MA2x5x Stereo Software Module

Software Specification

v1.0 / May 2018

Contents

1 Introduction

1.1 Scope

This document describes the API and functionality of the Movidius Stereo component. As it is a component, different application integrations are possible. In addition to describing the API, the usage of the component in an example app is described.

1.2 Features and limitations

The Stereo Evaluation Kit is a component optimized for the Myriad 2 platform. It currently supports VGA resolution and is running on 6 SHAVE processors.

Main features of the stereo evaluation kit:

- VGA resolution camera (640x480 or 640x400).
- Synchronized stereo camera inputs.
- Single auto-exposure mechanism based on statistics for both cameras.
- Narrow FOV input for improved accuracy at farther distances.
- Fixed disparity range (128).
- Dynamic loading of SHAVE applications.

Main features of the stereo library:

- Image preprocessing and rectification of inputs.
- Advanced descriptor computation.
- Semi-global matching for improved performance on low texture input.
- Post-filtering of disparity map with confidence threshold.

	Baseline
Synchronized stereo cameras	Yes
Dynamic loading of SHAVE applications	Yes
Configurable input resolution	No
Configurable disparity	No
Auto-exposure	Yes
Confidence Thresholding	Yes

Table 1: Supported features

1.3 High-level overview

Any user application will have to include the following header files:

- DisparityMapApiDefines.h – contains type definitions required for the Stereo library.
- DisparityMapApi.h – contains prototype definitions for the Stereo library.

1.4 Processor usage

The stereo component requires 6 processor cores and two threads in the following way:

- 1 SHAVE for preprocessing.
- 5 SHAVES for stereo matching cost aggregation.

If dynamic loading is being used, the user can choose which particular SHAVEs are available to the library when the library is being dynamically loaded.

The component also needs to run a portion of the code on a LEON processor, providing the glue layer under the API functions that are called by the application, which typically also runs on LEON, to the stereo component SHAVE code.

1.5 I/O buffers

	Resolution	**Bit depth**	**Notes**
Input Buffers			
L / R	640x480	8	rectified
L / R downsampled	320x240	8	rectified
Output Buffers			
Disparity Map	640x480	16	4 fractional bits

Table 2: Input and Output Buffers

1.6 Memory usage and layout

Stereo component SHAVE code is placed in DDR and the associated data in CMX.

The location of an extra CMX slice, used for the Auto-Exposure pipeline, can be configured in the application makefile (`SIPP_AEC_SLICE_NUMBER`).

1.7 DMA usage

The stereo component uses DMA, both for faster transfers, and to speed up the depth algorithm, as a number of operations can be performed in parallel with the DMA transfer.

Each SHAVE will initialize a DMA requester ID at the beginning of each new frame processing by calling `dmaInitRequester()`.

1.8 Cache configuration

The LEON data cache should be configured as write-through cache memory to avoid possible SHAVE L2 cache coherency issues.

Shave L2 needs to be flushed when using same SHAVE to run different dynamic loaded applications (i.e., convert, warp, downsample, stereo library).

2 Component structure

2.1 Folder structure

The folder structure follows that of a typical MDK component:

common/components/stereo/

- sgbm/leon/ → Leon code for stereo component
- sgbm/shave/ → SHAVE optimized code for stereo component
- sgbm/shared/ → Definitions for stereo component
- downsample/shave → Downsample SHAVE application
- shared/ → Definitions for Leon-Shave communication

The stereo component is wrapped in a FLIC plugin:
(see also MDK-MA2xxx_RobotVisionEvalKit_UserManual.pdf)

common/components/visionStereoPlg

- arch/ma2x5x/leon/rtems/include → Plugin definitions
- arch/ma2x5x/leon/rtems/src → Plugin for stereo component

2.2 Exported files

The following files are part of the generic stereo API and should not be modified:

- disparityMapApi.h
- disparityMapApiDefines.h
- defines.h

3 Passive stereo data structures definition

3.1 Stereo parameters

3.1.1 UserConfig

3.1.1.1 Description

A handle to the stereo parameters to be passed on to the SHAVEs.

3.1.1.2 Definition

```
typedef struct
{
    frameSpec*              frame;
    u32                     maxDisparities;
    u32                     confidenceThreshold;
    u32                     list_of_shaves_thread_1[SHAVES_USED];
    u8*                     aggCostPaths[][];
} stereoUserConfig_t;
```

3.1.1.3 Fields

Values	Description
`frame_spec`	Stereo input frame size and type.
`maxDisparities`	Disparity range in pixels.
`confidenceThreshold`	Invalidate pixels threshold based on confidence value.
`list_of_shaves_thread_1`	List of SHAVEs used for stereo thread.
`aggCostPaths[][]`	List of shared pointers in SemiGlobal Mathing algorithm.

3.1.2 TileConfig

3.1.2.1 Description

A handle to the input and output addresses of left/right images from each pyramid level.

