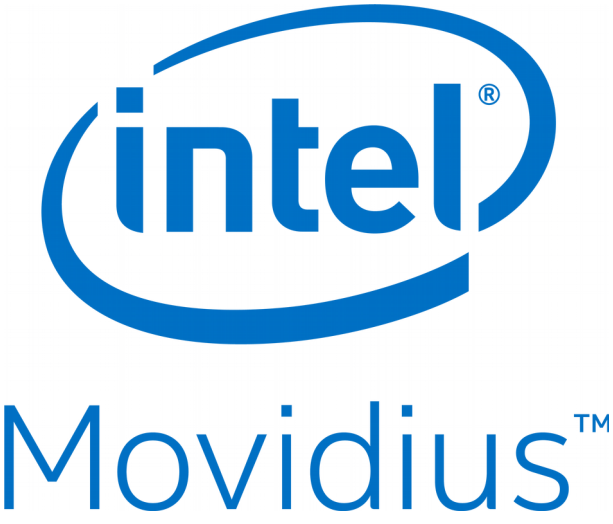


MA2xxx Robot Vision Evaluation Kit

User Manual

v1.02 / July 2018

Copyright and Proprietary Information Notice

Intel® Movidius™
2200 Mission College Blvd
M/S SC11-201
Santa Clara, CA 95054
http://www.movidius.com/

Revision History

Date	Version	Description
July 2018	1.02	Updated with new configuration options and details of new host GUI
May 2018	1.01	Corrected typo in section 4.1 Software Partitioning regarding Stereo and vTrack SHAVE use.
May 2018	1.0	Initial version.

Contents

1 Introduction

This document describes the Robot Vision Evaluation Kit: Myriad application, visionHal and example PC applications.

The Robot Vision Evaluation Kit has the following setup, which can be used to evaluate Myriad VPI targeted computer vision algorithms:

- Host
  - Configures the CV algorithms for various supported use cases.
  - Downloads any algorithm specific configuration and initialization data.
  - Issues the command to start Myriad processing.
- Myriad Input sensors
  - Synchronized stereo camera pair.
  - IMU sensor.
- Myriad high-level processing
  - Simple AEC on the input frames.
  - Rotation matrix generated from the IMU samples.
  - One or more nodes of algorithm/specific processing which produce metadata.
- Output
  - A consumer sends all the collected data (frames, metadata, IMU) to host.

The Robot Vision Evaluation Kit standardizes the Myriad application in a framework and presents a unified picture of the captured and generated data to the host.

The host can request stereo depth results only, vTrack results only, or both of them together, generated in parallel.

The evaluation kit also provides a simple, generic host interface to query the required data.

2 Overview

2.1 Description

The Evaluation Kit has four main parts:

1. Myriad FW producing data.
2. A simple PC interface to have access to the data (visionHal).
3. Algorithm specific visualization and evaluation components.
4. PC applications.

2.2 Application use-cases

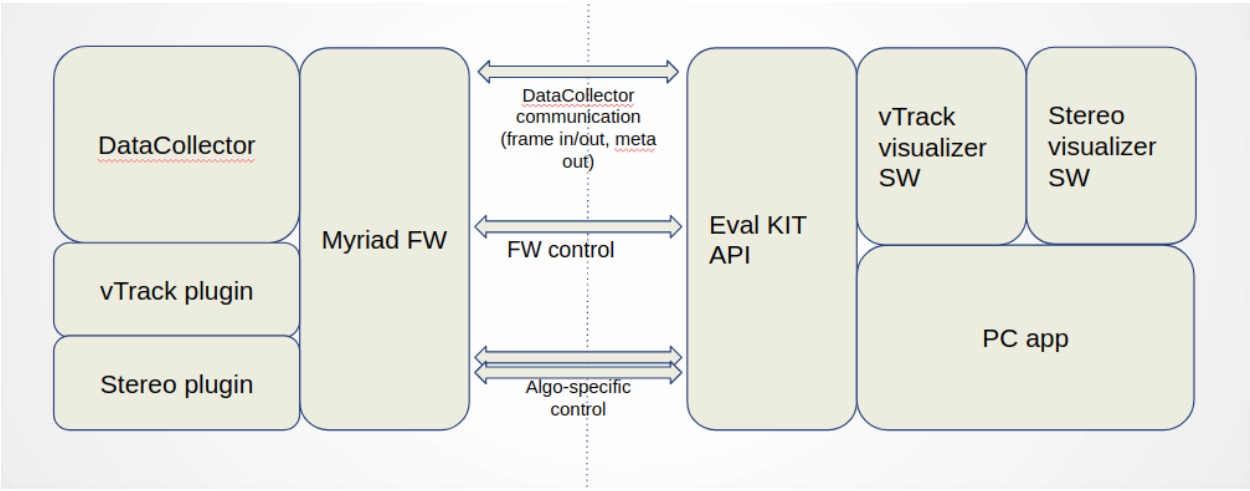
This evaluation kit serves a couple of use-cases:

- Collect raw datasets from a stereo camera +IMU setup.
- Demo vTrack and Stereo.
- Show live KPI data about vTrack.
- Show live KPI data about Stereo.

All the use-cases will share a common Myriad FW configured by a standalone PC application.

3 Architecture

3.1 Logical block diagram



3.2 FLIC Pipelines and plugins

The Myriad application consists of three main building blocks:

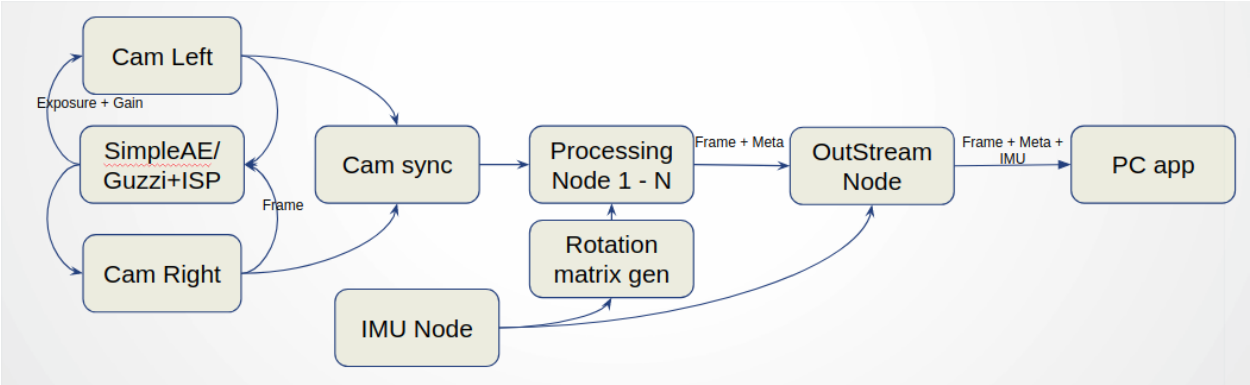
- Firmware control: reads algorithm configuration from the host and starts the pipeline. In other words, visionHal (PC side) is communicating with this module (Myriad side) to create and start the desired pipeline.
- Image Processing FW (DataCollector): common pre- and post-processing for evaluating CV algorithms. This module is responsible of generating all the inputs for the processing plugins, take their output in a standardized way and send them to the host.
- Algorithm plugins: Currently vTrack and Stereo.

3.2.1 Firmware control

This component allows the host API to boot and control the Myriad FW. The control includes selecting the processing components (algorithms), algorithm control, preprocessing policy and camera frame rate selection. The controls will be described in a later section where we present the host API. The implementation will use XLink channels to implement the communication between host and myriad.

3.2.2 Image processing FW (Data Collector)

The image processing framework on the Myriad is implemented as a FLIC pipeline. A basic diagram of the FLIC plugins used can be seen on the diagram below. The framework is designed such that multiple algorithm plugins can be added without much extra effort.



The elements of the diagram above are logical blocks, not necessarily FLIC plugins. The translation of this diagram to FLIC plugin diagram can be found in later sections.

