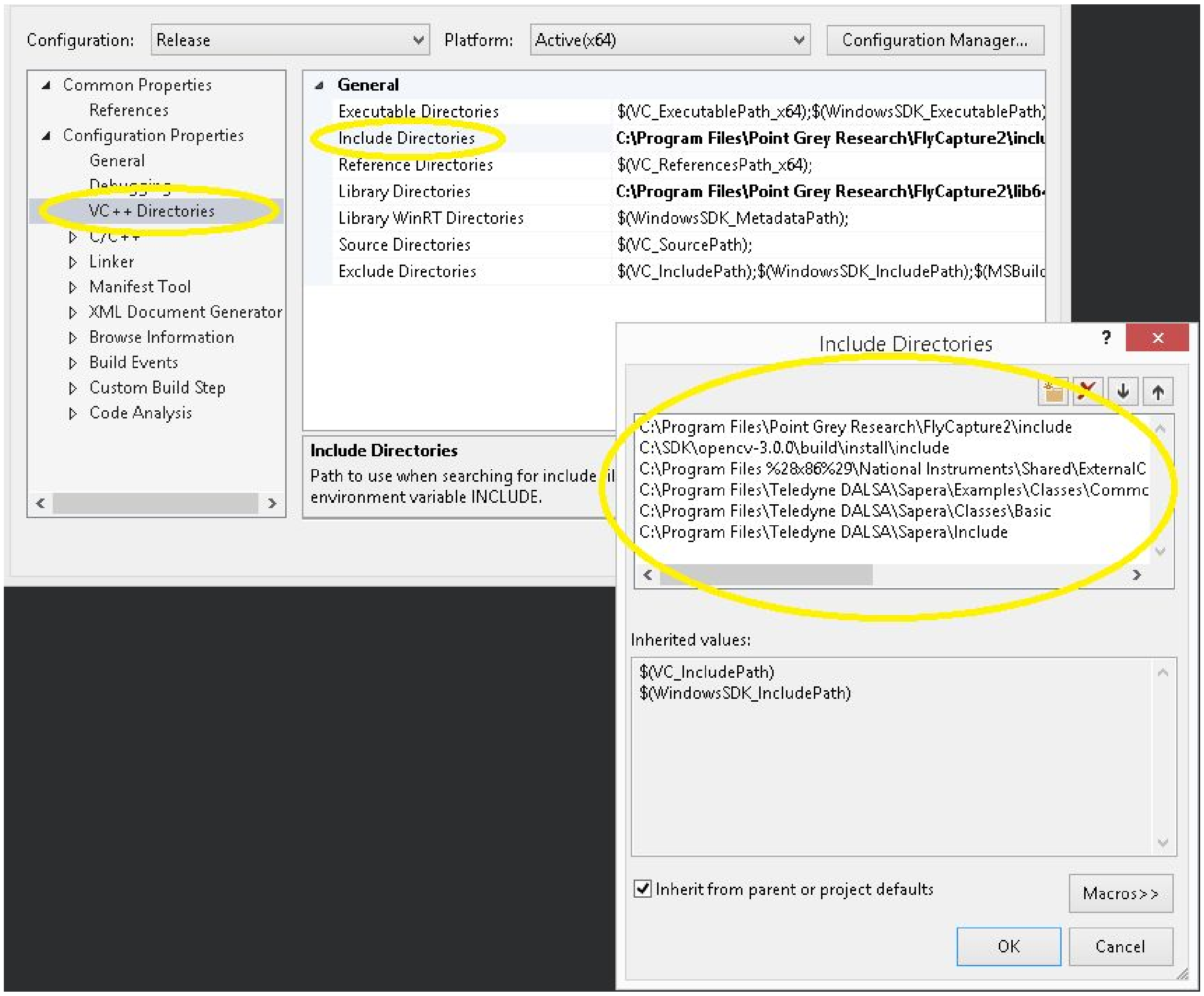
Compiling Flyception

Navigate to the Flyception\flybam folder and run the flybam.sln solution file.
Verify that the x64\RELEASE solution platform is selected.

The provided project file includes the path to all prerequisite libraries, however, if custom folders were used, the library paths need to be changed here.

