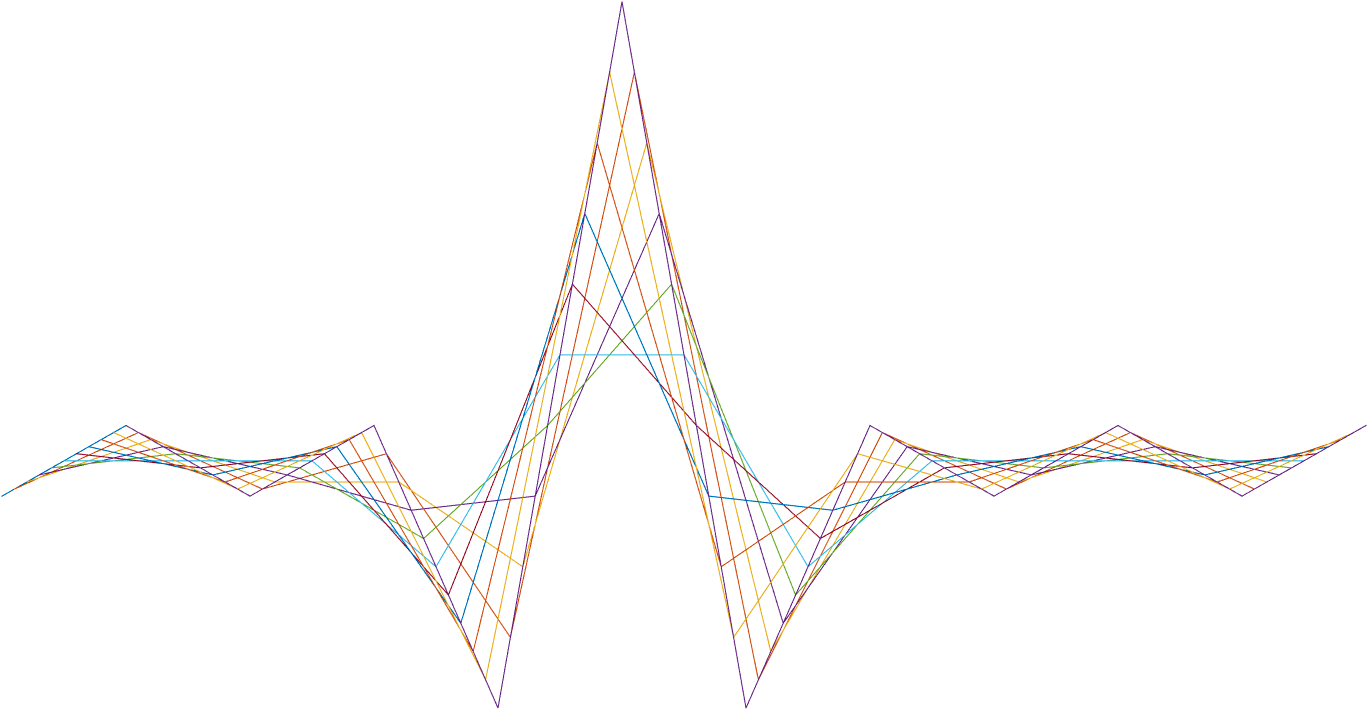
IVCAM2.0 3D Imaging Camera



ASIC A0 Depth Correlation specification

20 December 2017

Revision 0.9.0

Intel Top Secret

Table : Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matlab Version | Revision Number | Revised by | Description | Revision Date |
| TBD | 0.3.0 | Ohad Menashe | Initial Release | 24 Feb 2016 |
| 2016a | 0.5.0 | Vitaly Surazhsky | Detailed description | May 26, 2016 |
| 2016a | 0.5.1 | Vitaly Surazhsky | PSNR 6bit output, IR output can be replaces with NEST | June 16, 2016 |
|  | 0.5.2 | Yoni Chechik | -added pipe flags | 14/09/2016 |
| 0.7.46 | 0.7.0 | Vitaly Surazhsky | DCORbinTemplate, DCORbinTemplate added for power save | 3 Nov 2016 |
| 0.7.46 | 0.7.1 | Ohad Menashe | DCOR template mechanism | 10 Nov 2016 |
| 0.7.46 | 0.7.2 | Yoni Chechik | Coarse masking +tamplate length regs | 13 Nov 2016 |
| 0.7.47 | 0.7.3 | Vitaly Surazhsky | bypass | 16 Nov 2016 |
| 0.8.49 | 0.8.1 | Vitaly Surazhsky | Coarse masking 128 to 256 bits in 3.1.3 Flags description in 2.1.1, 2.1.2 Clarifications added in 3.4.4 | 28 Nov 2016 |
| 0.8.49 | 0.8.1 | Ohad Menashe | Index selector in Range finder mode | 29 Nov 2016 |
| 0.8.5 | 0.8.2 | Ohad Menashe | Correlation direction bug | 25 Dec 2016 |
| 0.9 | 0.9.0 | Vitaly Surazhsky | Section 3.6 added | 27 Dec 2016 |
| 0.9 |  | Yoni Chechik | Coarse masking renew | 29 Dec 2016 |
|  | 0.9 | Ohad Menashe | Coarse masking register order | 01 Feb 2017 |
|  |  | Ohad Menashe | CMA as IR | 07 Feb 2017 |
|  |  | Ohad Menashe | Replace range finder mode with template mode 3 | 22 Mar 2017 |
|  |  | Ohad Menashe + Yoni Chechik | PSNR appendix overview | 13 Nov 2017 |

7

Contents

[1 Introduction 3](#_Toc470704515)

[2 Interfaces 5](#_Toc470704516)

[2.1.1 Input 5](#_Toc470704517)

