IVCAM2.0 3D Imaging Camera



ASIC A0 Joined Filters (JFIL) specification

1 March 2017

Revision 0.1.1

Intel Top Secret

Table 1: Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matlab Version | Revision Number | Revised by | Description | Revision Date |
| TBD | 0.1.0 | David Silver | Initial Release | 8 May 2016 |
| TBD | 0.1.1 | David Silver |  | 21 June 2016 |
|  |  | Omer Sella | Revised invalidationFilter | 16/08/2016 |
| 0.57 |  | Omer Sella | Revised version | 01/09/2016 |
|  |  | Omer Sella | Match to latest arch.   * Changed according to Visio. * Reordered filters. * Revised sort. * Added upscaler. | 03/09/2016 |
| 0.6.36 |  | Omer Sella | Added upper and lower thresholds to invalidation filter for clipping. | 05/09/2016 |
|  |  |  | Unified depth and IR description.  Updated visio.  Corrected GAMMA  Changed Invalidation registers | 13/09/2016 |
| 0.6.37 |  | Omer Sella | Matched invalidation filter logic to code.  Changed register names to match code as well. | 18/09/2016 |
| 0.6.37 |  | Omer Sella | Added test plan.  Added constraints. | 21/09/2016 |
| 0.7.39 |  | Ohad Menashe | Range finder support | 24/09/2016 |
| 0.7.39 |  | Omer Sella | Review towards 0.7   * Explain that the LUT for the gamma function is a linear interpolation LUT * Review and corrections to Gamma | 26/09/2016 |
| 0.7.4 |  | Yoni Chechik | * JFIL & gamma bypass==1 bug fixed * Range finder through gamma * pipeFlags is non an input * Column buffer | 1/11/2016 |
| 0.7.41 |  | Yoni Chechik | Jfil bypass IR –overflow explained | 15/12/2016 |
|  | 0.9 | Ohad Menashe | Remove flags | 02-Feb-2017 |
|  | 0.9 | Ohad Menashe | Update block diagram | 05-02-2017 |
|  | 0.91 | Ohad Menashe | Update invalidation logic | 01-Mar-2017 |

7

Contents

[1 Introduction 4](#_Toc473800386)

[2 Memory and interfaces 5](#_Toc473800387)

[2.1 Interfaces 5](#_Toc473800388)

[2.1.1 Input 5](#_Toc473800389)

[2.1.2 Internal channels 5](#_Toc473800390)

[2.1.3 Output 5](#_Toc473800391)

[2.2 Memory 6](#_Toc473800392)

[2.2.1 Column buffer 6](#_Toc473800393)

[3 Depth/IR/Confidence Modules 7](#_Toc473800394)

[3.1 Introduction 7](#_Toc473800395)

[3.2 List of Depth Filters 9](#_Toc473800396)

[3.2.1 SortEdge Filter 9](#_Toc473800397)

[3.2.2 Sort Filter 9](#_Toc473800398)

[3.2.3 Depth Dependent Gradient Filter (DDGradient) 9](#_Toc473800399)

[3.2.4 Geometric Filter 9](#_Toc473800400)

[3.2.5 UpscalerXY 9](#_Toc473800401)

[3.2.6 Edge Filter 10](#_Toc473800402)

[3.2.7 Distance Dependent Bilateral Filter 10](#_Toc473800403)

[3.2.8 Feature Extraction (5x5) 10](#_Toc473800404)

[3.2.9 Neural Network (NNET) 10](#_Toc473800405)

[3.2.10 Invalidation Filter 10](#_Toc473800406)

[3.2.11 Gamma 11](#_Toc473800407)

[3.3 Range finder mode 13](#_Toc473800408)

[3.3.1 MaxPooling 13](#_Toc473800409)

[4 Appendix 14](#_Toc473800410)

[4.1 Registers 14](#_Toc473800411)

[5 Test Plan 15](#_Toc473800412)

[5.1 General randomization 15](#_Toc473800413)

Tables

[Table 1: Revision history 1](#_Toc455650164)

[Table 2: Depth filter sizes 6](#_Toc455650165)

[Table 3: IR filter sizes 6](#_Toc455650166)

[Table 4: Register 19](#_Toc455650167)

Introduction

The purpose of this block is to apply quality enhancement filters to the raw Depth and IR frames. As the CBUF block handles the non-rasterized scan data, the input to the JFIL is rasterized scan columns, occasionally with missing pixels (i.e. holes). The resulting depth, IR and confidence images are passed to the controller.

