Introduction

This Read Me file contains a brief overview of software used to complete this project, how to build and compile, and some example outputs.

Project Objective

The goal of this project was to solve a common stereoscopic vision problem: the calculation of image disparity and comparison of results to competing methods. In this study only sparse disparity maps are considered, where the disparity space is sampled at key points detected via two methods. First, the canonical SIFT algorithm is used to both detect key points and describe key point descriptors and treated as the reference system. Literature for this algorithm can be found in [1], and a digital copy is included in the /papers directory in this software package. \*

For comparison, the FAST feature detector is used to generate key points in the system under test. The FAST feature detector described in [2]\* is a resource and speed optimized detector that is intended for embedded systems or high-speed applications. To generate the descriptors for each key point, the FREAK algorithm described in [3]\* is chosen. This descriptor is very different from SIFT in how it describes the intensity of a descriptor. The descriptor still reports a similar orientation that uses gradients though, and OpenCV still provides a BruteForceMatcher flavor that is designed for processing this type of descriptor.

All of these algorithms are available is ordinary OpenCV distributions or the opencvcontrib/xfeatures2d module, the later for which requires non-free features to be enabled. Building this into a custom OpenCV library will be described in this document.

*\*copyright protected and distribution-controlled papers are included only for convenience under the assumption that ASU students and staff have access to the same documents. For this purpose, this software should not be distributed publicly, and only to persons whom have legal access to these documents.*

Contents of This Software Package

Only the files and folders of interest are shown below in the tree structure, please not that many more files are actually required to successfully run, and they are included but left out of the following graph to be consice.

# TREE:.

# | OpenCVSolution.sln

# | README.docx

# | Report.docx

# |

# +---FastFreak\_Stereo

# | errors.txt

# | FastFreak\_Stereo.vcxproj

# | Main.cpp

# |

# +---inputs

# | cones2.png

# | cones6.png

# | cones\_disp2.png

# |

# +---Matlab

# | results\_analysis.m

# |

# +---opencv

# | |

# | +---include

# | |

# | \---lib

# | opencv\_world343.lib

# | opencv\_world343d.lib

# |

# +---outputs

# | FastFreakMatches.csv

# | SiftMatches.csv

# |

# \---papers

# Root Directory

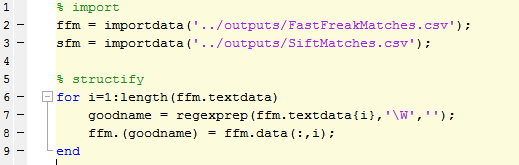
To open the solution, simply use the OpenCVSolution.sln file to launch Visual Studio. The project containing the detection software is in /FastFreak\_Stereo, where the Visual Studio project lives. The folder /papers contains reference documents, and the /opencv folder has the custom OpenCV build outputs, compiled for Windows 64-bit using Visual Studio v14 toolchain (ships with VS15 by default). For the rest of this document, paths will be relative to this folder, or this folder will be referenced as “<root>”, for example <root>/opencv/include referes to the /opencv/include directory provided **as part of this software package.**

# Input Files

The input files are included in the /inputs folder for ease of use. There may be more pictures here than required, the “cones” data will be the only set analyzed.

# Output Files

The raw match information has been included in CSV format for each method. This format is easy to read, for example, in my analysis I take in the data with Matlab using a built in import function:



Also included is a sample text output file taken directly from the console output during execution.

Using the Software Package

# Target System Information

The flavor of OpenCV used for this project was built from sources using the 3.4 branch of opencv and opencv\_contrib hosted on GitHub:

# <https://github.com/opencv/opencv/tree/3.4>

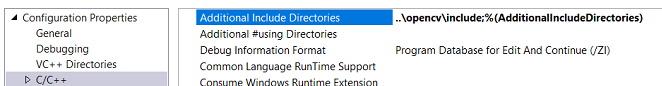
# <https://github.com/opencv/opencv_contrib/tree/3.4>

These modules were build using Visual Studio 2015, using the v14 toolset, on a Windows 64-bit machine, Intel processor, and Cuda was *not* enabled for this build. Python was also excluded specifically to reduce support requirements. CMake 3.11.2, downloaded and installed August 2018, was used for project configuration.

