The APEX-CV Pro Library

UG-10328-02-14

Copyright

Disclaimer

NXP Semiconductors assumes no responsibility for the accuracy or completeness of the information presented which is subject to change without notice. In no event will NXP Semiconductors be liable for any direct, indirect, special, incidental or consequential damages, including lost profits, lost business or lost data, resulting from the use of or reliance upon the information, whether or not NXP Semiconductors has been advised of the possibility of such damages.

Mention of non-NXP Semiconductors products or services is for information purposes only and constitutes neither an endorsement nor a recommendation.

Uncontrolled Copy

The master of this document is stored on NXP Semiconductors' document management system. Viewing of the master electronically ensures access to the current issue. Any hardcopies are considered uncontrolled copies.

Version	Details of Change	Author	Date
01	Initial Revision	J. Lalonde, C. Garrard	August 1, 2014
02	Update to add Image Pyramid, Interpolate, Accumulate, Accumulate Squared and Gaussian.	A. Saechao, J. Cairns	October 1, 2014
03	Update Harris Corner Detection documentation	A. Saechao	October 31, 2014
04	APEX-CV base update: abs, clz	G. Billig	January 29, 2015
05	APEX-CV pro update: BRIEF, Block Matching, Harris, Hough	A. Saechao	July 9, 2015
06	updated the document to match release 1.8.4, APEX-CV high level description	S. Ashby, S. Francois	August 20, 2015
07	APEX-CV pro update: Canny, HOG, Affine, GFTT, Remap, Resize	A. Saechao, .Garrard	November 1, 2015
08	APEX-CV pro update: Fast, Harris Corners, LK-TrackerOpticalFlow, LKPyramidOptical, PyramidMultiCreation	N. Zhu,	May 13, 2016
09	APEX-CV pro update: TMO, LBP, Harris/GFTT new interface	M. Mai, N.Zhu	September 27, 2016
10	APEX-CV pro update: Laplacian Pyramid, ORB and AggCF	A. Grigore, M. Petre D. Zheng	March 09, 2017
11	APEX-CV pro update: update documentation to RTM 1.0 content	N. Zhu, S. Francois	July 11, 2017
12	APEX-CV pro update: Hog, Orb, GFTT, Image Pyramid, Canny, Resize, Affine	K. Pham	March 06, 2018
13	APEX-CV pro update: Resize, PyramidMultiCreation, Orb, Hog	K. Pham	August 11, 2018
14	Umat replace by SUmat, APEX-CV pro update: Remap, HOG	K. Pham	Dec 06, 2018

Contents

Chapter 1

APEX-CV Pro Library

The APEX-CV Pro library provides high-level functionality for developers to design their own computer vision applications while taking advantage of NXP's massively parallel APEX-CV architecture. The library contains the following modules:

1. APEX-CV Base Library - Basic image processing functionality.
2. APEX-CV Feature Detection Library
    - Block Matching
    - Binary Robust Independent Elementary Features
    - Oriented Fast and Rotated BRIEF
    - Canny Edge Detector
    - GFTT/HARRIS Corner Detector
    - HOG Object Detector
    - Aggregated Channel Feature Based Pedestrian Detector
    - Hough Line Detector
    - FAST9 corner detection
3. APEX-CV Image Pyramid Library
    - Gaussian Image Pyramid
    - Multi-scale Gaussian Image Pyramid
    - Laplacian Image Pyramid
4. APEX-CV Image Transform Library
    - Affine Transformation
    - Histogram Equalization
    - Image Remap
    - Image Resize
    - Tone Mapping Operation
5. APEX-CV Feature Tracking Library
    - Single-Scale Lucas-Kanade Optical Flow
    - Multi-Scale Lucas-Kanade Optical Flow

Chapter 2

APEX-CV Base Library

The APEX-CV Base library provides basic functionality for developers to design their own imaging-based applications while taking advantage of NXP's massively parallel APEX architecture. Currently various arithmetic operations, color conversions and image filters are provided as well as image calculations such as histogram and integral image as listed below.

- Arithmetic Operations:
  - Absolute value
  - Absolute difference
  - Accumulate
  - Accumulate squared
  - Accumulate weighted
  - Addition
  - Bitwise AND, NOT, OR, XOR
  - Count Leading Zeros
  - Magnitude
  - Max
  - Min
  - Pixel-wise Multiplication
  - Gradient Phase Computation
  - Subtraction
  - Table Lookup
  - Thresholding (binary)
  - Thresholding (range)
- Interpolation Operations:
  - Linear Grayscale
  - Bilinear Grayscale
  - Bicubic Grayscale
- Color Conversion and Channel Manipulation Operations:
  - Color conversion and color rotation

  - Color conversion and color rotation (optimized)
  - Convert bit depth
  - Extract Channel
  - Insert Channel
  - Split Channel
  - Merge Channel

- Image Filters Operations:
  - Bilateral filter
  - Box filter
  - Box filter (optimized)
  - Census filter
  - Convolve filter
  - Convolve filter (optimized)
  - Derivative X filter (optimized)
  - Derivative Y filter (optimized)
  - Dilate filter
  - Erode filter
  - Gaussian filter
  - Median filter
  - Prewitt X filter
  - Prewitt Y filter
  - Saturate filter (optimized)
  - Scharr Filter
  - Scharr Filter X
  - Scharr Filter XY
  - Scharr Filter Y
  - Separable filter (optimized)
  - Sobel filter
  - Sobel filter (optimized)
  - Sobel X filter
  - Sobel X filter (optimized)
  - Sobel Y filter
  - Sobel Y filter (optimized)
  - Sobel XY filter

- Histogram Operations:
  - Histogram
  - Mean
  - Standard deviation

- Integral Image

Chapter 3

Block Matching

The Block Matching algorithm is used to locate matching macroblocks between two images. This is done in APEX-CV by using the Sum of Absolute Differences approach. See apexcv::Blockmatching.

To perform the search, a number of search points and locations need to be specified. The the algorithm will then search within a 28x28 region of pixels, the search window, around those points for a matching macroblock. This is done by calculating the Sum of Absolute Differences (SAD) score for all 16x16 region of pixels within the search window. A block is considered to be matching if the SAD score is below a user specified maximum SAD threshold.

Chapter 4

Binary Robust Independent Elementary Features

BRIEF is a fast method for the feature descriptor calculation. It finds the binary strings directly without calculating descriptors in floating point numbers. BRIEF takes smoothed image patch and selects a set of nd(x,y) location pairs in Gaussian distribution pattern. Then pixel intensity comparisons are done for each pair, and the results are stored in binary. This is applied for all the nd location pairs to get a nd-dimensional bitstring.

APEX-CV BRIEF is implemented based on OpenCV BRIEF implementation. First, the sum of 9x9 pixel patch is calculated. To reduce the computational cost, integral image is used. Then, the comparison of pixel intensity between selected pair is performed. This comparison pairs are selected from 48x48 regions around a keypoint, with Gaussian distribution pattern. The result of comparison is stored as binary string. For example, let one location pair be p and q. If I(p) < I(q), then its result is 1, else it is 0. The size of descriptor is either 16 (default), 32, or 64 bytes.

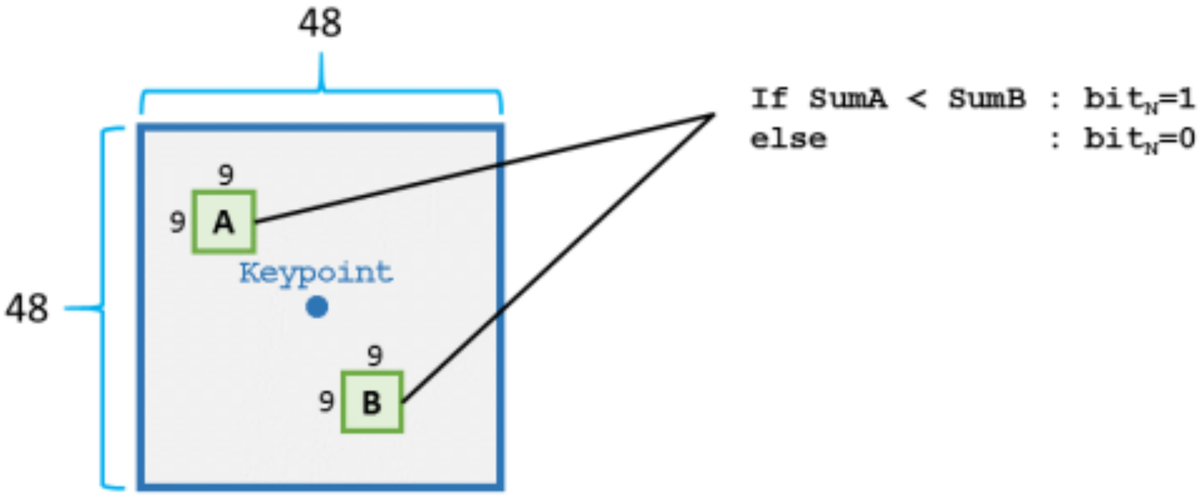


Figure 4.1: APEX-CV BRIEF

Chapter 5

Oriented Fast and Rotated BRIEF

5.1 Overview

Orb is basically a feature matching algorithm that is very similar to SIFT/SURF but much more simple to compute. Descriptors are in a binary form and an Hamming distance can be used to find the differences between two descriptors. The output length of a descriptor can be 16, 32 or 64 Bytes. The added value of this implementation is the fact that the pattern used to generate the descriptors needs to be given by the user.

5.2 Implementation details

The processing is split into 3 stages:

1st stage: FAST9 + HARRIS CORNER SCORE + ARM sorting + ICO
Orb algorithm uses corners as the center of a rectangular region of an image and the applies a BRIEF over this region. The first part of the processing finds the corners using FAST9 then runs a Harris Coners Score to calculate the cornerness of each feature. The top N most powerfull corners are selected based on the harris score and for each region of interest the ICO calculates the patch orientation. ICO outputs a value from [0, 255] that is mapped to [0, 360] degrees. The function that implements stage 1 is called: detect()

2nd stage: User defined smoothing, this step is very important to have a decent detection rate. The usual filters used to blur the image prior to brief are: box 7x7, gaussian blur 5x5, 7x7 and 9x9. The demo uses a 5x5 gaussian blur.

3rd stage: rBRIEF - to make BRIEF invariant to rotations we steer the image patch according to the information calculated at step 1. BRIEF works by comparing pairs of points in a greyscale image, the points are randomly select from a pattern buffer, here rBRIEF applies a rotation matrix on the pattern buffer to have a rotation of the coordinate pairs not the image. The angle is calculated at step 1. The function that implements stage 3 is called: compute()

The most time consuming stages are the 1st and 3rd because they are directly dependent on the image size, the rest are closely related to the number of keypoints that the user needs. APEX-CV ORB is implemented based on OpenCV BRIEF implementation. A full description of the algorithm can be found by searching for "ORB: an efficient alternative to SIFT or SURF"

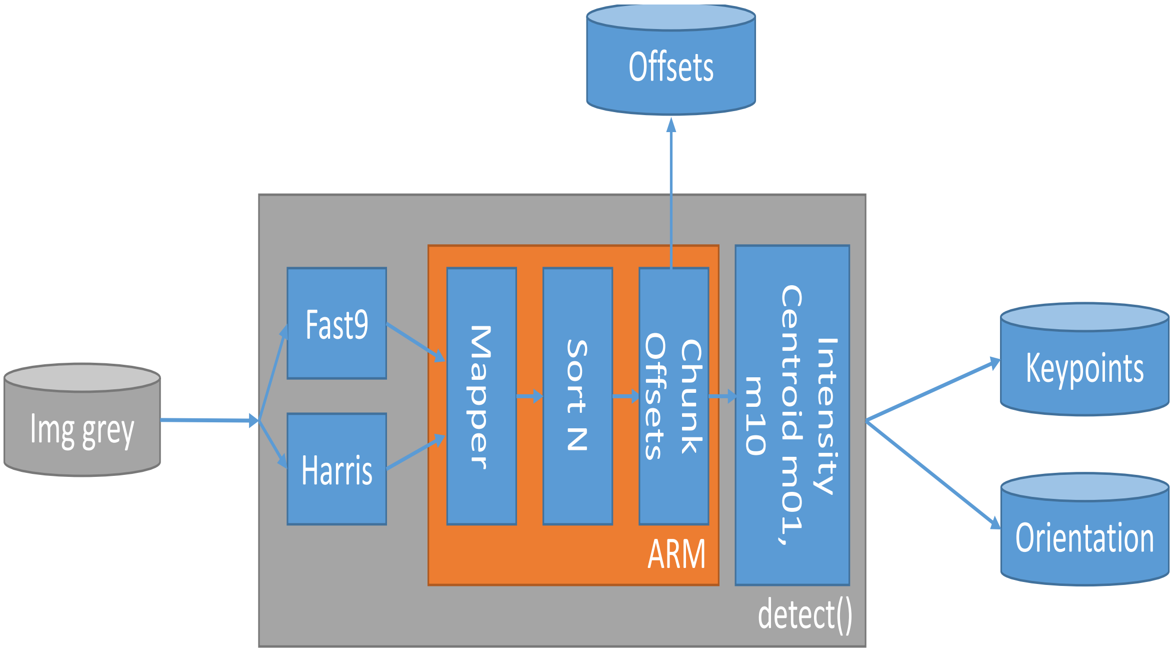


Figure 5.1: Keypoint detection process

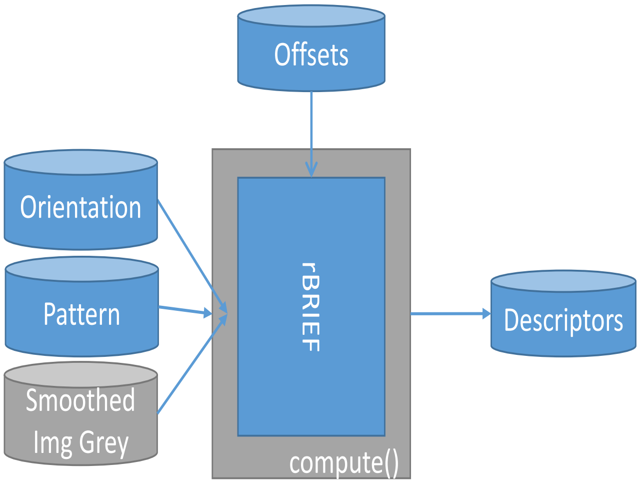


Figure 5.2: Steered BRIEF

Chapter 6

Canny Edge Detector

6.1 Overview

The APEX-CV Canny Edge Detector follows the standard Canny edge detection algorithm. It calculates the X and Y gradients using 3x3 Sobel filters, computes the magnitudes, and performs non-maxima suppression, and double thresholding to determine good edge points.