Navigate to Project -> flybam properties -> configuration properties -> vc++ directories
Verify the following directories are included

Include directories
C:\Program Files\Point Grey Research\FlyCapture2\include
C:\SDK\opencv-3.0.0\build\install\include
C:\Program Files x86\National Instruments\Shared\ExternalCompilerSupport\C\include
C:\Program Files\Teledyne DALSA\Sapera\Examples\Classes\Common
C:\Program Files\Teledyne DALSA\Sapera\Classes\Basic
C:\Program Files\Teledyne DALSA\Sapera\Include



Executable Directories	$(VC_ExecutablePath_x64);$(WindowsSDK_ExecutablePath)
Include Directories	**C:\Program Files\Point Grey Research\FlyCapture2\inclu**
Reference Directories	$(VC_ReferencesPath_x64);
Library Directories	**C:\Program Files\Point Grey Research\FlyCapture2\lib6**
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(VC_IncludePath);$(WindowsSDK_IncludePath);$(MSBuild

Include Directories
Path to use when searching for include fil
environment variable INCLUDE.

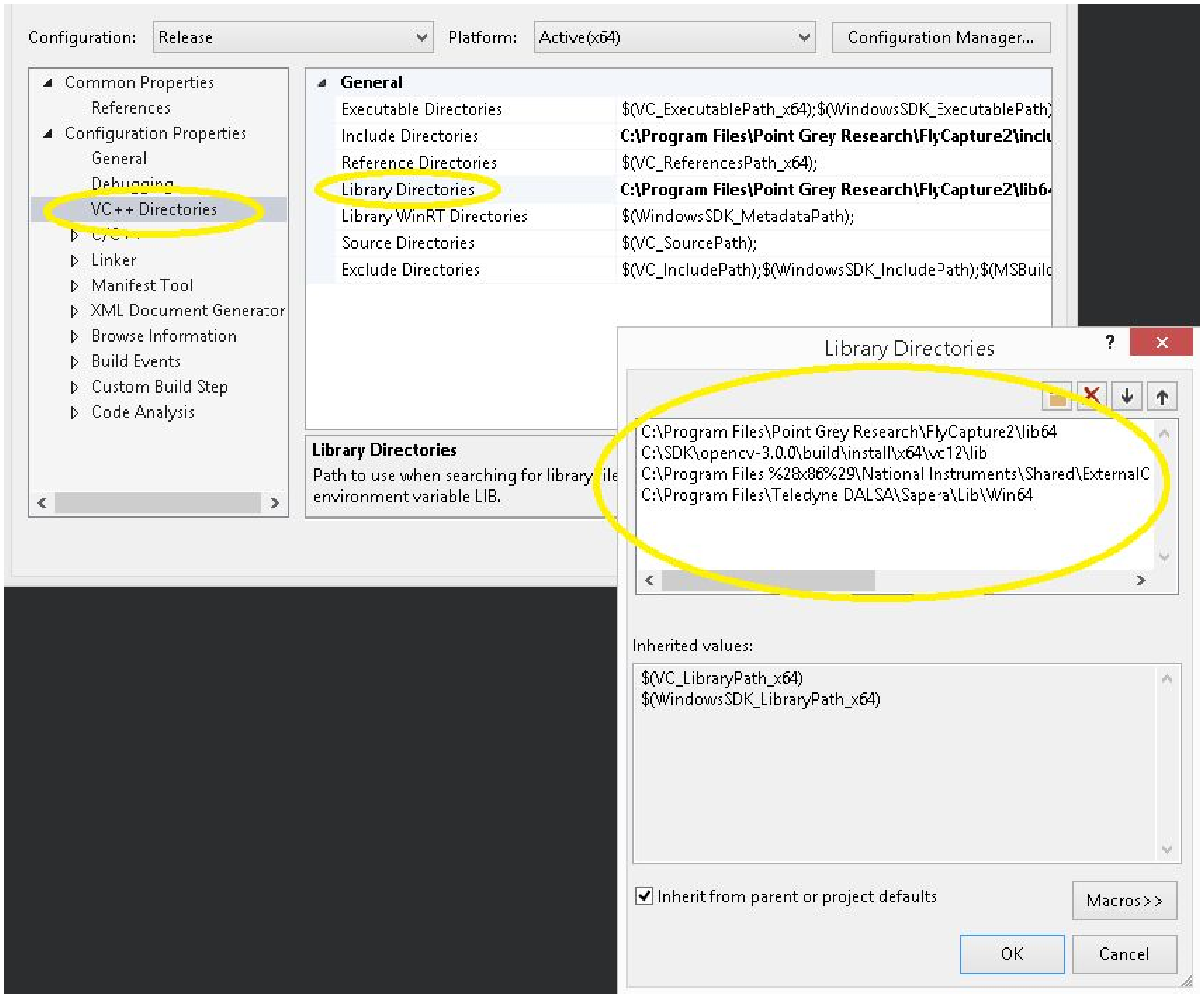
Include Directories

```
C:\Program Files\Point Grey Research\FlyCapture2\include
C:\SDK\opencv-3.0.0\build\install\include
C:\Program Files %28x86%29\National Instruments\Shared\ExternalC
C:\Program Files\Teledyne DALSA\Sapera\Examples\Classes\Commc
C:\Program Files\Teledyne DALSA\Sapera\Classes\Basic
C:\Program Files\Teledyne DALSA\Sapera\Include
```