[2.1.2 Output 5](#_Toc470704518)

[2.1.3 Bypass 5](#_Toc470704519)

[3 Detailed description 6](#_Toc470704520)

[3.1 Coarse Correlation 6](#_Toc470704521)

[3.1.1 CMA Decimation 6](#_Toc470704522)

[3.1.2 Correlation 6](#_Toc470704523)

[3.1.3 Coarse masking 6](#_Toc470704524)

[3.1.4 7](#_Toc470704525)

[3.1.5 MAX detection 7](#_Toc470704526)

[3.2 Fine Correlation 7](#_Toc470704527)

[3.3 PSNR estimation 7](#_Toc470704528)

[3.4 ML template bank 8](#_Toc470704529)

[3.4.1 RXONLY 8](#_Toc470704530)

[3.4.2 RXTX 8](#_Toc470704531)

[3.4.3 RXFREQ (multi focal) 8](#_Toc470704532)

[3.4.4 RXTXLONGT (Range finder) 9](#_Toc470704533)

[3.5 Power save mode 9](#_Toc470704534)

[3.5.1 Sample bits 9](#_Toc470704535)

[3.5.2 Template bits 9](#_Toc470704536)

[3.6 Debug features 9](#_Toc470704537)

[4 Appendix A: PSNR Estimation 10](#_Toc470704538)

[5 Implmentation Notes 11](#_Toc470704539)

[6 Registers 12](#_Toc470704540)

[7 LUTs 14](#_Toc470704541)

[8 Memories and computations 15](#_Toc470704542)

[9 Test plan 16](#_Toc470704543)

Tables

[Table 1: Revision history 1](#_Toc444077505)

[Table 2: Registers 6](#_Toc444077506)

[Table 3: DCOR memory and bandwidths 7](#_Toc444077507)

Introduction

This block outputs a segment of the correlation between the input CMA data and a given template from the set of predefined templates.

The correlation is performed using two multi resolution steps. First, a coarse, low resolution correlation is a calculated. Second, a fine, accurate correlation is calculated in the vicinity of the maximum value of the coarse correlation.

Let be the sampled code length, where  
*N* is the code length. *N* is even number in [8, 128].  
*FS* is the ratio between the sampling and transmission frequencies (from here onwards: sample ratio). It can be 1:4, 1:8, and 1:16.