6.2 Implementation details

With the Canny edge detection algorithm, it promotes potential edge points into edge points if they are a neighbour with a definite edge point. This causes some problems when parallelizing the algorithm due to the processing methodology of breaking the image into smaller macro blocks. The information from one processing block must be propagated to other processing blocks. This means that in order to get a complete edge mapping of the image, the information from all blocks must be propagated to every single block, which is just not feasible. Therefore, the approach taken with the AP↩ EX-CV Canny detector is to propogate the information to neighbouring blocks only. The amount of times this progation occurs can be controlled by the user. More propagation iterations will allow increase the range in which a definite edge point has an affect on the edge mapping.

Chapter 7

GFTT/HARRIS Corner Detector

7.1 Overview

The algorithm is a fixed point implementation of Harris Corner [4] and GFTT corner [7] and is similar to the function `goodFeaturesToTrack` in OpenCV.
Please refer to apexcv::GFTTCorners interface for API information.

apexcv::GFTTCorners interface combines Harris and GFTT into the same interface. Toggle between Harris and GFTT can be done using **useHarrisDetector** parameter.

The corner detector produces a list of corners of an image. It is done by:

1. Computes a 16-bit corner response image from an 8-bit grayscale image using Harris Score Formula [4] or Minimum Eigen Value (GFTT) [7].

2. Performs non-maxima supression in a 3x3 neighborhood.

3. Removes any corners that are below threshold. GFTT Threshold = QualityLevel $*$ MaxEigenValue. Harris Threshold is specified by users.

4. The remaining corners are sorted by the strength in the descending order.

5. Minimum distance filtering will be applied. A strongest corners within minimum distance radius will be kept. Other corners will be discarded.

7.2 Harris Corner Detector

For each pixel I(x,y), the Harris corner response is calculated:

1. Find gradient in x and y direction using Sobel in a 3x3 window:

   $G_x = Sobel_x(3x3, I(x,y))$

$G_y = Sobel_y(3x3, I(x,y))$

2. Calculate Trace and Det:

   $trace = G_x^2 + G_y^2$
   $det = G_x^2 * G_y^2 - (G_x * G_y)^2$

3. The Harris corner response is then calculated from Trace and Det:

   $dst(x,y) = det(x,y) - k * trace(x,y)^2$ where k is Harris Corner Coefficient

7.3 Good Features To Track

For each pixel I(x,y), the Minimum Eigen Value is calculated as corner response:

1. Find gradient in x and y direction using Sobel in a 3x3 window:

   $G_x = Sobel_x(3x3, I(x,y))$
   $G_y = Sobel_y(3x3, I(x,y))$

2. The Minimum Eigen Value can be approximated using:

   $dst(x,y) = Min(\lambda_1, \lambda_2) \approx \frac{G_x^2}{2}\frac{G_y^2}{2} - \sqrt{(\frac{G_x^2}{2} - \frac{G_y^2}{2})^2 + (G_xG_y)^2}$

7.4 Limitation in this release

1. Maximum Image Width is 1280 pixel.

2. Sobel size is fixed to 3x3, NMS size is fixed to 5x5, Box Size default value is 7x7. Only the box size can be changed with apexcv::GFTTCorners::SetShiftValue, acceptable box size dimesions are 3x3, 5x5 or 7x7.

3. Maximum corners before sorting and extraction is 4096 in total:

   APEX only keeps the first 4096 corners and discards all corners after 4096 limitation is reached. Hence, some of the strong corners might be discarded. To limit the impact, a good threshold value can be applied to filter out weak corners and save space for strong corners.

4. Maximum corners per chunk is 20:

   Similar to (3), each chunk can only keeps the first 20 corners and discards the rest. To avoid or limit having strong corners discarded a good threshold should be applied to keep the number of corners low per chunk.

5. Covariance Scale factor is set by default at 6. To change the value of the covariance scale factor use apexcv::G↩FTTCorners::SetShiftValue. A good scale value balances saturation (value too low) and precision loss (value too high).

Chapter 8

HOG Object Detector

The APEX-CV HOG Object Detector detects objects using `Histograms of Oriented Gradients` (HOGs). The algorithm is a fixed point implementation of [1] using a simpler HOG feature. It is similar to the function `HOG↩ Descriptor::detect` in OpenCV.

The detector takes an 8-bit grayscale image and detects objects every 4x4 pixels. That is, for each pixel on a 4x4 lattice of the image, the HOG descriptor is computed. The hog descriptor parameters are as follows: 8x8 cell, 8x8 block, 64x128 window and 8 bins. The trained linear SVM classifier is applied to each descriptor to produce a 16-bit *SVM score*. So for an input image of size $w \times h$, the detector returns a 16-bit score image of size $w/4 \times h/4$.

Chapter 9

Aggregated Channel Feature Based Pedestrian Detector

The apexcv::AggCFDetector detects pedestrian using `aggregated channel feature` (ACF). It is a fixed point implementation.

The aggCF detector takes a 24 bit RGB image and calculate aggregated channel feature pyramid which is divided into octaves. Each octave includes real scale and approximation scales with different scale sizes. Each scale includes L, U, V, magnitude and histogram of gradients (HOG) as channel features. HOG includes 6 gradient bins and can be calculated using both bi-linear and tri-linear interpolation based on pre-trained detector model parameter. The structure of pyramid including number of octaves/scales are determined by pre-trained detector model.

Once feature pyramid is calculated, aggCF detector perform pedestrian detection based on decision trees in pre-trained detector model, apply non maximal suppression (NMS) on bounding boxes of detected pedestrian. The detection is performed using sliding window approach, the size/stride of search window is also defined in pre-trained detector model.

Chapter 10

Hough Line Detector

10.1 Overview

The APEX-CV Hough Line Detector detects lines from an 8-bit grayscale image. The algorithm is based on [3] and is similar to the function `HoughLines` in OpenCV. A good overview of the Hough transform can be found on `Wikepedia`

10.2 Line Representation

The detected lines are expressed in polar coordinates (ρ, θ), where ρ is the nearest distance of the line to the image center (c_x, c_y) and θ is angle of the normal to the line. This is shown below.

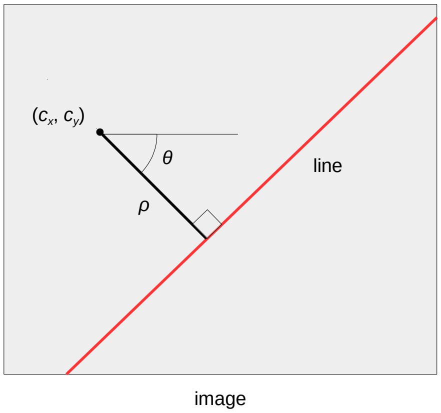


Figure 10.1: The geometric interpretation of the Hough line coordinates (rho, theta).

10.3 Supported Image Sizes

Currently four image widths are supported. Images passed to the detector must be exactly one of those widths. The image height must be a multiple of four. The maximum image height is determined by the image width and the available APU memory for the Hough accumulator. The supported widths and the corresponding maximum height are shown in the following table.

Supported Width	Maximum Height
192	1588
192	1568
640	1468
1280	960

Figure 10.2: Supported image sizes for the Hough Line Detector.

10.4 Specifying Angles for Detection

The APEX-CV Hough Line Detector is designed for maximum flexibility. The simplest way to specify the detection angles is by specifying the number of angles only. In this case the detector divides the range of angles $[0,\pi[$ evenly by the number of angles specified. A maximum of 256 angles can be specified. For example, calling

```
apexcv::HoughLineDetector hough;
vsdk::UMat image;
// ...setup image
hough.initialize(image);
hough.setTheta(180);
```

gives a detector with a line resolution of 1° (since $1^\circ = pi/180$ radians).

For more flexibility, the angle starting value and angle resolution can be specified. Angles must be expressed in radians. For those more familiar with degrees, the conversion factors apexcv::HoughLineDetector::rad2deg and apexcv::HoughLineDetector::deg2rad are supplied to convert from radians to degrees and degrees to radians, respectively. For example

```
using namespace apex;
HoughLineDetector hough;
vsdk::UMat image;
// ...setup image
hough.initialize(image);
hough.setTheta(32, 15*HoughLineDetector::deg2rad, 5*HoughLineDetector::deg2rad);
```

gives a detector sensitive to 32 lines with angles starting from 15° and each separated by 5°.

For the most flexibility, an arbitrary set of up to 256 angles my be specified. These angles are expressed in radians and are passed to the detector as an array of floats. For example,

```
const int thetaCount = 10;
float theta[thetaCount] = {-0.02f, -0.01f, 0.f, 0.01f, 0.02f, 1.55f, 1.56f, 1.57f, 1.58f, 1.59f};
apexcv::HoughLineDetector hough;
vsdk::UMat image;
// ...setup image
hough.initialize(image);
hough.setTheta(thetaCount, theta);
```

gives a detector sensitive only to vertical and horizontal lines (lines about 0 and $pi/2$ radians).

10.5 Non-Maxima Suppression

By default, non-maxima suppression (NMS) is performed on the Hough accumulator. NMS is simply a comparison of neighbouring accumulator values in both the rho and theta directions; If the accumulator value is greater than the previous value and greater or equal to the next value (in both directions), then that line is considered for detection (the remaining condition being that the accumulator value be above the specified threshold).

Control over NMS is provided by the flag apexcv::HoughLineDetector::NonMaxSupFlag. The possible states are

1. NMS_NONE: No NMS is performed.
2. NMS_RHO: NMS is performed in rho direction only.
3. NMS_THETA: NMS is performed in theta direction only.
4. NMS_RHO|NMS_THETA: NMS is performed in both directions (default state).

It is recommended to always use NMS in the rho direction (as the rho resolution is only 1 pixel). However, if a coarse angle resolution is used (i.e. the angle step is large), then NMS between angles should be turned off. For example

```
using namespace apex;
HoughLineDetector hough;
vsdk::UMat image;
// ...setup image
hough.initialize(image);
hough.setTheta(32, 15*HoughLineDetector::deg2rad, 5*HoughLineDetector::deg2rad, HoughLineDetector::NMS_RHO)
      ;
```

disables NMS between angles, which is appropriate since the angle resolution (5°) is coarse.

Chapter 11

FAST9 corner detection

11.1 FAST (Features from accelerated segment test)

APEX-CV FAST algorithm implements `FAST corner detection` [6] to extract feature points from 8-bit grayscale source image. When "nms" flag is enabled, FAST corner score will be computed to perform non-maxima suppresion post processing step.

Chapter 12

Gaussian Image Pyramid

There are two common kinds of image pyramids: Gaussian pyramids and Laplacian pyramids. Here, we present the APEX-CV Image Pyramid, an implementation of Gaussian image pyramid creation.

To upsample an image, the source image is upsized 2x in each dimension, with the new even-numbered rows and columns filled with zeros. Then, the expanded image is convolved with the 5x5 Gaussian kernel below. As a result, the area is increased to exactly four times the area of the source image.

$$\frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Figure 12.1: The Gaussian Matrix

To downsample an image, first the source image is convolved with the above 5x5 Gaussian kernel (divided by 4), then every even-numbered row and column is removed, 2x downsampling in each dimension. As a result, the area is reduced to exactly one-quarter the area of the source image.

Chapter 13

Multi-scale Gaussian Image Pyramid

APEX-CV Pyramid Multi supports 4 scales downsampling of an source image through one APEXCV call.

The source image is convolved with the Gaussian filter in the same way as image pyramid kernel then down sampled. Each APEXCV call will produce [1/2, 1/4, 1/8, 1/16] 4 scales pyramids.

Current version only support down sampling.

Chapter 14

Laplacian Image Pyramid

The APEX-CV Laplacian Image Pyramid is an implementation of the Laplacian image pyramid creation.

The Laplacian Image Pyramid is generated with the help of a Gaussian Image Pyramid. For each of the pyramid levels, the laplacian image is obtained by convoluting the Gaussian Image Pyramid level with a 5x5 Gaussian kernel and subtracting it from the input image. There is also an final output image whis can be used together with the Laplacian Image Pyramid to reconstruct the original image. This image is the convolution result of the last pyramid level.

Chapter 15

Affine Transformation

The APEX-CV Affine Transform performs an affine transformation on an 8-bit image. The algorithm is similar to the function `warpAffine in OpenCV`.

The usual way to represent affine transformation is by using a 3x3 matrix.

$$\begin{pmatrix} Dx \\ Dy \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Sx \\ Sy \\ 1 \end{pmatrix}$$

An inverse matrix for a given matrix is first calculated. Then, for each destination pixel, corresponding x and y coordinates in source image are determined using formulas below.

Given a transform matrix:

$$M = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix}$$

Inverse matrix of M is calculated as:

$$M^{-1} = \begin{pmatrix} e/det & -b/det & (bf-ce)/det \\ -d/det & a/det & (cd-af)/det \\ 0 & 0 & 1 \end{pmatrix}$$

where

$$det = (ae - bd)$$

Therefore, source coordinates Sx and Sy are calculated as:

$$\begin{pmatrix} Sx \\ Sy \\ 1 \end{pmatrix} = \begin{pmatrix} e/det & -b/det & (bf-ce)/det \\ -d/det & a/det & (cd-af)/det \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Dx \\ Dy \\ 1 \end{pmatrix}$$

Each pixel value is then calculated by bilinear interpolation of neighboring 2x2 pixels at the coordinate determined above.

Chapter 16

Image Remap

The APEX-CV Remap maps one image to another from a floating point lookup table. The algorithm is similar to the function `remap in OpenCV`.

The current implementation of apexcv::Remap supports image sizes that are an even multiple of the number of CUs (64). So supported images are 128, 256, 384, 512, ... , 2048.

The current version of remap supports bilinear interpolation only.

Chapter 17

Histogram Equalization

The APEX-CV Histogram Equalization transforms the pixel values of an image so that the resulting histogram uses the full range of grey values equally. The algorithm is similar to the function `Histogram Equalization` in OpenCV.

The current implementation of apexcv::HistogramEqualization supports image sizes which have a width multiple of 4.