Inherited values:

```
$(VC_IncludePath)
$(WindowsSDK_IncludePath)
```

Library directories
C:\Program Files\Point Grey Research\FlyCapture2\lib64
C:\SDK\opencv-3.0.0\build\install\x64\vc12\lib
C:\Program Files %28x86%29\National Instruments\Shared\ExternalCompilerSupport\C\lib64\msvc
C:\Program Files\Teledyne DALSA\Sapera\Lib\Win64
Under Linker -> Input, verify the following .lib files are included



Executable Directories: $(VC_ExecutablePath_x64);$(WindowsSDK_ExecutablePath)
Include Directories: C:\Program Files\Point Grey Research\FlyCapture2\inclu
Reference Directories: $(VC_ReferencesPath_x64);
Library Directories: C:\Program Files\Point Grey Research\FlyCapture2\lib6
Library WinRT Directories: $(WindowsSDK_MetadataPath);
Source Directories: $(VC_SourcePath);
Exclude Directories: $(VC_IncludePath);$(WindowsSDK_IncludePath);$(MSBuild

Library Directories
Path to use when searching for library file environment variable LIB.

Library Directories

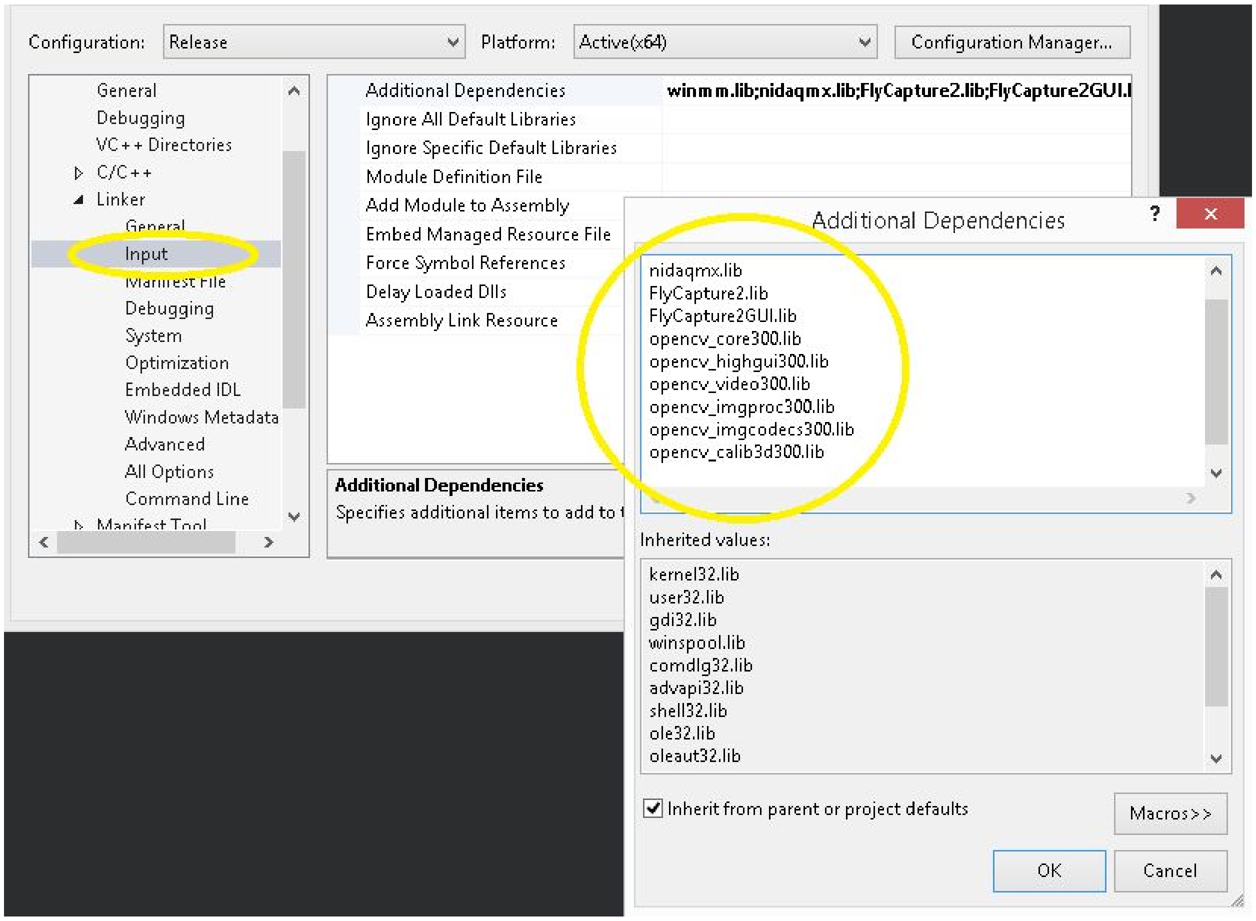
```
C:\Program Files\Point Grey Research\FlyCapture2\lib64
C:\SDK\opencv-3.0.0\build\install\x64\vc12\lib
C:\Program Files %28x86%29\National Instruments\Shared\ExternalC
C:\Program Files\Teledyne DALSA\Sapera\Lib\Win64
```

Inherited values:

```
$(VC_LibraryPath_x64)
$(WindowsSDK_LibraryPath_x64)
```

[✓] Inherit from parent or project defaults

nidaqmx.lib
FlyCapture2.lib