- Cam Left + Cam Right
  - FramePump plugins providing raw image frames. They simply configure the camera and provide RAW frames.
  - The cameras can be configured to any frame-rate between 1 and 100 FPS.
  - Evaluation kit supports the OV9282 daughter card at a resolutions of 1280x720 and 640x400 (with binning).
- SimpleAE
  - Takes the camera frames, computes image stats and determines a new value for the camera exposure and gain. It configures the camera with these new values.
- Camera sync
  - Takes image frames and synchronizes them to produce a stereo pair. Here, the synchronization only handles eventual frame drops, the lower level synchronization being solved at camera source plugin level with HW connections.
- IMU node
  - Configures and takes samples from the on-board IMU sensor.
- Rotation matrix generator
  - Takes gyro samples from the IMU node and integrates them to a rotation matrix. Certain CV

algorithms use this data to improve performance and accuracy.

- The processing node(s)
  - A plugin that inherits from visionBaseFlicPlg. All CV algorithms are written to have this plugin as a base, so the user can easily plug in any of them into the pipeline. There is support for multiple processing plugins producing independent metadata. There is also support for using multiple copies of the same plugin to increase frame rate. They all produce an optional output frame and some algorithm specific metadata.
- Out stream plugin
  - Collects the data produced by the plugins in the pipeline and sends them to the host in a standard form. The data will contain: RAW/preprocessed frames, algorithm output frames, IMU samples, algorithm metadata.

3.2.3 Processing node(s)

DataCollector allows plug in of multiple processing nodes to the pipeline. Each plugin must be a `visionBaseFlicPlg`. vTrack and Stereo, both inherit from `visionBaseFlicPlg`.

The user application can configure one or multiple instances of visionBaseFlicPlg producing different metadata (vTrack in parallel with Stereo). All these processing nodes will get the input stereo pair with a rotation matrix and can produce metadata.

Additionally, there is support for creating multiple vision plugins which will work on different frames in parallel to achieve higher frame-rate with fixed latency.

3.2.4 Resource requirements

Most of the resources will be used by the processing plugins. For more details around the resource usage of the specific plugin, check the documentation of the algorithm (and/or FLIC plugin).

The following sub-sections will detail just the application resource usage without the actual processing nodes.

3.2.4.1 Memory requirements

The allocated memory will be either on heap or managed by FLIC pools. The pipeline needs some buffers for frames. The number of the buffers is configurable, and it depends on the amount of processing plugins used.

For example, in case of a Myriad X stereo, which runs at 10 Hz and it is configured to run on three threads, you will need ~12 frame buffers: 6 frames will be blocked by stereo, at least 2 by camera, and in certain cases (when the input frame is directly sent to the HOST), 2 by the transmission to host.

There is also a need for the output metadata. The number of metadata buffers needed is similarly dependent on the number of the processing nodes. To keep generality (for now) these buffers are fixed size, which covers the current use-case requirements.

3.2.4.2 HW requirements

When simple AE is configured, the RAW block is used. The filter is part of a SIPP pipeline. There are no other filters or SHAVEs used.

3.2.5 Performance

The application supports 30Hz in general use-cases. While both vTrack and Stereo are capable of much higher framerates, the bottlenecks for this can be:

- Camera source plugin performance.
- Auto-exposure algorithm should be configured to give maximum 16ms exposure.
- xLink needs to be able to transmit the data with the above mentioned frequency.

3.3 Components

The overall application will use DataCollector, FramePump, simpleAE, FLIC, vTrack, Stereo and xLink components from MDK. Note that the components above may depend on other components which are not listed.

3.4 Interfaces

Both the FramePump source plugin and the CDK ISP plugin are producing `ImgFramePtr` FLIC messages. This will allow easy switch between the two components.

The IMU source plugin is using the gyro driver and produces a FLIC message containing an IMU sample as defined in the `OsIMUHal.h`. The rotation matrix generator plugin is compatible with this output.

The stereo sync plugin gets the `ImgFramePtr` from the source nodes and after synchronizing, it sends a pair of `ImgFramePtr` messages.

The MIG FLIC plugin has two `ImgFramePtr` inputs, which allows to connect to the synchronization plugin. Use of this plugin with a stereo producer is not enforced, so in theory it can be connected directly to any `ImgFramePtr` producer.

The output plugin has inputs for the IMU, frame and metadata. This is an application specific plugin describing the communication between the host and Myriad.

The DataCollector is creating the connections between the plugins described above. The number of `visionBasePlugins` will depend on the `DataCollectorStart` parameters. The parameters are acquired from the host side through the provided API.