Chapter 18

Image Resize

The Image Resize performs a vertical and horizontal resize on the input image, according to the required size of the output image. Note that vertical and horizontal scale factors (and directions) are independent. It is similar to the function `resize()` in OpenCV.

Currently, all even image dimensions are supported from 64 to 1024 for both input and output.

The current version of resize supports bicubic interpolation only.

Chapter 19

Tone Mapping Operation

The Tone Mapping Operation generates an LDR image from an HDR image by compressing the HDR's dynamic range. The APEX implemetation of TMO is a fixed-point implementation of tone mapping operation as described in [2]. The implementation uses integer data and fixed-point arithmetic for all calculations in TMO to reduce computational and memory cost.
Please refer to the apexcv::tmo API for more detail.

Chapter 20

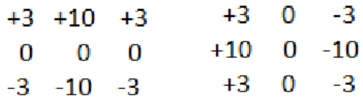
Single-Scale Lucas-Kanade Optical Flow

The APEX-CV LKTracker Optical Flow implements single scale Lucas-Kanade Sparse Optical Flow [5]. The algorithm is similar with `OpenCV.`

Currently, the size of the window of the box filter can only be 7x7. The 'iteration' parameter, has a default value of 10, and it can be changed to adjust the trade-off between tracking accuracy and processing time. Maximum motion vector support up to +/-6. Points with out of range displacement will be marked as untracked as stated in next paragraph

Algorithm will take in previous frame, frame[t-1], in 8 bits greyscale format; previous points which are X/Y coordinates in signed Q23.8 format and next frame, frame[t] in 8 bits greyscale format. It outputs next points as X/Y coordinates in signed Q23.8 format along with strength and reserve fields. Strength field represents the sum of absolute greyscale difference between input 7x7 window and output tracked 7x7 window. Reserve field represents whether displacement vector is out of range (1: within range, valid tracking; 0: out of range, invalid tracking)

Image gradient Dx and Dy is calculated by Scharr filter:



$$\begin{array}{ccc} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{array} \qquad \begin{array}{ccc} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{array}$$

Input and output X/Y coordinates have 8 bits sub-pixel accuracy. Bilinear interpolation will be applied when handling sub-pixel greyscale or gradient.

Chapter 21

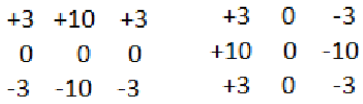
Multi-Scale Lucas-Kanade Optical Flow

The APEX-CV LKPyramid Optical Flow implements multiple scale Lucas-Kanade Sparse Optical Flow. The algorithm is similar with `OpenCV.`

Currently, the size of the window of the box filter can only be 7x7. The 'iteration' parameter, has a default value of 10, and it can be changed to adjust the trade-off between tracking accuracy and processing time. Maximum motion vector support up to +/-6. Points with out of range displacement will be marked as untracked as stated in next paragraph

Algorithm will take in previous frame, frame[t-1], in 8 bits greyscale format; previous points which are X/Y coordinates in signed Q23.8 format and next frame, frame[t] in 8 bits greyscale format. It outputs next points as X/Y coordinates in signed Q23.8 format along with strength and reserve fields. Strength field represents the sum of absolute greyscale difference between input 7x7 window and output tracked 7x7 window. Reserve field represents whether displacement vector is out of range (1: within range, valid tracking; 0: out of range, invalid tracking). Algorithm will create and loop through all pyramid levels which can be up to 5: [1/16, 1/8, 1/4, 1/2] plus the original scale. Number of pyramid levels can be configurable.

Image gradient Dx and Dy is calculated by Scharr filter:



$$\begin{matrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{matrix} \qquad \begin{matrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{matrix}$$

Input and output X/Y coordinates have 8 bits sub-pixel accuracy. Bilinear interpolation will be applied when handling sub-pixel greyscale or gradient.

Chapter 22

LBP Face Recognition

The APEX LBP face recognition is similar to `Local Binary Patterns Histograms` in OpenCV.

The algorithm will run APEX-LBP train process to extract an LBP descriptor for each grid cell. The process will take as input an unsigned 8 bit image and output an 8 bit descriptor for each grid cell. Then the APEX-LBP predict process will be executed to compare the test descriptors with the model descriptors in order to find the closest descriptor. This predict process will output an unsigned 16 bit value representing the closest ID and a set of signed 32 bit distances.

Chapter 23

Class Index

23.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 24

Class Documentation

24.1 apexcv::Abs Class Reference

Absolute value, *aDst*(x,y) = abs(*aSrc*(x,y))

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.1.1 Detailed Description

Absolute value, *aDst*(x,y) = abs(*aSrc*(x,y))

Object of this class computes the absolute value of every pixel.
Output dimensions are same as input.
Supported input type: VSDK_CV_8SC1, output type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels

24.1.2 Member Function Documentation

24.1.2.1 APEXCV_LIB_RESULT apexcv::Abs::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.1.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.1.2.3 APEXCV_LIB_RESULT apexcv::Abs::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.1.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

in	*a↩ ApexId*	ID of the APEX core used for performing the processing (0 or 1).

24.2 apexcv::AbsDiff Class Reference

Absolute difference.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.2.1 Detailed Description

Absolute difference.

Object of this class computes the absolute difference pixel for every pixel.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1
Supported input type: VSDK_CV_16SC1, output type: VSDK_CV_16SC1
Supported width: 128 to 2048 pixels

24.2.2 Member Function Documentation

24.2.2.1 APEXCV_LIB_RESULT apexcv::AbsDiff::Initialize (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1 or VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1 or VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).

24.2.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.2.2.3 APEXCV_LIB_RESULT apexcv::AbsDiff::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1 or VSDK_CV_16SC1).
`in`	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1 or VSDK_CV_16SC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).

24.2.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.3 apexcv::Accumulate Class Reference

Accumulate.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

Select the APEX Core.

- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.3.1 Detailed Description

Accumulate.

Object of this class accumulates *aSrc* into *aDst*.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1
Supported width: 128 to 2048 pixels

24.3.2 Member Function Documentation

24.3.2.1 APEXCV_LIB_RESULT apexcv::Accumulate::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.3.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.3.2.3 APEXCV_LIB_RESULT apexcv::Accumulate::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.3.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.4 apexcv::AccumulateSquared Class Reference

Accumulate Squared.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, const uint8_t acScale, vsdk::SUMat &aDst)
  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)
  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetScale (const uint8_t aScale)
  Set Scale.
- APEXCV_LIB_RESULT GetScale (uint8_t &aScale)
  Get Scale.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)
  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()
  Start processing and return when done.

24.4.1 Detailed Description

Accumulate Squared.

Object of this class accumulates a squared value from *aSrc* to _aDst.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1
Supported width: 128 to 2048 pixels

24.4.2 Member Function Documentation

24.4.2.1 APEXCV_LIB_RESULT apexcv::AccumulateSquared::GetScale (uint8_t & *aScale*)

Get Scale.

This function allows to read the scale value.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aScale*	Scale amount. Right shift the square of aSrc by aScale (0 <= aScale <= 15)

24.4.2.2 APEXCV_LIB_RESULT apexcv::AccumulateSquared::Initialize (vsdk::SUMat & *aSrc,* const uint8_t *acScale,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in`	*acScale*	Scale amount. Right shift the square of aSrc by aScale (0 <= aScale <= 15)
`in,out`	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.4.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.4.2.4 APEXCV_LIB_RESULT apexcv::AccumulateSquared::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.4.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ *ApexId*	ID of the APEX core used for performing the processing (0 or 1).

24.4.2.6 APEXCV_LIB_RESULT apexcv::AccumulateSquared::SetScale (const uint8_t *aScale*)

Set Scale.

This function allows to change the scale value.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aScale*	Scale amount. Right shift the square of aSrc by aScale (0 <= aScale <= 15)

24.5 apexcv::AccumulateWeighted Class Reference

Accumulate Weighted.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, const float acAlpha, vsdk::SUMat &aDst)

Initialize object (required).

- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetAlpha (const float acAlpha)

  Set Alpha.
- APEXCV_LIB_RESULT GetAlpha (float &aAlpha)

  Set Alpha.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.5.1 Detailed Description

Accumulate Weighted.

Object of this class accumulates a weight value from *aSrc* to *aDst*
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1 or
Supported width: 128 to 2048 pixels.

24.5.2 Member Function Documentation

24.5.2.1 APEXCV_LIB_RESULT apexcv::AccumulateWeighted::GetAlpha (float & *aAlpha*)

Set Alpha.

This function allows to read the alpha value.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aAlpha*	Weight amount. Scalar value with a value in the range of [0, 1]

24.5.2.2 APEXCV_LIB_RESULT apexcv::AccumulateWeighted::Initialize (vsdk::SUMat & *aSrc,* const float *acAlpha,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in	*acAlpha*	Weight amount. Scalar value with a value in the range of [0, 1]
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.5.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.5.2.4 APEXCV_LIB_RESULT apexcv::AccumulateWeighted::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.5.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.5.2.6 APEXCV_LIB_RESULT apexcv::AccumulateWeighted::SetAlpha (const float *acAlpha*)

Set Alpha.

This function allows to change the alpha value.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*acAlpha*	Weight amount. Scalar value with a value in the range of [0, 1]

24.6 apexcv::Add Class Reference

Add.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetPolicy (apexcv::eConvertPolicy aPolicy)

  Set Policy type.
- APEXCV_LIB_RESULT GetPolicy (apexcv::eConvertPolicy &aPolicy)

  Get Policy type.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.6.1 Detailed Description

Add.

Object of this class adds pixel value from *aSrc1* and *aSrc2* pixel by pixel. *aDst* can be VSDK_CV_8UC1 only if both source images are VSDK_CV_8UC1 and *aDst* is explicitly set to VSDK_CV_8UC1. It is otherwise VSDK_CV_16SC1.
Supported input 1 type: VSDK_CV_8UC1, input 2 type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1 or
Supported input 1 type: VSDK_CV_8UC1, input 2 type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1 or
Supported input 1 type: VSDK_CV_8UC1, input 2 type: VSDK_CV_16SC1, output type: VSDK_CV_16SC1 or
Supported input 1 type: VSDK_CV_16SC1, input 2 type: VSDK_CV_16SC1, output type: VSDK_CV_16SC1
Supported width: 128 to 2048 pixels.

24.6.2 Member Function Documentation

24.6.2.1 APEXCV_LIB_RESULT apexcv::Add::GetPolicy (apexcv::eConvertPolicy & *aPolicy*)

Get Policy type.

This function allows to read the value of the overflow policy type.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

out	*aPolicy*	Overflow policy type.

24.6.2.2 APEXCV_LIB_RESULT apexcv::Add::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.6.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.6.2.4 APEXCV_LIB_RESULT apexcv::Add::ReconnectIO (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.6.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.6.2.6 APEXCV_LIB_RESULT apexcv::Add::SetPolicy (apexcv::eConvertPolicy *aPolicy*)

Set Policy type.

This function allows to change the overflow policy type.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aPolicy*	Overflow policy type

24.7 apexcv::Affine Class Reference

Host-ACF interface for the affine transformation.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aMat, vsdk::SUMat &aDst)

  Execute the process for affine transformation.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Execute the process for affine transformation.
- APEXCV_LIB_RESULT Process ()

  Execute the process for affine transformation.

24.7.1 Detailed Description

Host-ACF interface for the affine transformation.

This class is an interface for creating, initializing, processing and releasing the APU implementation of the affine transformation on APEX.

24.7.2 Member Function Documentation

24.7.2.1 APEXCV_LIB_RESULT apexcv::Affine::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aMat,* vsdk::SUMat & *aDst*)

Execute the process for affine transformation.

Returns

Error code for the execution (zero on success).

Execute chunk offset calculation for affine transformation and apply bilinear interpolation.

Parameters

aSrc	Source image buffer.
aMat	Affine matrix.
aDst	Destination image buffer. Needs to over allocate as double image size as DMA safe guard

24.7.2.2 APEXCV_LIB_RESULT apexcv::Affine::Process ( )

Execute the process for affine transformation.

Returns

Error code for the execution (zero on success).

Execute chunk offset calculation for affine transformation and apply bilinear interpolation.

24.7.2.3 APEXCV_LIB_RESULT apexcv::Affine::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Execute the process for affine transformation.

Returns

Error code for the execution (zero on success).

Execute chunk offset calculation for affine transformation and apply bilinear interpolation.

Parameters

aSrc	Source image buffer.
aDst	Destination image buffer. Needs to over allocate as double image size as DMA safe guard

24.8 apexcv::AggCFDetector Class Reference

apexcv::aggcf_detector is the host-ACF interface for aggregated channel feature (aggCF) based pedestrian detection.

Public Member Functions

- void InitDetectorModel (const char ∗)

  aggCF load pre-trained detector model
- void CalcScaleParameters (int aInWidth, int aInHight)

  aggCF based on input image size and detector model (octaves/scales/bin) calculate size of each scale
- int CalcChannelsPyramid (vsdk::SUMat &aDstLUV, vsdk::SUMat ∗apOutputPy)

  aggCF calculate feature pyramid on default APEX
- int CalcChannelsPyramid (vsdk::SUMat &aDstLUV, vsdk::SUMat ∗apOutputPy, int aApexID)

  aggCF calculate feature pyramid and assign the task to specified APEX
- std::vector< apexcv::bbs > ApplyPedDetectionDET (vsdk::SUMat ∗apChannelFeatures)

  aggCF perform pedestrian detection on extracted feature pyramid using detector model and return bounding boxes of detected pedestrians
- void ShowDetectorParameters ()

  aggCF display loaded detector model parameters
- void ApplyPedDetectionNMS (std::vector< apexcv::bbs > &aBbs, int aGreedy)

  aggCF perform non maximum suppression (NMS) on bounding boxes
- vsdk::SUMat ∗ InitPyramidBuf ()

  aggCF init feature pyramid structure and allocate feature data buffer
- void DeInitPyramidBuf (vsdk::SUMat ∗&apOutPy)

  aggCF release feature data buffer
- int CalcChannelsOctave (vsdk::SUMat &aDstLUV, vsdk::SUMat ∗apOutputPy, uint32_t aRealScaleIdx, int a↩ApexID)

  aggCF calculate one feature octave and assign the task to specified APEX. using this function, APP has full control on how to calculate feature pyramid including the order of each octave calculation within the pyramid, the assignment of octave calculationi task to different APU to achieve optimal performance.
- int CalcChannelsOctave (vsdk::SUMat &aDstLUV, vsdk::SUMat ∗apOutputPy, std::vector< uint32_t > &aReal↩ScaleIdx, std::vector< uint32_t > &aApexID)

  aggCF calculate feature octaves with APU using specified octave and APEX array. APEX code will decide how to dispatch octave calculation tasks to different APEX for performance and load balance purpose.
- bool IsDetectorModelFailToLoaded () const

  aggCF return the flag to show if a detector model is loaded successfully

24.8.1 Detailed Description

apexcv::aggcf_detector is the host-ACF interface for aggregated channel feature (aggCF) based pedestrian detection.