FlyCapture2GUI.lib
opencv_core300.lib
opencv_highgui300.lib
opencv_video300.lib
opencv_imgproc300.lib
opencv_imgcodecs300.lib
opencv_calib3d300.lib
SapClassBasic.lib


Configuration: Release Platform: Active(x64) Configuration Manager...

General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool

Additional Dependencies	winmm.lib;nidaqmx.lib;FlyCapture2.lib;FlyCapture2GUI.l
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to t

Additional Dependencies

```
nidaqmx.lib
FlyCapture2.lib
FlyCapture2GUI.lib
opencv_core300.lib
opencv_highgui300.lib
opencv_video300.lib
opencv_imgproc300.lib
opencv_imgcodecs300.lib
opencv_calib3d300.lib
```

Inherited values:

```
kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib
advapi32.lib
shell32.lib
ole32.lib
oleaut32.lib
```

☑ Inherit from parent or project defaults

With these folder paths included, the flyception software is ready to be compiled and executed.

Arena-view Camera Calibration

We follow a similar procedure to the OpenCV tutorial on camera calibration. In short, the distortion and camera matrices are calculated from arena-view input images of a checkerboard target at different orientations (roughly 8-10 images are sufficient for accurate calibration, see calibration\arena\images folder for samples). The calibration\arena\images subfolder includes images taken during a successful calibration.