If the register regsJFILbypass ==1 then the JFIL block is bypassed. In this case we save only the 8 LSB’s of the IR (with overflow- ex: if a pixel is over 2^8-1- we put 2^8-1 == ‘FF’).



Figure 1: JFIL position in the pipe

In the case of range finder mode, data is skipped and received directly from the DEST block. The additional data from the DEST block is omitted.

Memory and interfaces

Interfaces

Input

The input to the JFIL is coming from the CBUF. The data is structured per image column. The data is four channels. Each pixel contains the following data:

1. DEPTH: 16-bit of depth data.
2. IR: 12-bit of IR data.
3. CONFIDENCE: 4-bits of confidence
4. General registers: there is global information that is shared among the pipe blocks (see Register table below). E.g., scale factor for the DN to mm conversion.

This block is the first to interact with the raw data arriving from the depth estimation. As the input data contains both depth, IR and confidence with the special case of depth equal zero as invalid pixel, there are 4 possible options:

|  |  |  |  |
| --- | --- | --- | --- |
| Depth | IR | Confidence | state |
| 0 | **NA** | **0** | Missing scan |
| 0 | **NA** | **[1-15]** | Illegal state |
| [1-65535] | **NA** | **0** | Illegal state |
| [1-65535] | **NA** | **[1-15]** | Valid pixel |

Table 2: Depth / Confidence combinations and their meaning

Internal channels

JFIL block includes internal data stream that is not part of the I/O, and is used for internal data deliveries between the units:

1. Features generation – data extracted from specific point in the pipe in favor of the neural network algorithm

Output

The output of the JFIL are 3 channels:

1. depth: 16-bit of processed depth data.
2. ir: 8-bit of processed IR data.
3. confidence: 4-bit of processed confidence data.

Memory

### Column buffer

If we have L filters, each filter of window size, then - the number of columns needed is:

Here is a table of the filters and the respective sizes:

|  |  |  |  |
| --- | --- | --- | --- |
| Filter | Size | Duplications | Effective addition to buffer size |
| SortEdge | 3X3 | 2 | 4 |
| Sort | 3X3 | 1 | 2 |
| Edge | 3X3 | 1 | 2 |
| Gradient | 3X3 | 2 | 4 |
| Geometric | 5X5 | 1 | 4 |
| Upscaler | 1X1 | 1 | 0 |
| Bilateral | 5X5 | 4 | 12 |
| Feature Extraction | 5X5 | 1 | 4 |
| DNN | 5X5 | 4 | 4 |
| INN | 5X5 | 4 | 4 |
| Gamma | 1X1 | 0 | 0 |
| invalidation | 1X1 | 0 | 0 |
| TOTAL |  |  | 40 |

Table 3: Depth filter sizes

We need 40 buffer columns of depth + confidence + IR image.