This class provides interfaces to load pre-trained detector model, calculate aggregated channel features including L↩ UV, magnitude and histogram of gradient, then perform pedestrian detection and output bounding boxes of detected pedestrians

24.8.2 Member Function Documentation

24.8.2.1 std::vector<apexcv::bbs> apexcv::AggCFDetector::ApplyPedDetectionDET (vsdk::SUMat ∗ *apChannelFeatures*)

aggCF perform pedestrian detection on extracted feature pyramid using detector model and return bounding boxes of detected pedestrians

24.8.2.2 void apexcv::AggCFDetector::ApplyPedDetectionNMS (std::vector< apexcv::bbs > & *aBbs,* int *aGreedy*)

aggCF perform non maximum suppression (NMS) on bounding boxes

24.8.2.3 int apexcv::AggCFDetector::CalcChannelsOctave (vsdk::SUMat & *aDstLUV,* vsdk::SUMat ∗ *apOutputPy,* uint32_t *aRealScaleIdx,* int *aApexID*)

aggCF calculate one feature octave and assign the task to specified APEX. using this function, APP has full control on how to calculate feature pyramid including the order of each octave calculation within the pyramid, the assignment of octave calculationi task to different APU to achieve optimal performance.

24.8.2.4 int apexcv::AggCFDetector::CalcChannelsOctave (vsdk::SUMat & *aDstLUV,* vsdk::SUMat ∗ *apOutputPy,* std::vector< uint32_t > & *aRealScaleIdx,* std::vector< uint32_t > & *aApexID*)

aggCF calculate feature octaves with APU using specified octave and APEX array. APEX code will decide how to dispatch octave calculation tasks to different APEX for performance and load balance purpose.

24.8.2.5 int apexcv::AggCFDetector::CalcChannelsPyramid (vsdk::SUMat & *aDstLUV,* vsdk::SUMat ∗ *apOutputPy*)

aggCF calculate feature pyramid on default APEX

24.8.2.6 int apexcv::AggCFDetector::CalcChannelsPyramid (vsdk::SUMat & *aDstLUV,* vsdk::SUMat ∗ *apOutputPy,* int *aApexID*)

aggCF calculate feature pyramid and assign the task to specified APEX

24.8.2.7 void apexcv::AggCFDetector::CalcScaleParameters (int *aInWidth,* int *aInHight*)

aggCF based on input image size and detector model (octaves/scales/bin) calculate size of each scale

24.8.2.8 void apexcv::AggCFDetector::DeInitPyramidBuf (vsdk::SUMat ∗& *apOutPy*)

aggCF release feature data buffer

24.8.2.9 void apexcv::AggCFDetector::InitDetectorModel (const char *)

aggCF load pre-trained detector model

24.8.2.10 vsdk::SUMat* apexcv::AggCFDetector::InitPyramidBuf ( )

aggCF init feature pyramid structure and allocate feature data buffer

24.8.2.11 bool apexcv::AggCFDetector::IsDetectorModelFailToLoaded ( ) const [inline]

aggCF return the flag to show if a detector model is loaded successfully

24.8.2.12 void apexcv::AggCFDetector::ShowDetectorParameters ( )

aggCF display loaded detector model parameters

24.9 apexcv::BilateralFilter Class Reference

Bilateral filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, int aSigmaColor, int aSigmaSpace, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetSigmaColor (int aSigmaColor)

  Set sigmaColor.
- APEXCV_LIB_RESULT SetSigmaSpace (int aSigmaSpace)

  Set sigmaSpace.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.9.1 Detailed Description

Bilateral filter.

Object of this class applies a bilateral filter on *aSrc*. *aSigmaColor* represents the weight of color/intensity difference and *aSigmaSpace* represents the weight of spacial difference. See: [8] for more information.
Supported window size: 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.9.2 Member Function Documentation

24.9.2.1 APEXCV_LIB_RESULT apexcv::BilateralFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* int *aSigmaColor,* int *aSigmaSpace,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size, 5.
in	*aSigmaColor*	Sigma value for color space.
in	*aSigmaSpace*	Sigma value for distance space.
in,out	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.9.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.9.2.3 APEXCV_LIB_RESULT apexcv::BilateralFilter::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.9.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.9.2.5 APEXCV_LIB_RESULT apexcv::BilateralFilter::SetSigmaColor (int *aSigmaColor*)

Set sigmaColor.

This function allows to change the value of sigmaColor

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSigmaColor*	Sigma value for color space.

24.9.2.6 APEXCV_LIB_RESULT apexcv::BilateralFilter::SetSigmaSpace (int *aSigmaSpace*)

Set sigmaSpace.

This function allows to change the value of sigmaSpace

Returns

APEXCV_LIB_RESULT Error code.

Parameters

aSigmaSpace	Sigma value for distance space.

24.10 apexcv::BitwiseAND Class Reference

Bitwise AND.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.10.1 Detailed Description

Bitwise AND.

Object of this class performs a bitwise AND between pixel value of *aSrc1* and *aSrc2* pixel by pixel.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1, or
Supported input type: VSDK_CV_16UC1, output type: VSDK_CV_16UC1, or
Supported input type: VSDK_CV_32UC1, output type: VSDK_CV_32UC1
Supported width: 128 to 2048 pixels.

24.10.2 Member Function Documentation

24.10.2.1 APEXCV_LIB_RESULT apexcv::BitwiseAND::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
`in`	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).

24.10.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.10.2.3 APEXCV_LIB_RESULT apexcv::BitwiseAND::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).

24.10.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.11 apexcv::BitwiseNOT Class Reference

Bitwise NOT.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.11.1 Detailed Description

Bitwise NOT.

Object of this class performs a bitwise NOT of pixel value of *aSrc* pixel by pixel.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1
Supported width: 128 to 2048 pixels.

24.11.2 Member Function Documentation

24.11.2.1 APEXCV_LIB_RESULT apexcv::BitwiseNOT::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.11.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.11.2.3 APEXCV_LIB_RESULT apexcv::BitwiseNOT::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.11.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.12 apexcv::BitwiseOR Class Reference

Bitwise OR.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)
  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)
  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)
  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()
  Start processing and return when done.

24.12.1 Detailed Description

Bitwise OR.

Object of this class performs a bitwise OR between pixel value of *aSrc1* and *aSrc2* pixel by pixel.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1, or
Supported input type: VSDK_CV_16UC1, output type: VSDK_CV_16UC1, or
Supported input type: VSDK_CV_32UC1, output type: VSDK_CV_32UC1
Supported width: 128 to 2048 pixels.

24.12.2 Member Function Documentation

24.12.2.1 APEXCV_LIB_RESULT apexcv::BitwiseOR::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).

24.12.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.12.2.3 APEXCV_LIB_RESULT apexcv::BitwiseOR::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).

24.12.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.13 apexcv::BitwiseXOR Class Reference

Bitwise exclusive OR.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.13.1 Detailed Description

Bitwise exclusive OR.

Object of this class performs a bitwise XOR between pixel value of *aSrc1* and *aSrc2* pixel by pixel.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1, or
Supported input type: VSDK_CV_16UC1, output type: VSDK_CV_16UC1, or
Supported input type: VSDK_CV_32UC1, output type: VSDK_CV_32UC1
Supported width: 128 to 2048 pixels.

24.13.2 Member Function Documentation

24.13.2.1 APEXCV_LIB_RESULT apexcv::BitwiseXOR::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).

24.13.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.13.2.3 APEXCV_LIB_RESULT apexcv::BitwiseXOR::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16UC1 or VSDK_CV_32UC1).

24.13.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.14 apexcv::Blockmatching Class Reference

Blockmatching class.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aOutputPoints, vsdk::SUMat &aOutputStatus, const vsdk::SUMat &aInputTemplate, const vsdk::SUMat &aInputWindow, const vsdk::SUMat &aInputPoints, int aSad_threshold, int aTrackedPoints=-1, int aTsx=DEFAULT_TEMPLATE_SIZE, int aTsy=DEFAULT_TEMPLATE_SIZE, int aWsx=DEFAULT_WINDOW_SIZE, int aWsy=DEFAULT_WINDOW_SIZE, int aNcu_x=1, int aNcu_y=1)

  Initialize the block matching.
- APEXCV_LIB_RESULT Process (int aTracked_points=-1)

  Match the blocks.
- void Release ()

  Release Resources.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aOutputPoints, vsdk::SUMat &aOutputStatus, const vsdk::SUMat &aInputTemplate, const vsdk::SUMat &aInputWindow, const vsdk::SUMat &aInputPoints, int aTrackedPoints=-1)

  Reconnect IO.
- APEXCV_LIB_RESULT SetSadThreshold (int aSadThreshold)

  Set SAD Threshold.

24.14.1 Detailed Description

Blockmatching class.

This class is an interface for using the block matching algorithm on the APEX.

24.14.2 Member Function Documentation

24.14.2.1 APEXCV_LIB_RESULT apexcv::Blockmatching::Initialize (vsdk::SUMat & *aOutputPoints,* vsdk::SUMat & *aOutputStatus,* const vsdk::SUMat & *aInputTemplate,* const vsdk::SUMat & *aInputWindow,* const vsdk::SUMat & *aInputPoints,* int *aSad_threshold,* int *aTrackedPoints* = `-1`, int *aTsx* = `DEFAULT_TEMPLATE_SIZE`, int *aTsy* = `DEFAULT_TEMPLATE_SIZE`, int *aWsx* = `DEFAULT_WINDOW_SIZE`, int *aWsy* = `DEFAULT_WINDOW_SIZE`, int *aNcu_x* = `1`, int *aNcu_y* = `1`)

Initialize the block matching.

Initializes the internal buffers for the process. The *lOutputPoints* and *lInputPoints* are 32-bit packed values of form [hi, low] [Y_16S, X_16S] The effective search window size is: search_width = (wsx-tsx+1)∗(ncu_x-1) + wsx; search_height = (wsy-tsy+1)∗(ncu_y-1) + wsy;

The search window will be centered around the tracked point.

Parameters

aOutputPoints	The output tracked points (X, Y)
aOutputStatus	Indicates if the point's SAD score is below the threshold
aInputTemplate	Template image
aInputWindow	Search Image
aInputPoints	Set of points on the template and will be used to specify the locations on the search image
aSad_threshold	Maximum SAD score - Capped to 65535
aTrackedPoints	Number of points to track. Default -1: Uses size of *lInputPoints*
aTsx	Template window columns
aTsy	Template window rows

Parameters

aWsx	Search window columns
aWsy	Search window rows
aNcu_x	Number of CU's to use for the horizontal search window per tracked point
aNcu_y	Number of CU's to use for the vertical search window per tracked point

24.14.2.2 APEXCV_LIB_RESULT apexcv::Blockmatching::Process (int *aTracked_points* = `-1`)

Match the blocks.

The block matching process uses the Sum of Absolute Differences algorithm to perform block matching. The process will search in the "new" image within a window around the search points in the "previous" image. The template size determines how many pixels are used in the SAD calculation. The window size determines the area in which the template is used to perform SAD calculations.

Default Template Size: 16x16

- Must be multiple of 4 Default Window Size: 28x28
- Must be multiple of 4 In total template_x$*$template_y + window_x$*$window_y must be $<$= $\sim$1800

24.14.2.3 APEXCV_LIB_RESULT apexcv::Blockmatching::ReconnectIO (vsdk::SUMat & *aOutputPoints,* vsdk::SUMat & *aOutputStatus,* const vsdk::SUMat & *aInputTemplate,* const vsdk::SUMat & *aInputWindow,* const vsdk::SUMat & *aInputPoints,* int *aTrackedPoints* = `-1`)

Reconnect IO.

Use this to reconnect the input and output buffers. This only needs to be done if the connected Input/Outputs are changed. If only the data within (no size, data pointer, or type changes), then this does not need to be called.

Parameters

aOutputPoints	The output tracked points (X, Y)
aOutputStatus	Indicates if the point's SAD score is below the threshold
aInputTemplate	Template image
aInputWindow	Search Image
aInputPoints	Set of points on the template and will be used to specify the locations on the search image
aTrackedPoints	Number of points to track. Default -1: Uses size of *lInputPoints*

24.14.2.4 void apexcv::Blockmatching::Release ( )

Release Resources.

Releases the internal buffers and resets the class state to initial.

24.14.2.5 APEXCV_LIB_RESULT apexcv::Blockmatching::SetSadThreshold (int *aSadThreshold*)

Set SAD Threshold.

Change the SAD threshold

Parameters

aSadThreshold	Maximum SAD score - Capped to 65535

24.15 apexcv::BoxFilter Class Reference

Box filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.15.1 Detailed Description

Box filter.

Object of this class applies a box filter to *aSrc*. Supported window size: 3x3, 5x5, 7x7 and 9x9 (for VSDK_CV_8UC1)
Supported window size: 3x3 and 5x5 (for VSDK_CV_16SC1)
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1 and VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.15.2 Member Function Documentation

24.15.2.1 APEXCV_LIB_RESULT apexcv::BoxFilter::Initialize (vsdk::SUMat & *aSrc*, int *aWindowSize*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1)
`in`	*aWindowSize*	Window size (3, 5, 7 or 9)
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1)

24.15.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.15.2.3 APEXCV_LIB_RESULT apexcv::BoxFilter::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).

24.15.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ *ApexId*	ID of the APEX core used for performing the processing (0 or 1).

24.16 apexcv::BoxFilterHT Class Reference

Box filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.16.1 Detailed Description

Box filter, Hand Tuned (optimized).

Object of this class applies a Box filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.16.2 Member Function Documentation

24.16.2.1 APEXCV_LIB_RESULT apexcv::BoxFilterHT::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.16.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.16.2.3 APEXCV_LIB_RESULT apexcv::BoxFilterHT::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.16.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.17 apexcv::Brief Class Reference

BRIEF class.