See the following link for a tutorial with detailed description on the theory and use of the camera calibration routines.
(http://docs.opencv.org/3.0-beta/doc/tutorials/calib3d/camera_calibration/camera_calibration.html)

The calibration\calib folder includes code which reads in the images stored in calibration\arena\images folder and outputs the camera and distortion matrices to the out_camera_data.xml file. Next, the camera projection matrix is calculated using the distortion and camera matrices and an image of the checkerboard centered in line with the center of the fly arena. This is to ensure that fly coordinates outputted by the flyception program are in mm's with the origin being the center of the arena. See code in calibration\project folder. The code reads in the out_camera_data.xml file and outputs the camera_projection_data.xml file which is read into the main flyception program.

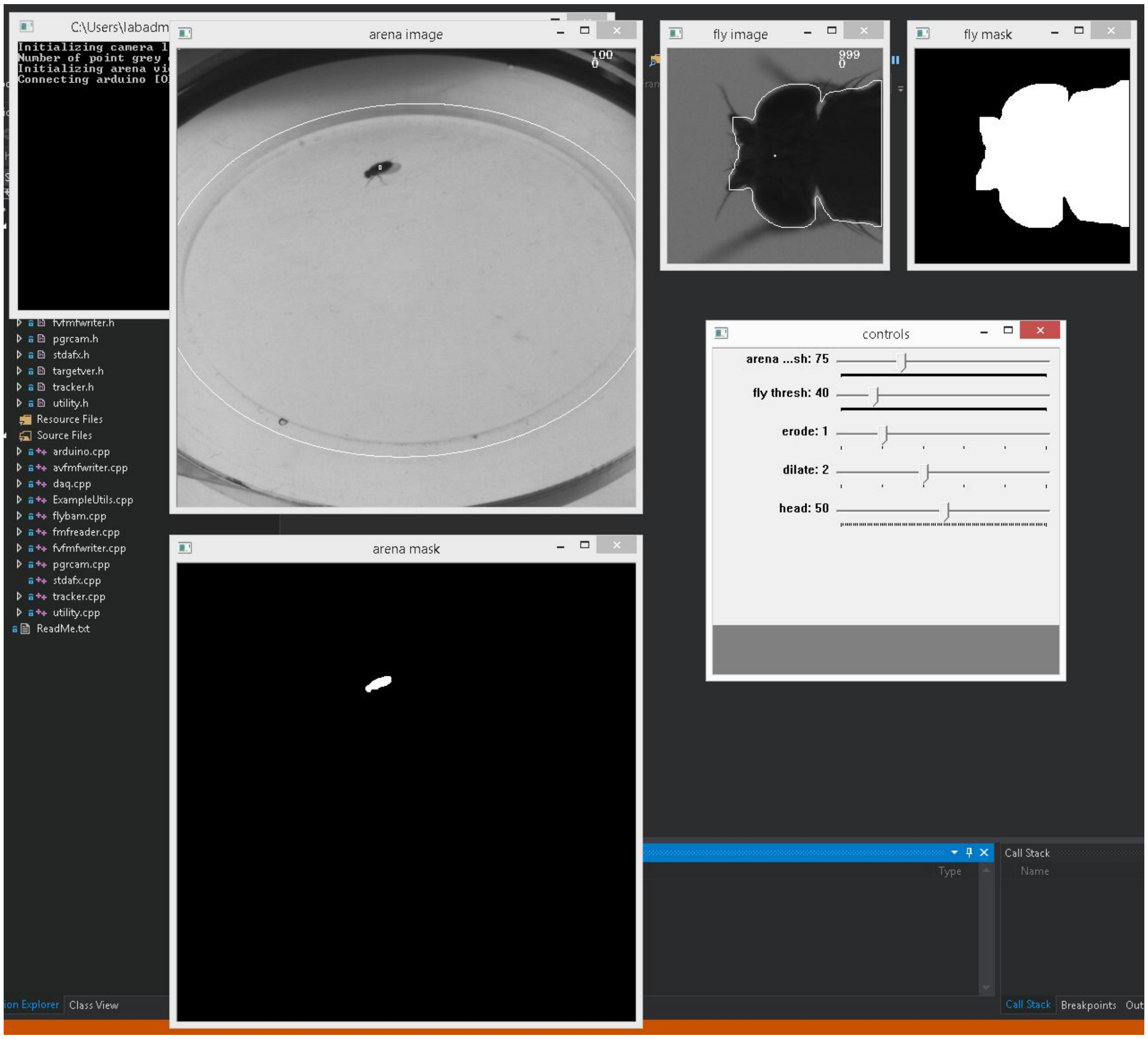
Using Flyception

When the flyception code is executed, the arena-view and fly-cameras are initialized (see screenshot below). The software automatically starts tracking flies in the arena-view video and turns the galvanometer mirrors to follow the first fly tracked. The following keyboard shortcuts used for controlling the software -