3.5 Application control

The application is controlled through the host visionHal API. Each PC application will need to use the API specified in `visionHal.h`. The main functions are the following:

- `visionError init(char* mvcmdName);`

Boots the Myriad chip and sets up initial communication through an xLink channel named “Config” (for now).

```
visionError registerFrameListener(frameHandler handler);
```

```
visionError registerVtrackListener(vTrackHandler handler,
iVisionAlgorithm* algo);
visionError registerStereoListener(stereoHandler handler,
iVisionAlgorithm* algo);
```

Each register function above will create and activate the corresponding processing node on Myriad. The callback function will be called each time when data arrives from the Myriad. The algorithm object must be created before calling this function. The supported algorithms are Stereo and vTrack. They both get different configuration options in the constructor. This configuration will be sent to the Myriad application and the corresponding FLIC plugin will be created.

- `visionError start();`
  - Start command for the myriad Firmware.
  - After this point, no listener can be added or configured.

- `visionError stop();`
  - Sends a stop command to Myriad.
  - Myriad stops the DataCollector FLIC pipeline.

Steps to follow to add a new algorithm:

1. Extend type enum.
2. Implement algorithm specific `iVisionAlgorithm` class.
3. Implement algorithm specific `getConfigSerialization` function which serializes the configuration for Myriad on a given xLink channel and will be called in the handler registration function.
4. Implement on Myriad the deserialization of the configuration.
5. Implement the algorithm specific vision plugin on Myriad.
6. Add new register handler function if needed.
7. Implement visualizer for the given data.