The full theoretical correlation output vector is given by:

,

Since calculating the full correlation is both computationally expensive and unnecessary, the data can be sub-sampled in 4 domains:

* Input decimation
* Output decimation
* Signal dynamic range
* Template dynamic range

Assuming (interval between 0 and 1, in our case the range will be quantized to N bits in the interval we can write:

Once the maximum location on the decimated vector is found, a fine correlation using all of the data is calculated over a finer grid adjacent to the peak location at a length of :

Additionally, instead of using a fixed template, using the data from the IR channel the temporal data’s PSNR is evaluated using noise measurements taken while the projector is turned off (i.e. between frames) in the NEST block (within analog sync block). The calculated PSNR value selects the appropriate template to correlate the data with.

The PSNR calculation is performed using a 2d LUT with the IR and ambient noise estimation as inputs.



Figure : DCOR block diagram

Interfaces

The input to DCOR is coming from the

Input

* CMA see Table 3: K\*8bit
  + The CMA is the pixel representation of all of the binary input data that assigned to a pixel. Every CMA has *K* (the length of the sampled code length) elements that store accumulated code, while every element is the normalized summation of the accumulated samples.
* *IR*: 12bit
* *xy\_position*: 12 bit + 11 bit unsigned
* *conf\_dc*: 4 bit
* pxl\_flags: 4 bits
  + [0] scan\_dir: 1 bit, scanline direction
  + [2:1] txrx\_mode: 2 bits, transmission mode
  + [3] eof: 1 bit, end of frame

Output

* *corr\_seg*: A segment of the Fine Correlation of the pixel, see Table 3
* *offset\_index*: The offset of the segment within the fine correlation, 10 bit
* *IR*: 12bit
* *xy\_position*: 12 bit + 11 bit unsigned
* *conf\_dc*: 4 bit
* *conf\_psnr*: 6 bit
* pxl\_flags: 4 bits
  + [0] scan\_dir: 1 bit, scanline direction
  + [2:1] txrx\_mode: 2 bits, transmission mode
  + [3] eof: 1 bit, end of frame

Bypass

When RegsDCORbypass is set the block outputs *IR*, *xy\_position*, *conf\_dc* and *pipe\_flags* from the block input, while *corr\_seg*, *offset\_index*, *conf\_psnr* of the ouput pixels are all outputted as 0.

Detailed description

Coarse Correlation

CMA Decimation

First, the input normalized CMA are decimated so that sampling (TX/RX) ratio is decreased down to 1:2 or 1:4. Effectively down sampling of factors 1:8, 1:4, 1:2, 1:1 are allowed. The decimation sampling ratio is defined by RegsCORRDecRatio, which is the exponent of the decimation factor. The decimation is computed by simple summation of consecutive CMA elements. The number of the resulting *CMADec* element size is increased from 8 bit to 11 bit, while the size of the decimated CMA is reduced to *KDec* = *K* / 2^RegsCORRDecRatio.

We have regsGNRLcodeLength register which gives us the code tamplate size (codeLength\*sampleRate ). We also have DCORcoarseTmplLength register that gives us the size of the decimated tamplate. (codeLength\*sampleRate/dec\_factor).

Correlation

The size of the coarse correlation, and the length coarse template (selected from ML bank), *TDec* is the same as the size of the decimated CMA and are equal to *KDec*. The coarse correlation element is computed as

The number of multiplication operation (11 by 3 bit) is (*KDec*) 2. The number of addition operation on 22 bit values is *KDec\* (KDec-1).* See Table 3 for the exact numbers for specific configurations.

The length of the template *TDec* is *KDec,* andthe size of its values are 3 bit. It is defined by ML Template block, see Section ‎3.3.

*Implementation notes:* Both the coarse and fine (Section ‎3.2) correlation computations require modulo operation. For codes that are not power of 2, direct modulo operation can be computationally expensive. It can be either replaced with comparison operation or it can removed altogether by storing two copies of the template, one after another such that T[i] = T[i%K].