F1 - initializes fly-view tracking
F2 - starts/stops recording of arena-view and fly-view images and tracking coordinates. If the fluo-view camera is initialized via MicroManager (see next section), recording of fluo-vidoes will also begin.
F3 - fires the flash, with an additional automatic flash fired 60s later.
F4 - opens pneumatic valve for odor delivery experiments.
UP, DOWN, LEFT, RIGHT keys - manual control of galvanometer mirror assembly
HOME - switches back from manual galvanometer control mode to automatic arena-view tracking mode
ESC - exits the program

In addition to the keyboard shortcuts, flyception also allows the user to tweak various tracking parameters via a control panel. For instance, thresholds for arena-view and fly-view tracking, morphological operations and head offset position from the top of the fly head in fly-view can be user controlled.

Other parameters such as number of flies, length of videos, arena dimensions, galvo settings can be changed in the stdafx.h file in the flybam directory. If the system is replicated exactly as outlined in this article, most of these settings need not be altered.



```
C:\Users\labadm
Initializing camera l
Number of point grey
Initializing arena vi
Connecting arduino [O
```

arena ...sh: 75
fly thresh: 40
erode: 1
dilate: 2
head: 50

The output of the system after a successful recording session is as follows -

- Arena view video (default length 100s) in .fmf format
- Fly-view video (default length 100s) in .fmf format
- Arena view and fly video log files to ensure cameras are functioning at preset frame rates
- Trajectory file, space delimited, with the following parameters per frame at 1000Hz [*Frame Number, Arena x-coordinate, Arena y-coordinate, Fly-view head center x-coordinate, Fly-view head center y-coordinate, Fly-view body edge center x-coordinate, Fly-view body edge center y-coordinate, Galvo x-angle, Galvo y-angle, Odor delivered (true/false)*]