Public Types

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aImage, std::vector< signed char > &aSmplPattern, std↩::vector< unsigned int > &aKeypoints, FilteringType aFilterType, unsigned char aDescrSizeBytes, unsigned char aBorderSize, vsdk::SUMat &aDescriptors)

  Initializes the Brief class and calculates the chunks offsets.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aImage, std::vector< signed char > &aSmplPattern, std↩::vector< unsigned int > &aKeypoints, FilteringType aFilterType, unsigned char aDescrSizeBytes, unsigned char aBorderSize, vsdk::SUMat &aDescriptors)

  Reinitializes the Brief class and calculates the chunks offsets.

- APEXCV_LIB_RESULT SelectApuConfiguration (ACF_APU_CFG aApuConfig=ACF_APU_CFG__DEFAULT, int32_t aApexId=0)

  APEX hardware configuration.
- APEXCV_LIB_RESULT Process ()

  Starts the APEX processing.

24.17.1 Detailed Description

BRIEF class.

This class is an interface for using the BRIEF (Binary Robust Independent Elementary Features) algorithm.

24.17.2 Member Enumeration Documentation

24.17.2.1 enum apexcv::Brief::FilteringType

Filtering type size enum.

24.17.3 Member Function Documentation

24.17.3.1 APEXCV_LIB_RESULT apexcv::Brief::Initialize (vsdk::SUMat & *aImage,* std::vector< signed char > & *aSmplPattern,* std::vector< unsigned int > & *aKeypoints,* FilteringType *aFilterType,* unsigned char *aDescrSizeBytes,* unsigned char *aBorderSize,* vsdk::SUMat & *aDescriptors*)

Initializes the Brief class and calculates the chunks offsets.

Parameters

in	*aImage*	- 8-bit grayscale source image
in	*aSmplPattern*	- cartesian coordinate pairs that describe the sampling pattern - <x0, y0>, <x1, y1>, ...
in	*aKeypoints*	- 16-bit unsigned <x, y> cartesian coordinates that pinpoint a keypoint
in	*aFilterType*	- See type to understand the filtering
in	*aDescrSizeBytes*	- 16, 32 or 64 Bytes descriptors
in	*aBorderSize*	- The descriptors for the keypoints inside the border will not be computed.
out	*aDescriptors*	- 8-bit descriptors

Returns

Check apexcv_error_codes.h to see to possible outcomes

Note

Max resolution is **1920 x 1080 pixels** with **32** CUs.

24.17.3.2 APEXCV_LIB_RESULT apexcv::Brief::Process ( )

Starts the APEX processing.

Returns

Check apexcv_error_codes.h to see to possible outcomes

Note

Maximum resolution is **1920 x 1080 pixels** with **64** CUs.

24.17.3.3 APEXCV_LIB_RESULT apexcv::Brief::ReconnectIO (vsdk::SUMat & *aImage,* std::vector< signed char > & *aSmplPattern,* std::vector< unsigned int > & *aKeypoints,* FilteringType *aFilterType,* unsigned char *aDescrSizeBytes,* unsigned char *aBorderSize,* vsdk::SUMat & *aDescriptors*)

Reinitializes the Brief class and calculates the chunks offsets.

Parameters

in	*aImage*	- 8-bit grayscale source image
in	*aSmplPattern*	- cartesian coordinate pairs that describe the sampling pattern - <x0, y0>, <x1, y1>, ...
in	*aKeypoints*	- 16-bit unsigned <x, y> cartesian coordinates that pinpoint a keypoint
in	*aFilterType*	- See type to understand the filtering
in	*aDescrSizeBytes*	- 16, 32 or 64 Bytes descriptors
in	*aBorderSize*	- The descriptors for the keypoints inside the border will not be computed.
out	*aDescriptors*	- 8-bit descriptors

Returns

Check apexcv_error_codes.h to see to possible outcomes

Note

Max resolution is **1920 x 1080 pixels** with **32** CUs.

24.17.3.4 APEXCV_LIB_RESULT apexcv::Brief::SelectApuConfiguration (ACF_APU_CFG *aApuConfig =* ACF_APU_CFG__DEFAULT, int32_t *aApexId =* 0)

APEX hardware configuration.

Parameters

in	*aApuConfig*	Apu CU size
in	*aApexId*	Apex id where the code will execute

Returns

Please check apexcv_error_codes.hpp

24.18 apexcv::Canny Class Reference

ApexCV Canny Edge Detector.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst, uint16_t aLow, uint16_t aHigh)

  Initialization.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO.
- APEXCV_LIB_RESULT SetConfiguration (CannyConfig config)

  Set Edge Promotion Iteration and select separated kernel process or combined kernel process.
- void GetConfiguration (CannyConfig &aConfig)

  Return the configuration structure.
- APEXCV_LIB_RESULT SetThresholds (uint16_t aLow, uint16_t aHigh)

  Set the Edge Hysteresis Thresholds.
- APEXCV_LIB_RESULT GetThresholds (uint16_t &aLow, uint16_t &aHigh)

  Return the Edge Hysteresis Thresholds.
- APEXCV_LIB_RESULT Process ()

  Runs the Canny algorithm.
- APEXCV_LIB_RESULT PromoteEdges (int aIterations)

  Runs the Edge promotion extra times.

24.18.1 Detailed Description

ApexCV Canny Edge Detector.

apexcv::Canny is the Host-ACF interface for creating, initializing, executing and releasing the Canny Edge Detector on Apex.

24.18.2 Member Function Documentation

24.18.2.1 void apexcv::Canny::GetConfiguration (CannyConfig & *aConfig*)

Return the configuration structure.

Returns current configuration structure in Canny class.

24.18.2.2 APEXCV_LIB_RESULT apexcv::Canny::GetThresholds (uint16_t & *aLow,* uint16_t & *aHigh*)

Return the Edge Hysteresis Thresholds.

Returns the low and high thresholds for edge hysteresis.

24.18.2.3 APEXCV_LIB_RESULT apexcv::Canny::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst,* uint16_t *aLow,* uint16_t *aHigh*)

Initialization.

Initializes the intermediate buffers needed for the processes, initializes the ACF processes and connect buffers to processes' IO. The number of iterations refers to the number of times the block connection process is to be run.

Parameters

aSrc	8-bit grayscale source image
aDst	8-bit destination image
aLow	16-bit low threshold for edge hysteresis
aHigh	16-bit high threshold for edge hysteresis

24.18.2.4 APEXCV_LIB_RESULT apexcv::Canny::Process ( )

Runs the Canny algorithm.

This will run the Canny algorithm. Before this is called, the apexcv::Canny::Initialize function must be called with the appropriate parameters.

The input image is an 8-bit grayscale image. The Canny detection algorithm works best if the image has been smoothed to get rid of noise. The output is an 8-bit grayscale image with the detected edges set to 255 and non edges set to 0.

24.18.2.5 APEXCV_LIB_RESULT apexcv::Canny::PromoteEdges (int *alterations*)

Runs the Edge promotion extra times.

This will run extra edge promotion iterations on a the previously obtained result using either *promoteEdges()* or *process()* or *processCombined()*. Only valid after running *process()* or *processCombined()*.

24.18.2.6 APEXCV_LIB_RESULT apexcv::Canny::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO.

Use this to reconnect the input and output buffers. This only needs to be done if the connected Input/Outputs are changed. If only the data within (no size, data pointer, or type changes), then this does not need to be called.

Parameters

aSrc	8-bit grayscale source image
aDst	8-bit destination image

24.18.2.7 APEXCV_LIB_RESULT apexcv::Canny::SetConfiguration (CannyConfig *config*)

Set Edge Promotion Iteration and select separated kernel process or combined kernel process.

Set give configuration data to the configuration structure in Canny class

24.18.2.8 APEXCV_LIB_RESULT apexcv::Canny::SetThresholds (uint16_t *aLow,* uint16_t *aHigh*)

Set the Edge Hysteresis Thresholds.

Sets the low and high thresholds for edge hysteresis. Only affects the *process()* call. This does not affect the *promote←Edges()* call.

24.19 apexcv::CensusFilter Class Reference

Census filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.19.1 Detailed Description

Census filter.

Object of this class applies a census filter to *aSrc*. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.19.2 Member Function Documentation

24.19.2.1 APEXCV_LIB_RESULT apexcv::CensusFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size (3).
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.19.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.19.2.3 APEXCV_LIB_RESULT apexcv::CensusFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.19.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.20 apexcv::Clz Class Reference

Count of Leading Zeros.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.20.1 Detailed Description

Count of Leading Zeros.

Object of this class counts the number of leading zeros in the pixel value of *aSrc* pixel by pixel.
Supported input type: VSDK_CV_8UC1, VSDK_CV_8SC1, VSDK_CV_16UC1 and VSDK_CV_16SC1
Supported output type: VSDK_CV_8UC1
Supported width: 128 to 2048 pixels.

24.20.2 Member Function Documentation

24.20.2.1 APEXCV_LIB_RESULT apexcv::Clz::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1, VSDK_CV_8SC1, VSDK_CV_16UC1 or VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.20.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.20.2.3 APEXCV_LIB_RESULT apexcv::Clz::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1, VSDK_CV_8SC1, VSDK_CV_16UC1 or VSDK_CV_16SC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.20.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.21 apexcv::ColorConverter Class Reference

Color converter class.

Public Types

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, ConversionType aCT, int aR2YFactor, int aG2YFactor, int aB2YFactor, vsdk::SUMat &aDst)

  Convert function.
- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, ConversionType aCT, vsdk::SUMat &aDst)

  Convert function. ..
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO. ..

- APEXCV_LIB_RESULT SetFactors (int aR2YFactor, int aG2YFactor, int aB2YFactor)

  Set factors for RGB888x to Y.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.21.1 Detailed Description

Color converter class.

Object of this class performs color conversions of images.
See ConversionType for supported converions.
Supported width: 128 to 2048 pixels.

24.21.2 Member Enumeration Documentation

24.21.2.1 enum apexcv::ColorConverter::ConversionType

List of conversion types.

Enumerator

eRGB565_TO_RGB888X 16-bit RGB565 (VSDK_CV_16UC1) to 32-bit representation of RGB888X (VSDK_CV_32SC1)

eRGB888X_TO_RGB565 32-bit representation of RGB888X (VSDK_CV_32SC1) to 16-bit RGB565 (VSDK_CV_16UC1)

eRGB888X_TO_Y 4-tuple 8-bit R, G, B, X (VSDK_CV_8UC4) to 8-bit Y (VSDK_CV_8UC1)

eRGB888X_TO_YUV 4-tuple 8-bit R, G, B, X (VSDK_CV_8UC4) to 4-tuple 8-bit Y, U, V, X (VSDK_CV_8UC4)

eRGB888_TO_GREY 3-tuple 8-bit R, G, B (VSDK_CV_8UC3) to 8-bit Grey (VSDK_CV_8UC1), (R∗21 + G∗72 + B∗7)

eBGR888_TO_GREY 3-tuple 8-bit B, G, R (VSDK_CV_8UC3) to 8-bit Grey (VSDK_CV_8UC1), (R∗21 + G∗72 + B∗7)

eGREY_TO_RGB888 8-bit grey (VSDK_CV_8UC1) to 3-tuple 8-bit B, G, R (VSDK_CV_8UC3), duplication on all 3 channels

24.21.3 Member Function Documentation

24.21.3.1 APEXCV_LIB_RESULT apexcv::ColorConverter::Initialize (vsdk::SUMat & *aSrc*, ConversionType *aCT*, int *aR2YFactor*, int *aG2YFactor*, int *aB2YFactor*, vsdk::SUMat & *aDst*)

Convert function.

Converts an image from one type to another based on ConversionType. R2YFactor, G2YFactor and B2YFactor are Q0.8 fixed point values used with RGB888X_TO_Y following the formula: $Y = \left\lfloor \frac{R2YFactor}{256} * R + \frac{G2YFactor}{256} * G + \frac{B2YFactor}{256} * B + 0.5 \right\rfloor$ For example, conversion following Recommendation ITU-R BT.601-7 (`http://www.itu.int/rec/R-REC-BT.601-7-201103-I/en`) would use factor values of 77(0.299), 150(0.587) and 29(0.114).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source memory buffer (Use the type corresponding to the ConversionType selected).
`in`	*aCT*	Color conversion type. See ConversionType
`in`	*aR2YFactor*	Conversion factor for red used with RGB888X_TO_Y
`in`	*aG2YFactor*	Conversion factor for green used with RGB888X_TO_Y
`in`	*aB2YFactor*	Conversion factor for blue used with RGB888X_TO_Y
`in,out`	*aDst*	Destination memory buffer (Use the type corresponding to the ConversionType selected).

24.21.3.2 APEXCV_LIB_RESULT apexcv::ColorConverter::Initialize (vsdk::SUMat & *aSrc*, ConversionType *aCT*, vsdk::SUMat & *aDst*)

Convert function. ..

Converts an image from one type to another based on ConversionType. R2YFactor, G2YFactor and B2YFactor are Q0.8 fixed point values used with RGB888X_TO_Y following the formula: $Y = \lfloor \frac{R2YFactor}{256} * R + \frac{G2YFactor}{256} * G + \frac{B2YFactor}{256} * B + 0.5 \rfloor$ For example, conversion following Recommendation ITU-R BT.601-7 (`http://www.itu.int/rec/R-REC-BT.601-7-201103-I/en`) would use factor values of 77(0.299), 150(0.587) and 29(0.114).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source memory buffer.
`in`	*aCT*	Color conversion type. See ConversionType
`in,out`	*aDst*	Destination memory buffer.

24.21.3.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.21.3.4 APEXCV_LIB_RESULT apexcv::ColorConverter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO. ..

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer (type should be the same one used when calling Initialize).
in,out	*aDst*	Destination memory buffer (type should be the same one used when calling Initialize).

24.21.3.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.21.3.6 APEXCV_LIB_RESULT apexcv::ColorConverter::SetFactors (int *aR2YFactor,* int *aG2YFactor,* int *aB2YFactor*)

Set factors for RGB888x to Y.

This function allows to change factors without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

aR2YFactor	Conversion factor for red used with RGB888X_TO_Y
aG2YFactor	Conversion factor for green used with RGB888X_TO_Y
aB2YFactor	Conversion factor for blue used with RGB888X_TO_Y

24.22 apexcv::ColorConverterHT Class Reference

Optimized color converter class containing support for converting image color types.