Coarse masking

As the maximum size of the correlation is 256 sample, a vector of 256 logical ( defined in the 8 registers RegsDCORcoarseMasking\_XX ) masks which samples to be calculated. Since the propogation distance varies from pixel to pixel (due to different txrx\_mode flag values), the sample which symbolized zero distance varies between pixels. In the masking vector, the first sample always symbolizes zero distance, and the the vector is shifted in Firmware.

For example: we can have crosstalk between the transmitter & the receiver, and because of that we can get a peak in the correlation at the start of each code, so we can put ‘0’ at the first place of the register, and then the first place of the correlation output will be zero.

The coarse data is saved in 7x32b register such that reg7..reg0 is used for txrx=0, reg15..reg8 for txrx=1, and reg23..reg16 for txrx=2 OR 3.

Due to DCOR HW implementation the registers are stored in reverese order, and circular shifter left by one.

MAX detection

For each of the calculated coarse correlations, the maximal value of is located and its index, *OffsetMax*, 8 bit is outputted to Fine CORR block, and to the block output. The maximal value is computed on the fly and at the coarse correlation output rate. This requires *KDec* comparison operations.

Fine Correlation

The fine correlation is computed only in the vicinity of *OffsetMax.* We call this vicinity as the fine correlation segment. The position *OffsetMax* in the coarse correlation corresponds to position *IMax* = *OffsetMax \** 2^RegsCORRDecRatio within the fine (full) correlation. Thus, the correlation segment is computed for the following interval [*IMax –* RegsDCORfineCorrRange, *IMax +* RegsDCORfineCorrRange]. When RegsDCORfineCorrRange is denoted by *R*, we compute the correlation segment as:

The length of the template *T*is *K*and the size ofits values are 3 bit. It is defined by ML Template block, see Section ‎3.3.

PSNR estimation

This block receives the IR value (12b) from RAST, and the ambient noise estimation (12b) from NEST, and calculates the peak signal to noise ratio. Due to the complexity of the mathematical model, this value is calculated from a 2D TLUT with 16x16 entries of 6 bit, which is a 6 bit index selector for the ML bank. The IR and noise estimation are dynamically reduced to 4 bits using LUT\_IR and LUT\_AMB in order to serve as TLUT entries.

IR and ambient entries are computed as follows:

PSNR estimation is now retriveied from LUTDCORpsnr as follows.

PSNR = RegsDCORpsnr [bitor(bitshiftLeft(T\_Amb\_index, 4), T\_IR\_index)]

In addition, to PSNR estimation, the block outputs IR values without changing them as pass thru. However, when RegsDCORoutNestAsIR is set the IR output are replaces with NEST input values.

ML template bank

The block keeps a bank or coarse and fine templates. Depending on the pixel IR value, ambient noise estimation, txrx mode, and Y location, the block selects a coarse and fine templates to be used in Coarse and Fine CORR blocks. The template bank stores:

* 64 Coarse templates, length: *KDec* (in [16, 256]), template element: 3 bits,
* 64 Fine templates, length: *K* (in [128, 1024]), template element: 3 bits.

Template selection has 4 working modes, set by register RegsDCORtmplMode.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| RXONLY | RXTX | RXFREQ | RXLONGT  (range Finder) |

RXONLY

RegsDCORtmplMode=0

The entire template bank is mapped to 64 PSNR dependant templates. The selected template is based soely on the 6b PSNR value.

RXTX

RegsDCORtmplMode=1

Due to eye safety issues, transmission power when the mirror speed is low is reduced. Since the mirror slow only on the edges, and the Y location is somewhat correlative to the angular location, we can deduce the connection between Y location and transmission power. As different output power values effect the shape of the transmitted signal, the correlation template changes with respect to the transmission power.  
Template bank is accecced by indices defined by the input power (RX/PSNR) and output power (TX/Y location). The Y location of the pixel defines RegsDCORyScalerBits MSB bits of the indices as follows:  
bitshiftRight(RegsDCORyScaler[bitand(127,bitshiftRight(y,RegsDCORyScalerDivExp))], 3-RegsDCORyScalerBits) The LSB bits (6 – RegsDCORyScalerBits) of the indices are defined by MSB of PSNR.

