Movidius
an Intel company

MDK Disparity Map Example

Application Note

v1.0

Movidius Confidential

Copyright and Proprietary Information Notice

Copyright © 2017 Movidius, an Intel Company. All rights reserved. This document contains confidential and proprietary information that is the property of Movidius Ltd. All other product or company names may be trademarks of their respective owners.

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

Table of Contents

1 Software Architecture

1.1 Introduction

Disparity Map is a cross-compilable example that runs on Myriad 2 chip and PC.

1. Myriad 2 project

   `mdk/examples/Demo/DisparityMap/myriad`

   The dense depth map is computed for each pair of left and right images captured by IMX208 sensors of 640x480 resolution. A rectification is applied on the right image for a better quality of the disparity map.

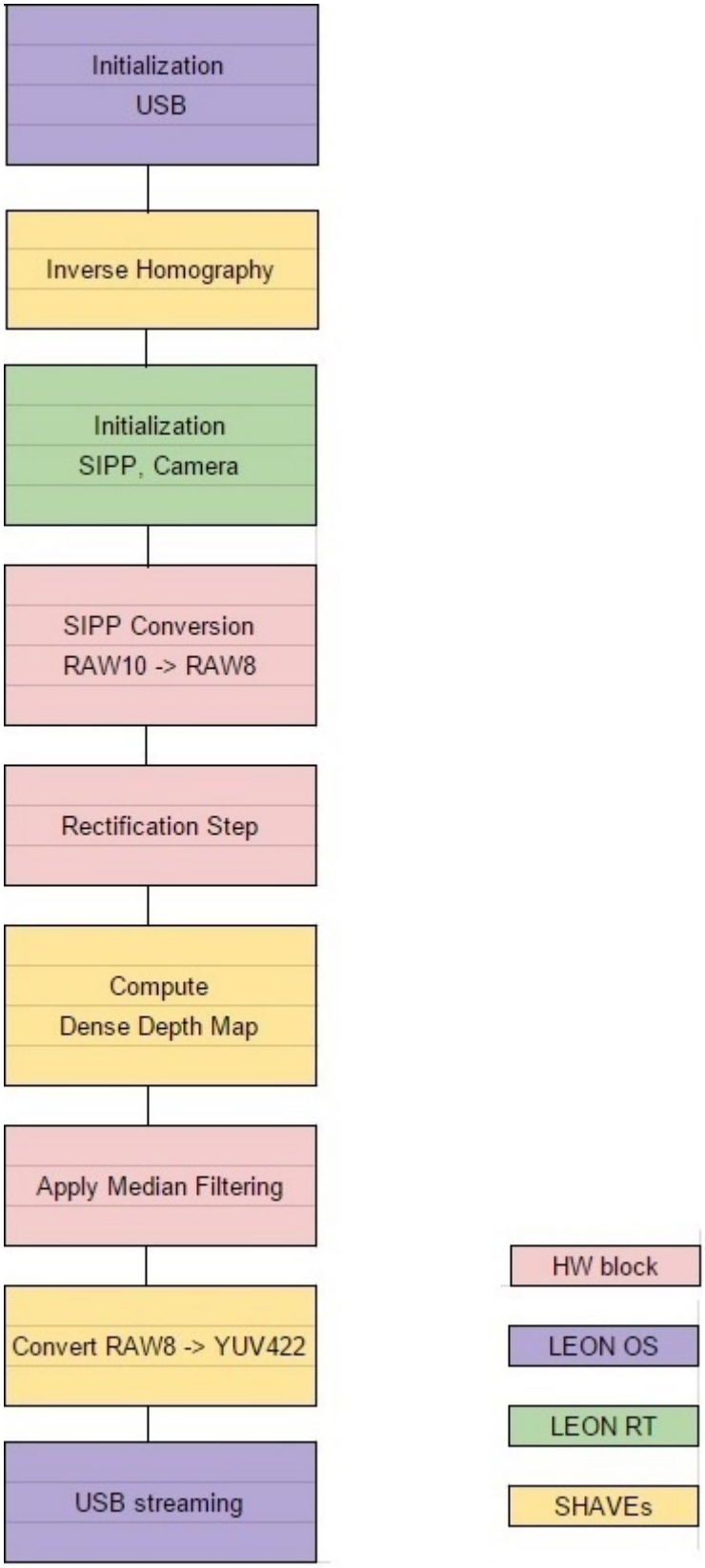
   The output is converted to YUV422 and streamed via USB.

2. PC project

   `mdk/examples/Demo/DisparityMap/pc`

   The dense depth map is computed for one pair of left and right images from the Middlebury dataset of 640x480 resolution.

1.2 High Level Architecture



2 Detailed Architecture

2.1 Requirements

The rectification step requires a Homography that is computed offline using OpenCV's function `findHomography()` after having detected the corners in both left and right images.

2.2 Camera Configuration

The IMX208 stereo sensor is connected in the following way:

- sensor 1 – CIF (RAW10).
- sensor 2 – SIPP MIPI (RAW10).

Sensor's resolution is 968x548 and images are center cropped to 640x480.

Camera runs at 30Hz and is configured in BINNING mode.

2.3 SIPP Conversion RAW10 → YUV400

A Lookup Table is used for converting the camera input from RAW10 to YUV400 needed for the algorithm.

2.4 Rectification Step

Rectification is done by a hardware block and consists in translation, rotation and warping.

The inverse of the precomputed Homography is needed, H^-1

$$L * H^{-1} = R$$

Rectification is applied on the right image in order to align the left and right images on the horizontal axis. This alignment offers the possibility to search in a range of maximum 64 disparities.

2.5 Disparity Map Computation

4 SHAVE processors are used and each one processes a patch of 640x130 of the image. Leon RT is in charge of splitting the image into 4 patches and reconstructing the final disparity map out of patches.

Disparity Map is computed by applying the following transformations to the images:

1. Census Transform (5x5 block) for left image
2. Census Transform (5x5 block) for right image
3. Census Cost
4. Census Minimum – Winner Takes All
5. Median Filtering

2.6 Conversion RAW8 → YUV422

An asm function is used to convert YUV400 algorithm's output to YUV422 to be streamed on USB.

2.7 USB Streaming

When a frame is finished being processed the USB is notified that he can stream it.

2.8 Memory footprint

The configured system frequency is 500MHz.

Processors used:

- LEON OS for USB code.
- LEON RT for camera streaming and managing image patches.
- SHAVES 0-3 for Depth Map computation.
- SHAVE 0 for preprocessing(H^-1) and postprocessing(YUV422 conversion).
- SHAVE 4 for SIPP internal lines.

Memory map:

1. CMX

   Each SHAVE requires ~70 KB for internal line buffers.

2. DDR

   Buffers for frame cameras, rectification, disparity maps and USB are located in DDR.

3 Performance

3.1 Optimized design

The processing is optimized in order to obtain 30 FPS:

- Frame N: capture frames.
- Frame N-1: rectification step.
- Frame N-2: dense depth map computation.

3.2 Performance

The table below shows performance for major processing steps.

Algorithm step	Measured ms @500Mhz
Preprocessing (H^-1)	0.01
Conversion RAW10 → YUV400	3.35
Rectification	16.24
Disparity Map Computation	22.13
Median filtering	0.82
Conversion YUV400 → YUV422	1.60