3.1.2.2 Definition

```
typedef struct
```

```
{
    frameBuffer            leftImage;
    frameBuffer            rightImage;
    frameBuffer            disparityMap;
    frameBuffer            confidenceMap;
    u32                    censusKernelSize;
    u32                    disparities;
} stereoTileConfig_t;
```

3.1.2.3 Fields

Values	Description
`leftImage`	Left image for specific pyramid level.
`rightImage`	Right image for specific pyramid level.
`disparityMap`	Disparity map for specific pyramid level.
`confidenceMap`	Confidence map for specific pyramid level.
`censusKernelSize`	Kernel size for Census transform.
`disparities`	Number of disparities for specific pyramid level.

3.1.3 SGBMConfig

3.1.3.1 Description

A handle of Semi-Global Matching input and output matching costs.

3.1.3.2 Definition

```
typedef struct
{
    u8*                    censusCost;
    u8*                    aggregatedCost;
    u32*                   penaltyTableP1;
    u32*                   penaltyTableP2;
    u32                    currentBuffer;
    u32                    currentLine;
    u32                    width;
} SGBMConfig;
```

3.1.3.3 Fields

Values	Description
`censusCost`	Disparity cost obtained from Census matching for each image tile.

Values	Description
aggregatedCost	Aggregated cost obtained from SGBM for each image tile.
penaltyTableP1	Penalty values for cost aggregation when the minimum disparity on path is found at +/-1.
penaltyTableP2	Penalty values for cost aggregation for larger disparity changes.
currentBuffer	Current index of the line buffers (current line or path line).
currentLine	Current line with respect to the aggregation paths.
width	Width of the tile in pixels.

4 Passive Stereo API

4.1 Control Mechanism

In order to compute a disparity map, the application must do the following steps:

1. Allocate needed frame buffers holding:
   - ❑ input frame – 1 byte per pixel format, size of each input frame is width * height.
   - ❑ warped input frames – 1 byte per pixel format, size of each warped frame is width * height.
   - ❑ downsampled input frames – 1 byte per pixel format, size of each downsampled frame is width * height / 4.
2. If using dynamic loading, copy the module data for each thread. This is needed due to a limitation in Shave Dynamic Loading where there is a single module data instance for all threads.
3. Warp the input images using the provided mesh.h (each stereo module comes with a mesh.h file that contains the rectification parameters).
4. Downsample the warped images.
5. Call the SGBM SHAVE component that will output a disparity map in u16 format having subpixel precision.

4.2 disparityMapRunFrame

4.2.1 Prototype

```
void sgbm(u8* costVolume,
                stereoUserConfig_t*   userCfg,
                stereoTileConfig_t*   tileCfg);
```

4.2.2 Arguments

Values	IN/OUT	Description
costVolume	IN	Pointer to matching cost volume computed on first thread.
userCfg	IN	Pointer to user configuration.
tileCfg	IN/OUT	Pointer to pyramid configuration for each level.

4.2.3 Functionality

This call allocates the CMX memory needed for stereo and computes the disparity map using SemiGlobal Matching with 5 aggregation paths and subpixel interpolation.

4.3 DMA transfers

The current stereo library implementation uses the Myriad 2 DMA driver calls from `scCmxDma.h`:

- `ScCmxDmaInitialize`
- `ScCmxDmaCreateTransaction`
- `ScCmxDmaLinkTransactions`
- `ScCmxDmaStartTransfer`
- `ScCmxDmaWaitTransaction`

5 Runtime Performance

5.1 Myriad 2 Performance

5.1.1 Myriad 2 Pyramidal (VGA)

The pyramidal version supports VGA output disparity map with decent subpixel information.

Thread	Runtime
Thread 0: Preprocessing	23 ms
Thread 1: SemiGlobal Aggregation	50 ms

Table 3: Input and Output Buffers

The application has a latency of 60 ms:

- camera readout: 10 ms
- stereo thread: 50 ms

5.1.2 Myriad 2 Nonpyramidal (VGA)

The nonpyramidal version supports QVGA output disparity map with very strong subpixel information.

Thread	Runtime
Thread 0: Preprocessing	23 ms
Thread 1: SemiGlobal Aggregation	30 ms

Table 4: Input and Output Buffers

The application has a latency of 40 ms:

- camera readout: 10 ms
- stereo thread: 30 ms

5.2 Myriad 2 quality KPIs

Depth quality metrics include pixel error thresholds (Middlebury table) and f-score and were measured using Middlebury dataset.

The metrics above refer to Myriad 2 Pyramidal stereo due to full resolution ground truth availability.

	Error % T=0.125	Error % T=0.25	Error % T=0.5	Error % T=0.75	F-score beta=1
M2 stereo	26.6	22.8	19.9	17.7	0.894

Table 5: Pixel error threshold KPI

The table above shows metrics for a confidence value of 190 and it can be interpreted as:

- 26.6 % of the non-occluded pixels have a difference more than 0.125 compared to the ground truth and are considered errors.
- f_score is a measure of disparity accuracy and a value closer to 1 is better.