* In rangefinder mode the Yinput is repleaced by the LSB of the x coordiante

RXFREQ (multi focal)

RegsDCORtmplMode=2

As the signal rise time is higly dependant on the transmission frequency, and since different frequencies are used simunteniesly in the same frame, `per-frequency` template is stored in the template bank.  
Template bank is accecced as a 3x16 template LUT between the input power(PSNR) and txrx\_mode.

RXTXLONGT (Range finder)

RegsDCORtmplMode=3

When the required template size is 2048, each two addresses in the template bank are accessed as a single template, yielding 32 template. So, the Template bank is accessed by 5 bit index, 1 MSB bit of y-scaler (y-coordinate), and 4 LSB bits taken from the MSB bits of the PSNR computation.

Template bank is accecced as a 2x16 template LUT between the input power(PSNR) and output power (Y location).

* In rangefinder mode the Yinput is repleaced by the LSB of the x coordiante

Power save mode

Sample bits

The number of normalized CMA sample bits can be reduced from 8 btis to DCORsampleSize by truncating. The truncation is done on the DCOR input and thus affects both the coarse and find correlations. The output segement of fine correlatation is multipled by 2^(8- DCORsampleSize) using bitShift in order to preserve the magnitude of the correlation.

Template bits

If DCORbinTemplate is set, the binary templates for both coarse and fine correlations are used. The binary templates are obtainded by extractint the MSB (or any single bit) of the first coarse/fine tempates from the ML template banks.

Debug features

If RegsDCORoutIRnest is set, the NEST values are sent to IR output. Note that template selection is performed on the original input IR values. If RegsDCORoutIRcma is set of the CMA bins specified by RegsDCORoutIRcmaBin is sent to IR output. If RegsDCORoutIRcmaBin is larger than K, the K-th CMA bin is taken. If both RegsDCORoutIRnest and RegsDCORoutIRcma are set, the NEST values are sent to IR output.

IR as CMA

This debug feature enables output a specific index from the CMA into the IR.  
In this mode (enabled using DCORoutIRcma) the 8bits of the CMA are copied to the 8lsb of the IR. The CMA bin is selected using the register DCORoutIRcmaBin.   
Due to hardware implementation, the CMA is divided into 84 segments. Using the selected index, the CMA segment and segment offset are calculated:

DCORoutIRcmaIndex[7:0] = DCORoutIRcmaBin/84 – segment num

DCORoutIRcmaIndex[15:8] = DCORoutIRcmaBin%84 – segment offset

Appendix A: PSNR Estimation

The purpose of this block is to calculate the PSNR:

* - illumination power.
* - noise.

Using known measurements:

* non-transmission measurement containing noise (ambient) data only- NEST.
* transmission measurement containing both signal and noise- IR channel.
* transmission when laser is always on – constant, capture during calibration, low amb, high SNR.

The estimation model assumes that the transmitted code undergoes multiplication by the illumination power and offset by the environment ambient light. Then, the APD noise (which is a poisson distributed noise that depends in the input to the APD) is added. The received signal is then transformed from optical power to voltage by the TIA using the conversion factor (Assuming is constant over time).

Finally, the system noise is added:

Where is the ambient noise standard deviation and is the electrical noise standard deviation. The std of the APD noise is the same as its mean because it’s a poisson noise.