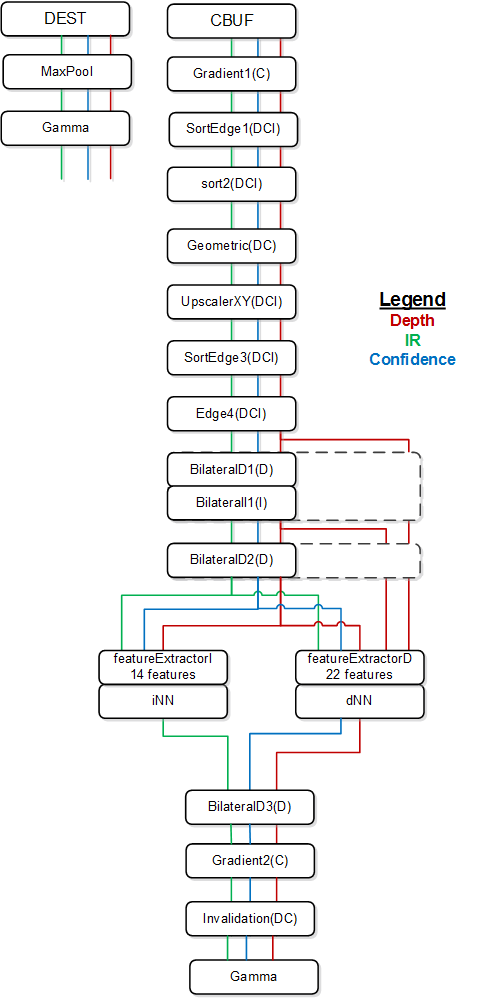
Depth/IR/Confidence Modules

Introduction

The JFIL block consists of several filters, some of them appear in the pipe more than once. A typical filter is operating as a square sliding window. The input can be either Depth, IR, or confidence. The output is usually diffused along the pipe to the next filter and in some cases it is sent to the final filter block (JDNN = Joined Depth Neural Network) as a feature vector.

In this document we will give an overview of the pipe and a simple description for each filter – mainly to align terminology across documents. Each filter will have its own document with self-contained description.

We start with a flow chart of the general block:

  
Figure 2JFIL block diagram

List of Depth Filters

### SortEdge Filter

This block act as a container for two filters: sortFilter and edgeFilter filters. This container appears twice in the pipe, labeled sortEdge1 and sortEdge3 with the corresponding registers regsJFILsortEdge1 and regsJFILsortEdge3 for the mux, where 0 means do sort, and 1 means do edge.

### Sort Filter

The sort filter appears twice in the pipe.

The filter sorts 9 values. It will then take a weighted combination of the pixel values based on their place in the sorted vector. For example, a weight vector of [0,0,0,0,1,0,0,0,0] will give the median depth value in the 3x3 window.

See additional documentation in sort.

### Depth Dependent Gradient Filter (DDGradient)

The DDGradient filter appears twice in the pipe. It is a filter to invalidate a pixel according to threshold of the difference between the pixel and its neighbors. The threshold is depth dependent (that is, dependent on the central pixels value).

See additional documentation in Gradient.

### Geometric Filter

This filter is looking on the validness neighborhood of a pixel (based on the confidence), and by finding the best match to the pattern of the validness neighborhood, decides whether to validate/invalidate the pixel. When it is validating a pixels it give it the median value based on pixels from the matching template.

See additional documentation in Geometric.

### UpscalerXY

UpscalerXY doubles the number of rows or columns of an image (but not both).

It does so by adding rows or columns of zeros between the original image rows or columns correspondingly. A zero value is given to the depth, IR and confidence.

Note – this filter either works on ALL streams (Depth, IR, confidence) or none.

The decision to upscale the rows or the columns is according to the registers RegsJFILupscalex1y0

A value of 1 means to upscale in the X dimension while a value of 0 means upscaling in the Y dimension.

Practically the pixel output rate is doubled with blank (i.e. zero) pixel for every second pixel (counting from the left and top).

In particular this means that that if an up scaling of the rows is done, then the new first line is the same as the input first line, the second line is all zeros etc.

Similarly if an up scaling of the columns is done, then the new leftmost column is the same as the input first column, the second column is all zeros etc.

Note that the maximum output after upscale cannot exceed SXVGA resolution.

If RegsJFILupscalexyBypass ==1 this filter is bypassed.

### Edge Filter

This filter looks for the best direction to interpolate the central pixel. That is, either using the two vertical adjacent pixels, or the horizontal or one of the diagonals.

See additional documentation in Edge.