Public Types

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, ConversionType aCT, uint8_t aR2YFactor, uint8_t aG2YFactor, uint8_t aB2YFactor, uint16_t aShiftFactor, vsdk::SUMat &aDst)

  Convert function.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO.
- APEXCV_LIB_RESULT SetFactors (uint8_t aR2YFactor, uint8_t aG2YFactor, uint8_t aB2YFactor, uint16_t aShiftFactor)

  Set factors for RGB888x to Y.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.22.1 Detailed Description

Optimized color converter class containing support for converting image color types.

This class is an interface for using color conversion functions on the host.

24.22.2 Member Enumeration Documentation

24.22.2.1 enum apexcv::ColorConverterHT::ConversionType

List of conversion types.

Enumerator

eHT_RGB888X_TO_Y 4-tuple 8 bit A, B, C, X (VSDK_CV_8UC4) to signed 16 bit Y (VSDK_CV_16SC1)

24.22.3 Member Function Documentation

24.22.3.1 APEXCV_LIB_RESULT apexcv::ColorConverterHT::Initialize (vsdk::SUMat & *aSrc*, ConversionType *aCT*, uint8_t *aR2YFactor*, uint8_t *aG2YFactor*, uint8_t *aB2YFactor*, uint16_t *aShiftFactor*, vsdk::SUMat & *aDst*)

Convert function.

Converts an image from one type to another based on ConversionType. R2YFactor, G2YFactor and B2YFactor are Q0.8 fixed point values used with RGB888X_TO_Y following the formula: $Y = \lfloor \frac{R2YFactor}{256} * R + \frac{G2YFactor}{256} * G + \frac{B2YFactor}{256} * B + 0.5 \rfloor$ For example, conversion following Recommendation ITU-R BT.601-7 (`http://www.itu.int/rec/R-REC-BT.601-7-201103-I/en`) would use factor values of 77(0.299), 150(0.587) and 29(0.114).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer (Use the type corresponding to the ConversionType selected).
in	*aCT*	Color conversion type. See ConversionType
in	*aR2YFactor*	Conversion factor for red used with RGB888X_TO_Y
in	*aG2YFactor*	Conversion factor for green used with RGB888X_TO_Y
in	*aB2YFactor*	Conversion factor for blue used with RGB888X_TO_Y
in	*aShiftFactor*	Shift factor. Use 0 by default. Used in HT_RGB888X_TO_Y.
in,out	*aDst*	Destination memory buffer (Use the type corresponding to the ConversionType selected).

24.22.3.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.22.3.3 APEXCV_LIB_RESULT apexcv::ColorConverterHT::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer (type should be the same one used when calling Initialize).
in,out	*aDst*	Destination memory buffer (type should be the same one used when calling Initialize).

24.22.3.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.22.3.5 APEXCV_LIB_RESULT apexcv::ColorConverterHT::SetFactors (uint8_t *aR2YFactor,* uint8_t *aG2YFactor,* uint8_t *aB2YFactor,* uint16_t *aShiftFactor*)

Set factors for RGB888x to Y.

This function allows to change factors without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aR2YFactor*	Conversion factor for red used with HT_RGB888X_TO_Y
in	*aG2YFactor*	Conversion factor for green used with HT_RGB888X_TO_Y
in	*aB2YFactor*	Conversion factor for blue used with HT_RGB888X_TO_Y
in	*aShiftFactor*	Shift factor. Use 0 by default. Used in HT_RGB888X_TO_Y.

24.23 apexcv::Hog::Config Struct Reference

The HOG parameters for the linear SVM classifier.

Public Attributes

- uint32_t mDetWinWidth
- uint32_t mDetWinHeight
- uint32_t mBlockWidth
- uint32_t mBlockHeight
- uint32_t mStrideWidth
- uint32_t mStrideHeight
- uint32_t mHistogramBins
- SVMTransformMode mSVMTransformMode

24.23.1 Detailed Description

The HOG parameters for the linear SVM classifier.

24.23.2 Member Data Documentation

24.23.2.1 uint32_t apexcv::Hog::Config::mBlockHeight

The vertical size of a Hog block. Keep unchanged.

24.23.2.2 uint32_t apexcv::Hog::Config::mBlockWidth

The horizontal size of a Hog block. Keep unchanged.

24.23.2.3 uint32_t apexcv::Hog::Config::mDetWinHeight

The height of the detection window in pixels

24.23.2.4 uint32_t apexcv::Hog::Config::mDetWinWidth

The width of the detection window in pixels

24.23.2.5 uint32_t apexcv::Hog::Config::mHistogramBins

The number of bins in HOG descriptor. Keep unchanged.

24.23.2.6 uint32_t apexcv::Hog::Config::mStrideHeight

The vertical stride of the detection window. Keep unchanged.

24.23.2.7 uint32_t apexcv::Hog::Config::mStrideWidth

The horizontal stride of the detection window. Keep unchanged.

24.23.2.8 SVMTransformMode apexcv::Hog::Config::mSVMTransformMode

SVM Transformation Mode for manipulating existing SVM detector

24.24 apexcv::ConvertDepth Class Reference

Converts image bit depth.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, const int32_t acShift, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetShift (const int32_t acShift)

Set Shift.

- APEXCV_LIB_RESULT GetShift (int32_t &aShift)

  Get Shift.
- APEXCV_LIB_RESULT SetPolicyType (apexcv::eConvertPolicy aPolicy)

  Set Policy type.
- APEXCV_LIB_RESULT GetPolicyType (apexcv::eConvertPolicy &aPolicy)

  Get Policy type.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.24.1 Detailed Description

Converts image bit depth.

Object of this class performs converts image bit depth.
Up convert, shifting left. Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1
Down convert, shifting right. Supported input type: VSDK_CV_16SC1, output type: VSDK_CV_8UC1
Supported width: 128 to 2048 pixels.

24.24.2 Member Function Documentation

24.24.2.1 APEXCV_LIB_RESULT apexcv::ConvertDepth::GetPolicyType (apexcv::eConvertPolicy & *aPolicy*)

Get Policy type.

This function allows to read the value of the Policy type.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`out`	*aPolicy*	Policy.

24.24.2.2 APEXCV_LIB_RESULT apexcv::ConvertDepth::GetShift (int32_t & *aShift*)

Get Shift.

This function allows to read the value of the shift.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`out`	*aShift*	shift.

24.24.2.3 APEXCV_LIB_RESULT apexcv::ConvertDepth::Initialize (vsdk::SUMat & *aSrc,* const int32_t *acShift,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).
in	*acShift*	Source pixel value shift amount (0 <= aShift < 8).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).

24.24.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.24.2.5 APEXCV_LIB_RESULT apexcv::ConvertDepth::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).

24.24.2.6 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.24.2.7 APEXCV_LIB_RESULT apexcv::ConvertDepth::SetPolicyType (apexcv::eConvertPolicy *aPolicy*)

Set Policy type.

This function allows to change the Policy type.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aPolicy*	Policy type

24.24.2.8 APEXCV_LIB_RESULT apexcv::ConvertDepth::SetShift (const int32_t *acShift*)

Set Shift.

This function allows to change the shift fact value.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*acShift*	Source pixel value shift amount (0 <= aShift < 8)

24.25 apexcv::ConvolveFilter Class Reference

Convolve filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, signed char(&aFilterCoeff)[9], int aFilterScale, vsdk::SU↩ Mat &aDst)

    Initialize object (required).

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, signed char(&aFilterCoeff)[25], int aFilterScale, vsdk::S←UMat &aDst)

    Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

    Reconnect IO (optional).
- APEXCV_LIB_RESULT SetFilterScale (int aFilterScale)

    Set Filter Scale.
- APEXCV_LIB_RESULT SetFilterCoeff (signed char(&filterCoeff)[9])

    Set Filter Coefficients.
- APEXCV_LIB_RESULT SetFilterCoeff (signed char(&filterCoeff)[25])

    Set Filter Coefficients.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

    Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

    Start processing and return when done.

24.25.1 Detailed Description

Convolve filter.

Object of this class applies a Convolve filter to *aSrc*. Supported window size: 3 x 3 or 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.25.2 Member Function Documentation

24.25.2.1 APEXCV_LIB_RESULT apexcv::ConvolveFilter::Initialize (vsdk::SUMat & *aSrc,* signed char(&) *aFilterCoeff[9],* int *aFilterScale,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aFilterCoeff*	9 Coefficients for 3x3 kernel.
in	*aFilterScale*	Right Shift to scale the data.
in,out	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.25.2.2 APEXCV_LIB_RESULT apexcv::ConvolveFilter::Initialize (vsdk::SUMat & *aSrc*, signed char(&) *aFilterCoeff[25]*, int *aFilterScale*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aFilterCoeff*	25 Coefficients for 5x5 kernel.
in	*aFilterScale*	Right Shift to scale the data.
in,out	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.25.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.25.2.4 APEXCV_LIB_RESULT apexcv::ConvolveFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.25.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.25.2.6 APEXCV_LIB_RESULT apexcv::ConvolveFilter::SetFilterCoeff (signed char(&) *filterCoeff[9]*)

Set Filter Coefficients.

This function allows to change the filter 9 coefficients for 3x3 kernels

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*filterCoeff*	9 Coefficients for 3x3 kernel.

24.25.2.7 APEXCV_LIB_RESULT apexcv::ConvolveFilter::SetFilterCoeff (signed char(&) *filterCoeff[25]*)

Set Filter Coefficients.

This function allows to change the filter 25 coefficients for 5x5 kernels

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*filterCoeff*	25 Coefficients for 5x5 kernel.

24.25.2.8 APEXCV_LIB_RESULT apexcv::ConvolveFilter::SetFilterScale (int *aFilterScale*)

Set Filter Scale.

This function allows to change the filter scale (right shift).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aFilterScale*	Set the filter Scale.

24.26 apexcv::ConvolveFilterHT Class Reference

Convolve filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, signed char(&aFilterCoeff)[9], signed char aFilterScale, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, signed char(&aFilterCoeff)[25], signed char aFilterScale, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetFilterScale (signed char aFilterScale)

  Set Filter Scale.
- APEXCV_LIB_RESULT SetFilterCoeff (signed char(&filterCoeff)[9])

  Set Filter Coefficients.
- APEXCV_LIB_RESULT SetFilterCoeff (signed char(&filterCoeff)[25])

  Set Filter Coefficients.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.26.1 Detailed Description

Convolve filter, Hand Tuned (optimized).

Object of this class applies a generic convolution filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.26.2 Member Function Documentation

24.26.2.1 APEXCV_LIB_RESULT apexcv::ConvolveFilterHT::Initialize (vsdk::SUMat & *aSrc*, signed char(&) *aFilterCoeff[9]*, signed char *aFilterScale*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aFilterCoeff*	9 Coefficients for 3x3 kernel.
in	*aFilterScale*	Right Shift to scale the data.
in,out	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.26.2.2 APEXCV_LIB_RESULT apexcv::ConvolveFilterHT::Initialize (vsdk::SUMat & *aSrc*, signed char(&) *aFilterCoeff[25]*, signed char *aFilterScale*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aFilterCoeff*	25 Coefficients for 5x5 kernel.
in	*aFilterScale*	Right Shift to scale the data.
in,out	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.26.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.26.2.4 APEXCV_LIB_RESULT apexcv::ConvolveFilterHT::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.26.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.26.2.6 APEXCV_LIB_RESULT apexcv::ConvolveFilterHT::SetFilterCoeff (signed char(&) *filterCoeff[9]*)

Set Filter Coefficients.

This function allows to change the filter 9 coefficients for 3x3 kernels

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*filterCoeff*	9 Coefficients for 3x3 kernel.

24.26.2.7 APEXCV_LIB_RESULT apexcv::ConvolveFilterHT::SetFilterCoeff (signed char(&) *filterCoeff[25]*)

Set Filter Coefficients.

This function allows to change the filter 25 coefficients for 5x5 kernels

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*filterCoeff*	25 Coefficients for 5x5 kernel.

24.26.2.8 APEXCV_LIB_RESULT apexcv::ConvolveFilterHT::SetFilterScale (signed char *aFilterScale*)

Set Filter Scale.

This function allows to change the filter scale (right shift).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aFilterScale*	Set the filter Scale.

24.27 apexcv::Orb::Corner Class Reference

ORB::Corner.

24.27.1 Detailed Description

ORB::Corner.

The class is used to classify the keypoints found by FAST9 The member - strength - is the metric for discriminating between different corners. The higher the value, the higher the probabily to be a corner.

24.28 apexcv::DerivativeXFilterHT Class Reference

Derivative X filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, signed char aK0, signed char aK1, signed char aK2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).

- APEXCV_LIB_RESULT SetK0 (signed char k0)

  Set K0.
- APEXCV_LIB_RESULT SetK1 (signed char k1)

  Set K1.
- APEXCV_LIB_RESULT SetK2 (signed char k2)

  Set K2.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.28.1 Detailed Description

Derivative X filter, Hand Tuned (optimized).

Object of this class applies a Derivative X filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.28.2 Member Function Documentation

24.28.2.1 APEXCV_LIB_RESULT apexcv::DerivativeXFilterHT::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* signed char *aK0,* signed char *aK1,* signed char *aK2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size, 3
in	*aK0*	K0
in	*aK1*	K1
in	*aK2*	K2
in,out	*aDst*	Destination Image buffer (VSDK_CV_16SC1).

24.28.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.28.2.3 APEXCV_LIB_RESULT apexcv::DerivativeXFilterHT::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.28.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.28.2.5 APEXCV_LIB_RESULT apexcv::DerivativeXFilterHT::SetK0 (signed char *k0*)

Set K0.

This function allows to change the Input parameter K0

Returns

APEXCV_LIB_RESULT Error code.

24.28.2.6 APEXCV_LIB_RESULT apexcv::DerivativeXFilterHT::SetK1 (signed char *k1*)

Set K1.

This function allows to change the Input parameter K1

Returns

APEXCV_LIB_RESULT Error code.

24.28.2.7 APEXCV_LIB_RESULT apexcv::DerivativeXFilterHT::SetK2 (signed char *k2*)

Set K2.

This function allows to change the Input parameter K2

Returns

APEXCV_LIB_RESULT Error code.

24.29 apexcv::DerivativeYFilterHT Class Reference

Derivative Y filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, signed char aK0, signed char aK1, signed char aK2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetK0 (signed char k0)

  Set K0.
- APEXCV_LIB_RESULT SetK1 (signed char k1)

  Set K1.
- APEXCV_LIB_RESULT SetK2 (signed char k2)

  Set K2.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.29.1 Detailed Description

Derivative Y filter, Hand Tuned (optimized).