The OpenCV Include, Bin, and Lib directories have all been included and should work stand-alone assuming the project is built and run on a Windows 64-bit computer.

It is also assumed that the user has access to MATLAB, however my results have been included in the normal output of the C++ project. Calculating RMS and Percent Bad Pixels is only supported in MATLAB at time of writing due to time constraints.

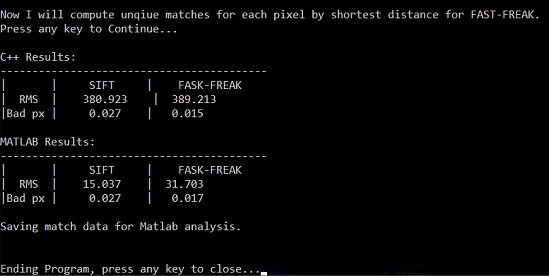
# Instructions for **Building** the project included in this ZIP folder

1. It is assumed the user is familiar with the PDF document “[OpenCV\_Instructions\_Windows\_Fall17.pdf](https://myasucourses.asu.edu/bbcswebdav/pid-19118610-dt-content-rid-135205475_1/xid-135205475_1)”.
2. It is also assumed that the user has compiled from sources, their own OpenCV 3.4 Libraries including at least the “xfeatures2d” module from opencv\_contrib 3.4, or obtained them from this software package for use only on Windows 64-bit systems.
3. To build and compile, start by opening the provided solution “OpenCVSolution.sln”.
4. The project used for this assignment is called “FastFreak\_Stereo”. To get this project ready for build, right click on it and open the properties menu.
5. For convenience, the project has already been configured to use the local repository for include and library directories.
   1. Verify that the Additional Include directory is set to the appropriate OpenCV include folder in the project settings:  
      
   2. Verify that the additional include directories points to the appropriate OpenCV lib directory:  
      
6. There **are two ways** for enabling program access to the required EXE and DLL files:
   1. **IF YOU DON’T WANT TO ADD <root>/opencv/bin to your system PATH** you will need to copy **<root>/opencv/bin** into your project folder **<root>/FastFreak\_Stereo.** *This was not done by default to keep the directories clean, and is not this authors preference.*
   2. If you don’t mind playing with your system PATH variable, simply add **<root>/opencv/bin** to your path.
7. Finally, ensure that the **opencv\_world330d.lib** library file from your OpenCV distribution is included in the **Linker🡪Input🡪Addition Dependencies** project setting.
8. Build and Run the Debug x64 project as normal, I prefer the “**F5**” key for consistency.

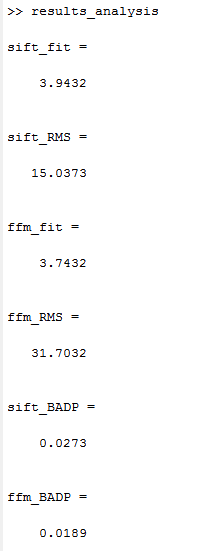
# Instructions for **Using** the program

The output console can be used to view events in the application as they happen. In general, the application will pause for each output picture, for which you will need to press a key to continue. You can disable most of these waitKey() events by setting DO\_ALL\_WAITKEYS to false. Similarly, most images can be suppressed with setting SHOW\_DEBUG\_IMAGES to false.

Once the program has been run, the final waitKey() event is guaranteed to pause the program and the results can been found on the console. Note that the Matlab results have been post process from a previous run since functions could not be completed in time for this submission. Example:



To replicate the Matlab results, simply run the <root>\Matlab\results\_analysis.m script. The results will be displayed in the console like so:



Representative Output