### Distance Dependent Bilateral Filter

This filter appears three times in the depth data path. It smoothens the depth image by weighted average of the pixels in the windows. The weights are computed from three parameters: 1) the difference between the depth value of the pixels to the depth value of the central pixel. 2) The spatial distance from the central pixel. 3) The confidence value of the pixel (1& 2 are the “bi” in bilateral, the 3rd is the “cross”).

The tweaking of the strength of each of the parameters is controlled by registers and by the depth of the central pixel (this part is the “distance dependency”).

In addition to its output to the next filter, each bilateral filter has an output towards the Depth neural Network.

See additional documentation in Bilateral.

### Feature Extraction (5x5)

The feature extraction block is responsible for preparing features for the NND/NNI block. It contains:

1. convolution filters
2. A featureSort block that sorts a 3x3 window

See additional documentation in featureExtraction.

### Neural Network (NNET)

The Depth Neural Network receives various inputs which are called “features”. These are passed from several points in the pipe (timed to arrive at NND/NNI concurrently), and outputs depth and confidence. It can be bypassed by directing the input to the output.

See additional documentation in NN.

### Invalidation Filter

The purpose of this filter is to set the confidence AND depth values to 0 if the confidence level of a pixel is less than a certain threshold written in register space, or to use a selective invalidation.

The depth value is invalidated according the following rules:

* Depth is lower than minDepth
* Depth is higher than maxDepth
* If regsJFILinvUseGlobalConf:
  + confidence is lower than regsJFILinvConfThr
* else
  + for the current confidence value c, invalidate if depth<= LUT[c]

For example: if the register regsJFILuseGlobalConf is set to 0, then pixels with confidence 3 will be invalidated (set to depth 0 and confidence 0) if the depth value is less than LUT[2].

Gamma

Due to the human eye over sensitivity to dark tones an image is encoded using a gamma instead of linearly.

The gamma block reduces the 12-bit input of ir to 8-bit output.

If JFILgammaBypass is on: we simply throw the 4 LSB’s of ir.

Implement as nonlinear function (with LUT f) with scale and shift at the input and output:

Where:

|  |  |  |
| --- | --- | --- |
| Parameter | Matching register name | Number of bits (signed/unsigned) |
|  | regsJFILgammaScale[0:15] | 16, signed |
|  | regsJFILgammaScale[16:31] | 16, signed |
|  | regsJFILgammaShift[0:15] | 16, signed |
|  | regsJFILgammaShift[16:31] | 16, signed |

The calculation is done as follows:

1. is given in 12 bits of IR.
2. is a 16 bit signed, so is calculated, and the bit shifted by 10 bits to produce an 18 bit number.
3. Next we add where is a 16 bit number signed, so the result it 19 bits wide. The result may be a negative number, so we take the maximum between the result and 0.
4. Of the 19 bit result, only the 12 least significant bits are used, where in case we have a result which value is larger than 0xfff we take 0xfff instead (saturate).
5. The 12 bits are passed through a LUT containing 65 12bit values. A linear interpolation is used for “missing” values.
6. Next we multiply by another 16 bit signed number to obtain and bit shift by 10 bits which yields an 18 bit signed number.
7. Next we add which is a 16 bit signed number and obtain a 19 bit signed number. We take the maximum between the result and 0 to obtain a non negative number.
8. Of the 19 bit result, only the 8 least significant bits are used, where in case we have a result which value is larger than 0xff we take 0xff instead, which is the final result of the Gamma block and also the output.

Range finder mode

If the register regsGNRLrangeFinder == 0 this mode is turned off, otherwise it is on.

In this mode the data to JFIL arrives directly from DEST by skipping the CBUF module. Any additional entries that DEST generates/passes in favor of RAST are omitted.

In this block there are two valid pixel that are selected according to their confidence. The Selected pixel is passed to the output “as is”.

The range finder mode takes the input and passes it through MaxPool & Gamma blocks.

### MaxPooling

This filter operates only in range finder mode and receives the data from the DEST block. It hold a memory of a single reading (that is depth, ir, and confidence data, 32bit) and compare it to the current reading. This filter outputs the reading with the highest confidence value if the confidence value is higher than the predefined register RegsJFILmaxPoolConfThr .