Fluorescence video recording with MicroManager

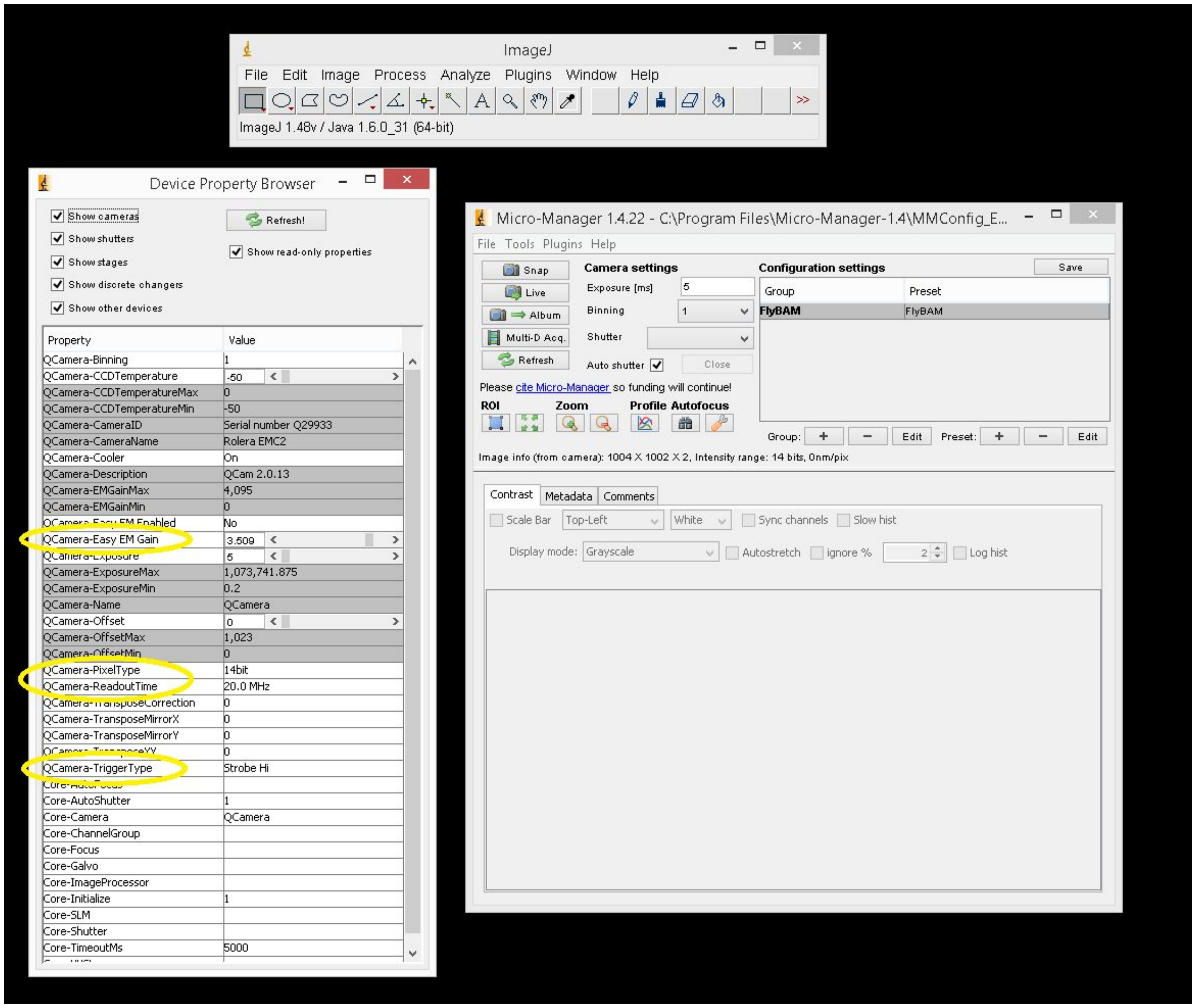
Fluorescence images are captured with a Rolera EM-C2 EM-CCD camera using MicroManager. The camera settings are controlled in the Device Property Browser shown in the screen capture below with the following parameters;

- Trigger Type: Strobe Hi
- EM Gain: 3,500
- Pixel Type: 14 bit
- Readout Time: 20.0 MHz
- Binning: 1 or 2

F2 key needs to be bound to multi-D acquisition function so that the image acquisition starts roughly at the same time as fly-view and arena-view. Number of images to capture is set to 10,000 in Multi-D Acquisition setting. Image size is reduced to 125x125 pixels (62x62 in the case of binning 2) by pressing the ROI button.

Note that users must launch MicroManager before the flyception tracking software, and make sure to assign only one processing core instead of all cores to the java process in order to avoid resource conflicts between the tracking software and MicroManager. This can be done through Task Manager of the Windows operating system.

Start the camera by hitting Live button in the main console, and then start recording with F2 shortcut key. Images will be saved in TIFF format and metadata will be saved in TXT format, with a user-specified filename prefix.



ImageJ 1.48v / Java 1.6.0_31 (64-bit)

- [✓] Show cameras
- [✓] Show shutters
- [✓] Show stages
- [✓] Show discrete changers
- [✓] Show other devices
- [✓] Show read-only properties