Object of this class applies a Derivative Y filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.29.2 Member Function Documentation

24.29.2.1 APEXCV_LIB_RESULT apexcv::DerivativeYFilterHT::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* signed char *aK0,* signed char *aK1,* signed char *aK2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size, 3
in	*aK0*	K0
in	*aK1*	K1
in	*aK2*	K2
in,out	*aDst*	Destination Image buffer (VSDK_CV_16SC1).

24.29.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.29.2.3 APEXCV_LIB_RESULT apexcv::DerivativeYFilterHT::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.29.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.29.2.5 APEXCV_LIB_RESULT apexcv::DerivativeYFilterHT::SetK0 (signed char *k0*)

Set K0.

This function allows to change the Input parameter K0

Returns

APEXCV_LIB_RESULT Error code.

24.29.2.6 APEXCV_LIB_RESULT apexcv::DerivativeYFilterHT::SetK1 (signed char *k1*)

Set K1.

This function allows to change the Input parameter K1

Returns

APEXCV_LIB_RESULT Error code.

24.29.2.7 APEXCV_LIB_RESULT apexcv::DerivativeYFilterHT::SetK2 (signed char *k2*)

Set K2.

This function allows to change the Input parameter K2

Returns

APEXCV_LIB_RESULT Error code.

24.30 apexcv::DilateFilter Class Reference

Dilate filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.30.1 Detailed Description

Dilate filter.

Object of this class applies a Dilate filter to *aSrc*. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1 and VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.30.2 Member Function Documentation

24.30.2.1 APEXCV_LIB_RESULT apexcv::DilateFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).
`in`	*aWindowSize*	Window Size (3).
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).

24.30.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.30.2.3 APEXCV_LIB_RESULT apexcv::DilateFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1 or VSDK_CV_16SC1).

24.30.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.31 apexcv::ErodeFilter Class Reference

Erode filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.31.1 Detailed Description

Erode filter.

Object of this class applies a Erode filter to *aSrc*. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.31.2 Member Function Documentation

24.31.2.1 APEXCV_LIB_RESULT apexcv::ErodeFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size (3).
in,out	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.31.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.31.2.3 APEXCV_LIB_RESULT apexcv::ErodeFilter::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.31.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.32 apexcv::ExtractChannel Class Reference

Channel extract class containing support for extracting a single channel from a multi-channel image.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aChannelIndex, vsdk::SUMat &aDst)

  Channel Extract function.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.32.1 Detailed Description

Channel extract class containing support for extracting a single channel from a multi-channel image.

This class is an interface for using channel extract functions on the host.

24.32.2 Member Function Documentation

24.32.2.1 APEXCV_LIB_RESULT apexcv::ExtractChannel::Initialize (vsdk::SUMat & *aSrc,* int *aChannelIndex,* vsdk::SUMat & *aDst*)

Channel Extract function.

Extracts a channel from a multiple channel image based on its index.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC2, VSDK_CV_8UC3, VSDK_CV_8UC4
`in`	*aChannelIndex*	Index of the channel to extract. Starts at 1.
`in,out`	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.32.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.32.2.3 APEXCV_LIB_RESULT apexcv::ExtractChannel::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC2, VSDK_CV_8UC3, VSDK_CV_8UC4
`in,out`	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.32.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.33 apexcv::Fast Class Reference

FAST Corner Detection.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aDst, vsdk::SUMat &aSrc, const int acThreshold, const bool acNonMaxSupp, const int acCircumference, const bool acOutIsList)

  Initializes the process parameters and allocates ACF resources.
- APEXCV_LIB_RESULT Process ()

  Launches the process for the FAST algorithm and waits for completion.
- APEXCV_LIB_RESULT ProcessNoPolling ()

  Launches the process for the FAST algorithm and immediately returns, allowing for other tasks to be run on the host. To wait for process completion call ProcessWait(). Make sure that every call to ProcessNoPolling() is paired with a call to ProcessWait().
- APEXCV_LIB_RESULT ProcessWait ()

  Waits for a FAST process previously launched with ProcessNoPolling() to finish.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aDst, vsdk::SUMat &aSrc, const int acThreshold, const bool acNms, const int acCircumference, const int acOutIsList)

  Reinitializes the ACF process graph connections. Used for reconnecting the input and output buffers. This operation is needed only if the connected Input/Outputs are changed. This operation is not needed if only the data values change, without any change of pointers or sizes.
- APEXCV_LIB_RESULT SelectApuConfiguration (ACF_APU_CFG aApuConfig, int32_t aApexId)

  SelectApuConfiguration Selects APU and APEX configuration.
- int GetNrOfFeatures ()

  Returns the number of features detected by FAST If the output is not a list the latency is very high !

24.33.1 Detailed Description

FAST Corner Detection.

apexcv::Fast is the host-ACF interface for creating, initializing, executing and releasing the FAST9 corner detection on Apex.

24.33.2 Member Function Documentation

24.33.2.1 int apexcv::Fast::GetNrOfFeatures ( )

Returns the number of features detected by FAST If the output is not a list the latency is very high !

24.33.2.2 APEXCV_LIB_RESULT apexcv::Fast::Initialize (vsdk::SUMat & *aDst,* vsdk::SUMat & *aSrc,* const int *acThreshold,* const bool *acNonMaxSupp,* const int *acCircumference,* const bool *acOutIsList*)

Initializes the process parameters and allocates ACF resources.

Parameters

out	*aDst*	Output buffer. Depending on the value of acOutIsList it can be: • an image where pixel values are: – 0 for non-corner – 1 for corner OR • a list of (x, y) pairs representing the coordinates of the detected corner pixels.
in	*aSrc*	Source image.
in	*acThreshold*	"t" threshold for the FAST algorithm.
in	*acNonMaxSupp*	enable or not non maximum suppression.
in	*acCircumference*	FAST circle circumference (e.g. 8, 12 or 16).
out	*acOutIsList*	If true, then the algorithm will output a list with the coordinates of the detected corners. If false, then the algorithm will output an image with values of 0 (non-corner) and 1 (corner).

Returns

Error code for the initialization - see apexcv_error_codes.hpp

24.33.2.3 APEXCV_LIB_RESULT apexcv::Fast::Process ( )

Launches the process for the FAST algorithm and waits for completion.

Returns

Error code for the execution - see apexcv_error_codes.hpp

24.33.2.4 APEXCV_LIB_RESULT apexcv::Fast::ProcessNoPolling ( )

Launches the process for the FAST algorithm and immediately returns, allowing for other tasks to be run on the host. To wait for process completion call ProcessWait(). Make sure that every call to ProcessNoPolling() is paired with a call to ProcessWait().

Returns

Error code for the execution - see apexcv_error_codes.hpp

24.33.2.5 APEXCV_LIB_RESULT apexcv::Fast::ProcessWait ( )

Waits for a FAST process previously launched with ProcessNoPolling() to finish.

Returns

Error code for the execution - see apexcv_error_codes.hpp

24.33.2.6 APEXCV_LIB_RESULT apexcv::Fast::ReconnectIO (vsdk::SUMat & *aDst*, vsdk::SUMat & *aSrc*, const int *acThreshold*, const bool *acNms*, const int *acCircumference*, const int *acOutIsList*)

Reinitializes the ACF process graph connections. Used for reconnecting the input and output buffers. This operation is needed only if the connected Input/Outputs are changed. This operation is not needed if only the data values change, without any change of pointers or sizes.

Returns

Error code for ACF graph process reinitialization - see apexcv_error_codes.hpp

24.33.2.7 APEXCV_LIB_RESULT apexcv::Fast::SelectApuConfiguration (ACF_APU_CFG *aApuConfig*, int32_t *aApexId*)

SelectApuConfiguration Selects APU and APEX configuration.

Returns

Error code for updating the APU configuration - see apexcv_error_codes.hpp

24.34 icp::Feature Struct Reference

ICP Feature structure. This structure is used by the ICP_FeatureDesc class to store the position (x and y), and the strength of a feature.

Public Attributes

- ICP_Point_16S position
- int16_t strength
- int16_t reserve

24.34.1 Detailed Description

ICP Feature structure. This structure is used by the ICP_FeatureDesc class to store the position (x and y), and the strength of a feature.

24.34.2 Member Data Documentation

24.34.2.1 ICP_Point_16S icp::Feature::position

The x and y location of the feature.

24.34.2.2 int16_t icp::Feature::reserve

Used for memory alignment.

24.34.2.3 int16_t icp::Feature::strength

The strength of the feature.

24.35 icp::Feature32S Struct Reference

ICP Feature32S structure. This structure is used by the ICP_FeatureDesc class to store the position (x and y), and the strength of a feature.

Public Attributes

- ICP_Point position
- int32_t strength
- int32_t reserve

24.35.1 Detailed Description

ICP Feature32S structure. This structure is used by the ICP_FeatureDesc class to store the position (x and y), and the strength of a feature.

24.35.2 Member Data Documentation

24.35.2.1 ICP_Point icp::Feature32S::position

The x and y location of the feature.

24.35.2.2 int32_t icp::Feature32S::reserve

Used for memory alignment.

24.35.2.3 int32_t icp::Feature32S::strength

The strength of the feature.

24.36 icp::Feature32SDescriptor Class Reference

ICP Feature32S Descriptor.

Public Member Functions

- Feature32SDescriptor ()

  Default constructor.
- Feature32SDescriptor (void ∗const lpData, void ∗const lpDataPhys, int32_t maxElements)

  Constructor.
- void ∗ GetDataPtr () const

  Return Data Pointer.
- void ∗ GetDataPtrPhys () const

  Returns the 'physical' Data Pointer.
- int32_t GetSpan () const

  Returns the span of a single feature.
- int32_t GetSize () const

  Returns the maximum number of features.
- int32_t GetCount () const

  Return the number of features available.
- int32_t SetCount (int32_t count)

  Set the number of features available.
- int32_t Add (int32_t x, int32_t y, int32_t strength=0)

  Add a feature.
- int32_t Remove (int32_t ind)

  Remove a feature at an index.
- Feature32S & GetFeature (int32_t ind)

  Get a feature at a specified index.
- int32_t Set (int32_t ind, int32_t x, int32_t y, int32_t strength=0)

  Set a feature.
- Feature32S & operator[] (int32_t ind)

  Operator [].
- void Init (void ∗const lpData, void ∗const lpDataPhys, int32_t maxElements)

  Initialize the descriptor.

24.36.1 Detailed Description

ICP Feature32S Descriptor.

ICP_Feature32SDesc is a container class designed to encapsulate a contiguous region of data of type ICP_Feature32S. It does not allocate or deallocate memory; it simply standardizes the representation of a contiguous memory region so it can be used by framework level services. ICP_Feature32S is a structure that contains the position of a feature and its strength.

The memory must be allocated using OAL for the number of features you want to store. i.e. If you want to be able to store 30 features:

```
void *lBuffOal = OAL_MemoryAllocFlag(sizeof(ICP_Feature32S)*30,
                  OAL_MEMORY_FLAG_ALIGN(ALIGN2_CACHELINE)|OAL_MEMORY_FLAG_CONTIGUOUS);
ICP_Feature32SDesc Desc(OAL_MemoryReturnAddress(lBuffOal, ACCESS_NCH_NB),
                     OAL_MemoryReturnAddress(lBuffOal, ACCESS_PHY),
                     30);
```

24.36.2 Constructor & Destructor Documentation

24.36.2.1 icp::Feature32SDescriptor::Feature32SDescriptor ( )

Default constructor.

24.36.2.2 icp::Feature32SDescriptor::Feature32SDescriptor (void ∗const *lpData,* void ∗const *lpDataPhys,* int32_t *maxElements*)

Constructor.

Constructor that initializes a contiguous data region. Note that the data region must be physically contiguous in memory (not just contiguous from the OS point of view).

Parameters

lpData	Pointer to contiguous data region.
lpDataPhys	Physical pointer to contiguous data region (for HW use).
maxElements	The maximum number of features.

24.36.3 Member Function Documentation

24.36.3.1 int32_t icp::Feature32SDescriptor::Add (int32_t *x,* int32_t *y,* int32_t *strength =* 0)

Add a feature.

Returns

The result of the operation (zero on success).

This adds a feature to the descriptor. This should only be used if the count of the number of features available is kept accurate. This will check the number of features against the maximum number of features, if it is not at full capacity, the new feature will be added and the count is incremented.

24.36.3.2 int32_t icp::Feature32SDescriptor::GetCount ( ) const

Return the number of features available.

Returns

The number of features currently stored

Returns the number of features currently stored in the descriptor. This value is incremented each time a feature is added with the ICP_Feature32SDesc::Add function. If the features are manually added/removed, the count can be updated using ICP_Feature32SDesc::SetCount. Only accurate if class functions are used to add/remove features, or if kept accurate by updating count.

24.36.3.3 void∗ icp::Feature32SDescriptor::GetDataPtr ( ) const

Return Data Pointer.

Returns

A void data pointer to the contiguous data region

Returns a void data pointer to the contiguous data region.

24.36.3.4 void∗ icp::Feature32SDescriptor::GetDataPtrPhys ( ) const

Returns the 'physical' Data Pointer.

Returns

A void 'physical' data pointer to the contiguous data region.

Returns a void 'physical' data pointer to the contiguous data region.

24.36.3.5 Feature32S& icp::Feature32SDescriptor::GetFeature (int32_t *ind*)

Get a feature at a specified index.

Returns

The ICP_Feature32S at the specified index

This will return a feature at index *ind*. If the index is greater than the maximum size of the descriptor, the feature at index 0 is returned.

24.36.3.6 int32_t icp::Feature32SDescriptor::GetSize ( ) const

Returns the maximum number of features.

Returns

The number of features the descriptor can hold

Returns the maximum number of features the descriptor can hold.

24.36.3.7 int32_t icp::Feature32SDescriptor::GetSpan ( ) const

Returns the span of a single feature.

Returns

The span of a feature

Returns the number of bytes a single ICP_Feature32S occupies

24.36.3.8 void icp::Feature32SDescriptor::Init (void ∗const *lpData,* void ∗const *lpDataPhys,* int32_t *maxElements*)

Initialize the descriptor.

This will initialize the descriptor with a contiguous data region. Note that the data region must be physically contiguous in memory (not just contiguous from the OS point of view).

Parameters

lpData	Pointer to contiguous data region.
lpDataPhys	Physical pointer to contiguous data region (for HW use).
maxElements	The maximum number of features.