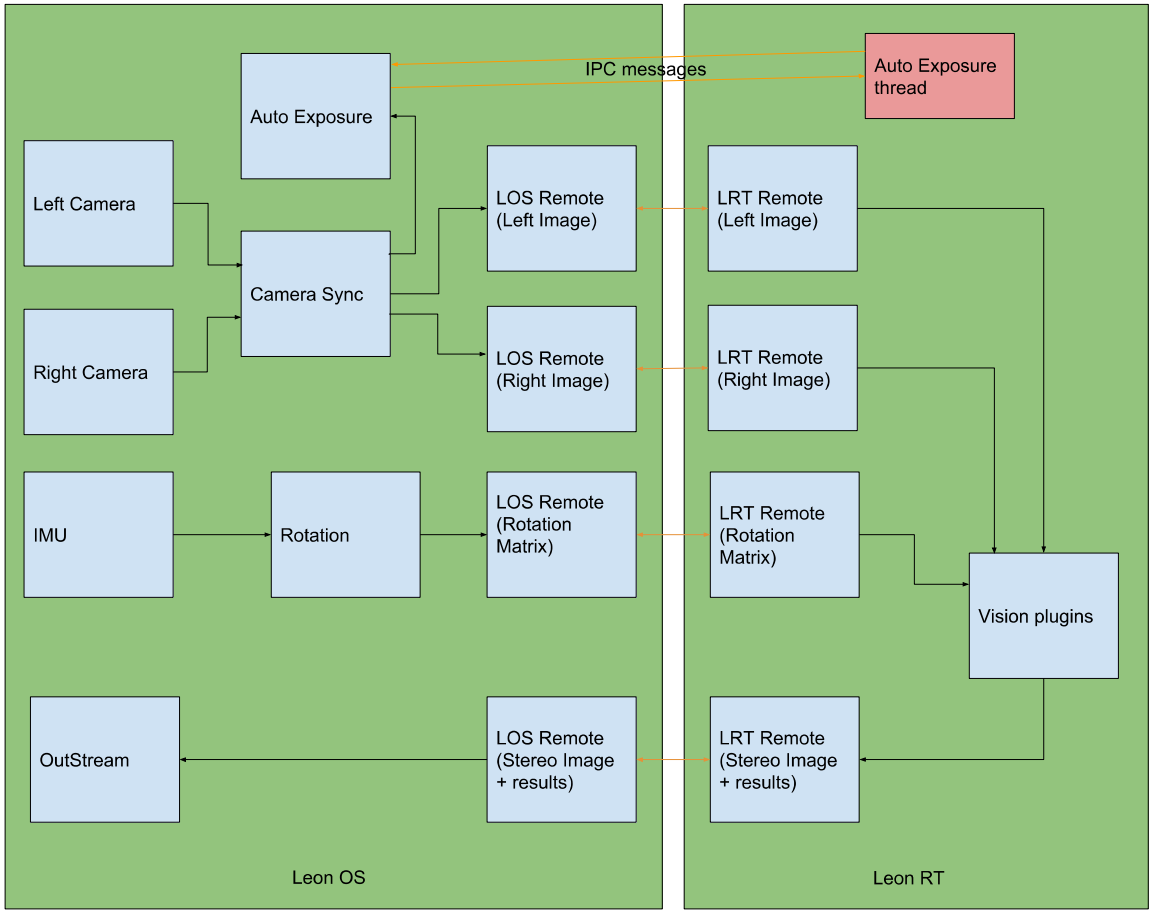
3.5.1 Resource concurrency

The HW filters will be protected entirely by the SIPP framework. Any usage of a filter outside the SIPP pipeline will need to be protected manually by the code code writer. SHAVEs will be protected by the shave driver itself, but the system integrator should also prefer assigning different SHAVEs to the algorithms.

4 Deployment and System integration

4.1 Software Partitioning

The Software pipeline is running on LeonOS (LeonMSS) and LeonRT (LeonCSS). The split can be seen in the diagram below.



Blue boxes: FLIC plugins **Red boxes:** non-FLIC

The vision plugins are running on LRT and can use other resources depending on the plugin. This application configures Stereo to use SHAVEs 0, 1, 2, 3, 4, 5 and vTrack to use SHAVEs 6, 7, 8, 9.

4.2 Configurations

In the current version, the vTrackStereo app contains minimal configuration possibilities:

- **Choice of algorithms**
  - Run the vTrack or Stereo algorithm.
- **Visualizations**
  - Show the graphs and histograms with vTrack (detailed visualization) or only the image with tracked features (basic visualization).
- **vTrack motion estimation type**
  - The motion estimation can use a dense block matching algorithm (MEST) or traditional Lucas-Kanade optical flow.
- **Stereo camera calibration**
  - The calibration of the cameras can (and should) be passed in for the stereo algorithm to work reliably.
- **Stereo confidence slider**
  - The stereo confidence can be changed dynamically while the app is running using the slider in the PC GUI

4.3 Physical setup

1. To use this application you need the following:
   a. MV0182, MV0212 or MV0235 development board with stereo cameras (MV0250 with 2 x OV9282)
   b. USB cable connected to PC (using USB3 cable and port is preferable for greater speeds).
2. Calibrate the cameras to generate a resulting .bin calibration file. Instructions for calibration are available in the AN004_MA2x8x_StereoCalibration.pdf document
3. In order to have permissions for the USB device (without `sudo`), you need to set up the `udev` rules for the device (note that you need to do this only once on a computer):

```
cd robotVision/utils && sh installRule.sh
```

5 Host side

The host side serves two purposes: (1) application control, (2) output visualization.

When the Myriad application starts, it awaits for a configuration message from the host. This configuration message contains the algorithms to be started along with the user determined configuration options. Current options are mentioned in section 4.2.

The output visualizations for stereo and vTrack are described in the component technical specs.

In this release, there are two C example applications: dataRecorder and vTrackStereo and a python GUI application. They show the usage of the visionHal class for a basic camera frame recording app and one that runs the vTrack and/or Stereo algorithms.

5.1 Host GUI

The python GUI is the recommended way to view the vTrack and Stereo performance. It can be found in mdk/robotVision/PcWxPython. Before first use, install the packages specified in the Readme.

To start the application: *python vTS_GUI.py*

Use ‘Connection’ from in the menu bar to configure the Algorithm and resolution required and provide the calibration file.

Once the Myriad app has started and is ready, click Connection → Connect.

The required algorithm should then start.

5.2 Stereo visualization

The stereo output can be seen as a heatmap in the ‘stereo vision’ tab as well as the ‘3D visualizer’ tab.

5.3 vTrack visualization

vTrack output can be visualized in the ‘vTrack’ tab. There is a button to show stats which displays vTrack statistics. These statistics are described in the vTrack component document.