Next, the signal undergoes through the linear time invariant system, which includes HPF (DC part is removed), absolute value of the signal, and LPF is preformed (avareging), resulting:

Let us define the noise for each of the cases:

When

Using the above model, a non-transmitting region measurement yields (:

Assuming the noise variance is at least an order of magnitude smaller than , we can safly write:

Sine we know the expectancy of a folded Gaussian distribution (because of the absolut value the Gaussian is folded):

Where is the normal cumulative distribution function:

Thus,

Receiving a signal when transmission is always on (cw = continues wave) signal over maximal reflectivity target, yields the following (:

And estimation :

Since the transmitting region measurement holds folded normal distribution statistics, the above equation boils to :

Assuming the noise variance is at least an order of magnitude smaller than , we can safly write:

Thus:

In practice, since extracting from the above equation is tricky, a look-up-table is built in order to create.  
Once is obtained from and is obtained from , the PSNR is equal to:

Dynamic ranges

Recall that:

Moving to the quntized domain:

Data is then quntized to 12b using adc:

Assuming :

Implmentation Notes

27-Dec-2016: Correlation block inside the DCOR was implemented as

Instead of

In both coarse and fine.

A fixe was applied by outputting the reverese correlation and reverse the max\_index

Registers

Table : Registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Size** | **Default** | **Range** | **Special values/ description** |
| **General** |  |  |  |  |
| RegsGNRLImgHsize | 12 | 640 | 1-1280 | Horizontal resolution |
| RegsGNRLimgVsize | 12 | 480 | 1-960 | Vertical resolution |
| RegsGNRLCodeLength | 8 | 26 | 8 .. 128 | The code length (even numbers only) |
| RegsGNRLSampleRate | 5 | 16 | 4, 8, 16 | Sampling rate |
| RegsGNRLtmplLength | 16 | 416 | 0-2^16-1 |  |
| **DCOR** |  |  |  |  |
| RegsDCORbypass | 1 | 0 | 0,1 | Block bypass mode |
| RegsDCORdecRatio | 2 | 3 | 0,1,2,3 | The exponent of the decimation ratio |
| RegsDCORfineCorrRange | 6 | 16 | 1-16 | Fine correlation range is obtained via  1+2\* RegsDCORfineCorrRange.  Currently, value is fixed at 16 |
| RegsDCORirStartLUT | 12 | 0 | 0..4031 | Starting value of the ML LUT mapping for IR |
| RegsDCORambStartLUT | 12 | 0 | 0..4031 | Starting value of the ML LUT mapping of ambient noise |
| RegsDCORirLUTExp | 4 | 9 | 5..9 | The ML fine LUT mapping exponent for IR |
| RegsDCORambLUTExp | 4 | 9 | 5..9 | The ML LUT mapping exponent for ambient noise |
| DCORbinTemplate | 1 | 0 | 0, 1 | Binary template is used |
| RegsDCORtmplMode | 2 | 0 | 0..2 | Template selection mode |
| RegsDCORyScalerDivExp | 3 | 2 | 0..6 | yScaler division by bitshift |
| RegsDCORyScalerBits | 2 | 3 | 1..3 | The number of bits of yScaler in RXTX template mode |
| RegsDCORcoarseTmplLength | 16 | 52 | 0-2^16-1 |  |
| RegsDCORcoarseMasking\_000..007 | 32 | 11…11 | 0-2^32-1 |  |
| DCORoutIRnest | 1 | 0 | 0, 1 | Output IR with the NEST values |
| DCORoutIRcma | 1 | 0 | 0,1 | Output IR with the one of the CMA bins |
| DCORoutIRcmaIndex |  |  |  |  |
| DCORoutIRcmaBin | 10 | 0 | 0..1023 | The CMA bin to output as IR |

LUTs

Table : LUTs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Entries | Entry size | Fixed | Description |
| RegsDCORpsnr | 2^8 | 6 | no | PSNR estimation |
| RegsDCORirMap | 2^6 | 4 | no | IR mapping of PSNR indices |
| RegsDCORambMap | 2^6 | 4 | no | Ambient mapping of PSNR indices |
| RegsDCORyScaler | 2^7 | 3 | no | Y scaler for PSNR |

Memories and computations

The DCOR block does not require any line buffers. All the operations are performed on a single pixel. The only buffers are pixel/CMA size pipeline buffers.



Table : DCOR memory and computations

Test plan