Appendix

Registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Size** | **Default** | **Range** | **Special values/description** |
| **GNRL** |  |  |  |  |
| regsGNRLrangeFinder | 1 | 0 | 0-1 | 0 Means “no range finder”. 1 Means this mode is on. |
| **JFIL** |  |  |  |  |
| regsJFILbypass | 1 | 0 | 0-1 | Bypass the JFIL |
| regsJFILsortEdge1 | 1 | 0 | {0,1} |  |
| regsJFILsortEdge3 | 1 | 0 | {0,1} |  |
| regsJFILupscalexyBypasss | 1 | 1 | {0,1} | 1 means Bypass. |
| regsJFILupscalex1y0 | 1 | 1 | {0,1} | 1 means upscale x dimension. 0 means upscale y. |
| regsJFILinvUseGlobalConf | 1 | 1 | {0,1} |  |
| regsJFILinvMinMax[0:15] | 16 | 0 | 0-2^15-1 |  |
| regsJFILinvMinMax[16:31] | 16 | 2^15 | 0-2^15-1 |  |
| regsJFILinvConfThr | 4 | 4 | 0-15 |  |
| regsJFILgammaScale[0:15] | 16 | 1 | [(-2^15-1)-(2^15-1)] |  |
| regsJFILgammaScale[16:31] | 16 | 1 | [(-2^15-1)-(2^15-1)] |  |
| regsJFILgammaShift[0:15] | 16 | 1 | [0-(2^16-1)] |  |
| regsJFILgammaShift[16:31] | 16 | 1 | [0-(2^16-1)] |  |
| regsJFILdnnBypass | 1 | 0 | 0-1 | Bypass the JDNN |
| regsJFILinnBypass | 1 | 0 | 0-1 | Bypass the JINN |
| RegsJFILmaxPoolConfThr | 4 | 3 | 0-2^4-1 |  |
| RegsJFILbypassIr2Conf | 1 | 0 | 0-1 | In case that JFIL.bypass==1 && this JFIL.bypassIr2Conf==1 we put the 4 MSB’s of IR into conf and deprecate conf |

Table 5: Registers

Test Plan

## General randomization

Table 5 describes the general test randomization distribution.

Table 6: General randomization distribution

|  |  |  |
| --- | --- | --- |
| **Name** | **Value** | **Distribution (%)** |
| **JFIL** |  |  |
| regsJFILbypass | 0 | 99% |
| 1 | 1% |
| regsJFILdnnBypass | 0 | 99% |
| 1 | 1% |
| regsJFILinnBypass | 0 | 99% |
| 1 | 1% |
| regsJFILupscalexyBypass | 0 | 50% |
| 1 | 50% |
| regsJFILupscalexyMode | 0 | 50% |
| 1 | 50% |
| regsJFILsortEdge1 | 0 | 50% |
| 1 | 50% |
| regsJFILsortEdge3 | 0 | 50% |
| 1 | 50% |
| regsJFILinvConfThr | 1 | 20% |
| 2 | 20% |
| 3 | 20% |
| 4 | 20% |
| 5-15 | 20% |
| regsJFILinvMinMax[0:15] | 0-1000 | 60% |
|  | 1001 - 2000 | 20 % |
|  | 2001 - 2^15-1 | 20% |
| regsJFILinvMinMax[16:31] | 10001 - 2^15-1 | 30% |
|  | 5001 - 10000 | 50% |
|  | 0-5000 | 20% |
| regsJFILinvUseGlobalConf | 0 | 80% |
|  | 1 | 10% |
| regsJFILGammaScale[0:15] |  |  |
|  |  |
| regsJFILGammaScale[16:31] | (0-(2^16-1)) | 100% |
| regsJFILScale[0:15] | (0-(2^16-1)) | 100% |
| regsJFILScale[16:31] | (0-(2^16-1)) | 100% |