Property	Value
QCamera-Binning	1
QCamera-CCDTemperature	-50
QCamera-CCDTemperatureMax	0
QCamera-CCDTemperatureMin	-50
QCamera-CameraID	Serial number Q29933
QCamera-CameraName	Rolera EMC2
QCamera-Cooler	On
QCamera-Description	QCam 2.0.13
QCamera-EMGainMax	4,095
QCamera-EMGainMin	0
QCamera-Easy EM Enabled	No
QCamera-Easy EM Gain	3.509
QCamera-Exposure	5
QCamera-ExposureMax	1,073,741.875
QCamera-ExposureMin	0.2
QCamera-Name	QCamera
QCamera-Offset	0
QCamera-OffsetMax	1,023
QCamera-OffsetMin	0
QCamera-PixelType	14bit
QCamera-ReadoutTime	20.0 MHz
QCamera-TransposeCorrection	0
QCamera-TransposeMirrorX	0
QCamera-TransposeMirrorY	0
QCamera-TransposeXY	0
QCamera-TriggerType	Strobe Hi
Core-AutoFocus	
Core-AutoShutter	1
Core-Camera	QCamera
Core-ChannelGroup	
Core-Focus	
Core-Galvo	
Core-ImageProcessor	
Core-Initialize	1
Core-SLM	
Core-Shutter	
Core-TimeoutMs	5000

Camera settings
Exposure [ms]: 5
Binning: 1
Shutter:
Auto shutter [✓]

Configuration settings

Group	Preset
FlyBAM	FlyBAM

Please cite Micro-Manager so funding will continue!

Image info (from camera): 1004 X 1002 X 2, Intensity range: 14 bits, 0nm/pix

Scale Bar: Top-Left; White; Sync channels; Slow hist
Display mode: Grayscale; Autostretch; ignore %: 2; Log hist

Post-acquisition Image analysis on R

Quantification of fluorescence changes is done using custom scripts written in R. The scripts load a fly-view video and estimate a set of transformation functions that achieves near-stationary fly image via image registration. The same set of transformation is applied to fluo-view image so that the fluorescence intensity can be measured with a sub-brain resolution. All the image processing and analysis steps are included in one R function Flyception(), and this function outputs a total of about 60 files depending on the options chosen. Even though the analysis is highly automated, since R is an interpreted language, every step of analysis is customizable line-by-line if desired.

The two fmf files, four log files, one tiff file, and one txt file generated from flyception and MicroManager need to be placed in one directory.

In addition to the scripts included in this supplementary material, following R packages are required to analyze flyception data. Follow the instruction of each package for installation.

- RImageBook
- Rcpp
- zoo
- RNiftyReg
- plotrix
- ggplot2
- plyr
- reshape2
- rlogging

Usage:
Flyception(rdir, dir, prefix, autopos=T, video_out=F, mating=F, stimlen=200, stimplotlen=800, reuse=T, fmf2tif=T, fpsfl=100)

Arguments:

rdir	a character vector of directory name that contains R scripts
dir	a character vector of directory name that contains video data flyception generated
prefix	a character vector used for output file names
autopos	logical: should image alignment automatically performed using FNCC?
video_out	logical: should .mp4 videos generated?
mating	logical: analyze mating behavior?
stimlen	integer indicating the length of stimulus in a number of frames
stimplotlen	integer indicating the length of a dF/F plot and dF/F image representation
reuse	logical: should RDS files be used from the previous process?
fmf2tif	logical: should fmf files be converted to tif?
fpsfl	integer indicating the frame rate of fluo-view camera

Details:
This function provides an automated pipeline for performing the following image processing steps and saves resulted images and plots in the folder specified in the dir argument. log.txt contains The main processes are alignment of fly-view and fluo-view videos; segmentation of the optical window on the fly head; image registration of fly-view, window mask, and fluo-view videos; calculation of fluorescence intensity changes with dF/F plots and dF/F heat maps; generation of fly trajectories; and generation of a number of matric such as speed of the fly, tracking error, image focus, and window size, which can be used for filtering out frames that don't fit user-defined criteria. The function also provides options to convert FMF files into TIFF files and to generate mp4 videos for presenting results. Users can also opt to reuse RDS files generated from the first run to skip time-consuming steps such as image registration in subsequent analyses with different parameters.

Examples:

```
rdir <- "C:/path/to/the/directory/containing/scripts"
source(paste0(rdir, "Flyception.R"))
dir <- "C:/path/to/the/directory/containing/sample_data"
prefix <- "OK107-Gal4_UAS-myr-EGFP_2"
Flyception(rdir=rdir, dir=dir, prefix=prefix, autopos=T, video_out=F,
mating=F, stimlen=20, stimplotlen=80, reuse=T, fmf2tif=T, fpsfl=100)
```