24.36.3.9 Feature32S& icp::Feature32SDescriptor::operator[ ] (int32_t *ind*)

Operator [].

Returns

ICP_Feature32S at index

This will return a feature at the specified index. If the index is greater than the maximum size of the descriptor, the feature at index 0 is returned.

24.36.3.10 int32_t icp::Feature32SDescriptor::Remove (int32_t *ind*)

Remove a feature at an index.

Returns

The result of the operation (zero on success).

This removes a feature from the descriptor at the specified index. The remaining features will be shifted to fill the space. This should only be used if the count is kept updated.

24.36.3.11 int32_t icp::Feature32SDescriptor::Set (int32_t *ind,* int32_t *x,* int32_t *y,* int32_t *strength* = 0)

Set a feature.

Returns

The result of the operation (zero on success).

This will modify the feature at index *ind* to contain the specified position and strength. The strength will default to 0 if not specified.

24.36.3.12 int32_t icp::Feature32SDescriptor::SetCount (int32_t *count*)

Set the number of features available.

Returns

The result of the operation (zero on success).

This sets the number of features available in the descriptor. The count is limited to the range [0, Max Elements];

24.37 icp::FeatureDescriptor Class Reference

ICP Feature Descriptor.

Public Member Functions

- FeatureDescriptor ()

  Default constructor.
- FeatureDescriptor (void ∗const lpData, void ∗const lpDataPhys, int32_t maxElements)

  Constructor.
- void ∗ GetDataPtr () const

  Return Data Pointer.
- void ∗ GetDataPtrPhys () const

  Returns the 'physical' Data Pointer.
- int32_t GetSpan () const

  Returns the span of a single feature.
- int32_t GetSize () const

  Returns the maximum number of features.
- int32_t GetCount () const

  Return the number of features available.
- int32_t SetCount (int32_t count)

  Set the number of features available.
- int32_t Add (int16_t x, int16_t y, int16_t strength=0)

  Add a feature.
- int32_t Remove (int32_t ind)

  Remove a feature at an index.
- Feature & GetFeature (int32_t ind) const

  Get a feature at a specified index.
- int32_t Set (int32_t ind, int16_t x, int16_t y, int16_t strength=0)

  Set a feature.
- Feature & operator[] (int32_t ind)

  Operator [].
- void Init (void ∗const lpData, void ∗const lpDataPhys, int32_t maxElements)

  Initialize the descriptor.

24.37.1 Detailed Description

ICP Feature Descriptor.

ICP_FeatureDesc is a container class designed to encapsulate a contiguous region of data of type ICP_Feature. It does not allocate or deallocate memory; it simply standardizes the representation of a contiguous memory region so it can be used by framework level services. ICP_Feature is a structure that contains the position of a feature and its strength.

The memory must be allocated using OAL for the number of features you want to store. i.e. If you want to be able to store 30 features:

```
void *lBuffOal = OAL_MemoryAllocFlag(sizeof(ICP_Feature)*30,
                 OAL_MEMORY_FLAG_ALIGN(ALIGN2_CACHELINE)|OAL_MEMORY_FLAG_CONTIGUOUS);
ICP_FeatureDesc Desc(OAL_MemoryReturnAddress(lBuffOal, ACCESS_NCH_NB),
                     OAL_MemoryReturnAddress(lBuffOal, ACCESS_PHY),
                     30);
```

24.37.2 Constructor & Destructor Documentation

24.37.2.1 icp::FeatureDescriptor::FeatureDescriptor ( )

Default constructor.

24.37.2.2 icp::FeatureDescriptor::FeatureDescriptor (void ∗const *lpData,* void ∗const *lpDataPhys,* int32_t *maxElements*)

Constructor.

Constructor that initializes a contiguous data region. Note that the data region must be physically contiguous in memory (not just contiguous from the OS point of view).

Parameters

lpData	Pointer to contiguous data region.
lpDataPhys	Physical pointer to contiguous data region (for HW use).
maxElements	The maximum number of features.

24.37.3 Member Function Documentation

24.37.3.1 int32_t icp::FeatureDescriptor::Add (int16_t *x,* int16_t *y,* int16_t *strength* = 0)

Add a feature.

Returns

The result of the operation (zero on success).

This adds a feature to the descriptor. This should only be used if the count of the number of features available is kept accurate. This will check the number of features against the maximum number of features, if it is not at full capacity, the new feature will be added and the count is incremented.

24.37.3.2 int32_t icp::FeatureDescriptor::GetCount ( ) const

Return the number of features available.

Returns

The number of features currently stored

Returns the number of features currently stored in the descriptor. This value is incremented each time a feature is added with the ICP_FeatureDesc::Add function. If the features are manually added/removed, the count can be updated using ICP_FeatureDesc::SetCount. Only accurate if class functions are used to add/remove features, or if kept accurate by updating count.

24.37.3.3 void∗ icp::FeatureDescriptor::GetDataPtr ( ) const

Return Data Pointer.

Returns

A void data pointer to the contiguous data region

Returns a void data pointer to the contiguous data region.

24.37.3.4 void∗ icp::FeatureDescriptor::GetDataPtrPhys ( ) const

Returns the 'physical' Data Pointer.

Returns

A void 'physical' data pointer to the contiguous data region.

Returns a void 'physical' data pointer to the contiguous data region.

24.37.3.5 Feature& icp::FeatureDescriptor::GetFeature (int32_t *ind*) const

Get a feature at a specified index.

Returns

The ICP_Feature at the specified index

This will return a feature at index *ind*. If the index is greater than the maximum size of the descriptor, the feature at index 0 is returned.

24.37.3.6 int32_t icp::FeatureDescriptor::GetSize ( ) const

Returns the maximum number of features.

Returns

The number of features the descriptor can hold

Returns the maximum number of features the descriptor can hold.

24.37.3.7 int32_t icp::FeatureDescriptor::GetSpan ( ) const

Returns the span of a single feature.

Returns

The span of a feature

Returns the number of bytes a single ICP_Feature occupies

24.37.3.8 void icp::FeatureDescriptor::Init (void ∗const *lpData,* void ∗const *lpDataPhys,* int32_t *maxElements*)

Initialize the descriptor.

This will initialize the descriptor with a contiguous data region. Note that the data region must be physically contiguous in memory (not just contiguous from the OS point of view).

Parameters

lpData	Pointer to contiguous data region.
lpDataPhys	Physical pointer to contiguous data region (for HW use).
maxElements	The maximum number of features.

24.37.3.9 Feature& icp::FeatureDescriptor::operator[ ] (int32_t *ind*)

Operator [].

Returns

ICP_Feature at index

This will return a feature at the specified index. If the index is greater than the maximum size of the descriptor, the feature at index 0 is returned.

24.37.3.10 int32_t icp::FeatureDescriptor::Remove (int32_t *ind*)

Remove a feature at an index.

Returns

The result of the operation (zero on success).

This removes a feature from the descriptor at the specified index. The remaining features will be shifted to fill the space. This should only be used if the count is kept updated.

24.37.3.11 int32_t icp::FeatureDescriptor::Set (int32_t *ind,* int16_t *x,* int16_t *y,* int16_t *strength =* 0)

Set a feature.

Returns

The result of the operation (zero on success).

This will modify the feature at index *ind* to contain the specified position and strength. The strength will default to 0 if not specified.

24.37.3.12 int32_t icp::FeatureDescriptor::SetCount (int32_t *count*)

Set the number of features available.

Returns

The result of the operation (zero on success).

This sets the number of features available in the descriptor. The count is limited to the range [0, Max Elements];

24.38 apexcv::GaussianFilter Class Reference

Gaussian filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.38.1 Detailed Description

Gaussian filter.

Object of this class applies a Gaussian filter to *aSrc*. Supported window size: 3x3, 5x5, 7x7 or 9x9
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.38.2 Member Function Documentation

24.38.2.1 APEXCV_LIB_RESULT apexcv::GaussianFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size: 3, 5, 7 or 9
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.38.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.38.2.3 APEXCV_LIB_RESULT apexcv::GaussianFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.38.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ *ApexId*	ID of the APEX core used for performing the processing (0 or 1).

24.39 apexcv::GFTTCorners Class Reference

apexcv::GFTTCorners is the host-ACF interface for creating, initializing, executing the GFTT/HARRIS Corner Detector on Apex.

Public Member Functions

- int Initialize (vsdk::SUMat &aSrc, icp::FeatureDescriptor &aCorners, int aQualityLevel=4, int aMinDistance=5, int aMaxTotalCorners=4096, int aUseHarrisDetector=0, int aHarrisK=10, int aHarrisThreshold=0, int aBoxSize=7)

  Initialize the corner detector. In this release, Sobel filter size are fixed to 3. Box filter size are fixed to 7. NMS filter size are fixed to 5.
  Note: *The feature need to be run in* ***64 CUs*** *configuration.*

- int ReconnectIO (vsdk::SUMat &aSrc, icp::FeatureDescriptor &aCorners)

  Reconnect the Input and Output Buffers.
- int SetShiftValue (int aCovarianceScaleFactor, int aBoxSize)

  Set custom rescale shift value for covariance and box filter.
- int ResetShiftValue ()

  reset box_size, nms_size and covariance to default values
- int SetParameters (int aQualityLevel=4, int aMinDistance=5, int aMaxCorners=4096, int aUseHarrisDetector=0, int aHarrisK=10, int aHarrisThreshold=0)

  Set Parameters for GFTT corners detector.
- int SetMaxNumberCorners (int aMaxTotalCorners)

  Set the total maximum number of corners to be returned.
- int Process ()

  Run APEX GFTT/HARRIS Interface.
- int RetBlockWidth ()

  Returns Block Width.
- int RetBlockHeight ()

  Returns Block Height.
- int RetNumberCorners ()

  Returns Number of detected corners.
- vsdk::SUMat RetCornerImage ()

  Returns a 16-bit corner score image.
- int SelectApuConfiguration (ACF_APU_CFG aApuConfig, int32_t aApexId)

  Select Apu configuration.

24.39.1 Detailed Description

apexcv::GFTTCorners is the host-ACF interface for creating, initializing, executing the GFTT/HARRIS Corner Detector on Apex.

The interface determines strong corners on an image based on GFTT Corners Response described in [7] (default) or Haris Corners Response described in [4].
Note: The feature needs to be run in **64 CUs** configuration.

24.39.2 Member Function Documentation

24.39.2.1 int apexcv::GFTTCorners::Initialize (vsdk::SUMat & *aSrc*, icp::FeatureDescriptor & *aCorners*, int *aQualityLevel* = 4, int *aMinDistance* = 5, int *aMaxTotalCorners* = 4096, int *aUseHarrisDetector* = 0, int *aHarrisK* = 10, int *aHarrisThreshold* = 0, int *aBoxSize* = 7)

Initialize the corner detector. In this release, Sobel filter size are fixed to 3. Box filter size are fixed to 7. NMS filter size are fixed to 5.
Note: The feature need to be run in **64 CUs** configuration.

Parameters

aSrc	8-bit grayscale source image. The maximum resolution supported is 1024 pixel width with 64 CUs configuration.
aCorners	16-bit signed integer destination corner list buffer.

Parameters

aQualityLevel	Quality Level used for GFTT threshold. The parameter is applied if GFTT detector is being used (useHarrisDetector = 0). Threshold = (maxEigenvalue ∗ qualityLevel/256).
aMinDistance	Minimum distance between 2 corners. For this release, minimum distance cannot be higher than 5.
aMaxTotalCorners	Maximum number of total corners to detect. If number of corners are higher than 4096, ARM will handle corners sorting and extraction and performance will be impacted.
aUseHarrisDetector	Indicate whether to use Harris detector or GFTT detector. If useHarrisDetector = 0, GFTT detector is used. If useHarrisDetector = 1, Harris detector is used.
aHarrisK	Fixed point Harris Corner Coefficent (k). Range 0 - 255. Fixed_point_K = Floating_point_K ∗ 256. Floating point k is described in [4]. The parameter is applied if Harris detector is being used (useHarrisDetector = 1).
aHarrisThreshold	User defined Harris Threshold. Range MIN_INT16_T - MAX_INT16_T. The parameter is applied if Harris detector is being used (useHarrisDetector = 1).
aBoxSize	User defined Box Size. Range MIN_INT16_T - MAX_INT16_T.

24.39.2.2 int apexcv::GFTTCorners::Process ( )

Run APEX GFTT/HARRIS Interface.

Returns

Error code for the ACF execution (zero on success).

For each pixel, the corner score is computed (referred to as "corner response" in [4] or [7]). The score image can be returned using retCornerImage(). From the score image, the list of strong corners are returned.

24.39.2.3 int apexcv::GFTTCorners::ReconnectIO (vsdk::SUMat & *aSrc*, icp::FeatureDescriptor & *aCorners*)

Reconnect the Input and Output Buffers.

Use this to reconnect the input and output buffers. This only needs to be done if the connected Input/Outputs are changed. If only the data within (no size, or type changes), then this does not need to be called.

Parameters

aSrc	8-bit grayscale source image.
aCorners	16-bit signed integer corner list buffer.

24.39.2.4 int apexcv::GFTTCorners::ResetShiftValue ( )

reset box_size, nms_size and covariance to default values

Use this method to reset the values of box filter and nms filter size back to the default

24.39.2.5 int apexcv::GFTTCorners::RetBlockHeight ( )

Returns Block Height.

Returns the block width used in the process.

24.39.2.6 int apexcv::GFTTCorners::RetBlockWidth ( )

Returns Block Width.

Returns the block width used in the process

24.39.2.7 vsdk::SUMat apexcv::GFTTCorners::RetCornerImage ( )

Returns a 16-bit corner score image.

24.39.2.8 int apexcv::GFTTCorners::RetNumberCorners ( )

Returns Number of detected corners.

24.39.2.9 int apexcv::GFTTCorners::SelectApuConfiguration (ACF_APU_CFG *aApuConfig*, int32_t *aApexId*)

Select Apu configuration.

24.39.2.10 int apexcv::GFTTCorners::SetMaxNumberCorners (int *aMaxTotalCorners*)

Set the total maximum number of corners to be returned.

To maximize the performance, it is recomended to keep the maximum corners to be lower than 4096. If number of conrers are higher than 4096, ARM will handle corners sorting and extraction and performance will be impacted.

24.39.2.11 int apexcv::GFTTCorners::SetParameters (int *aQualityLevel* = `4`, int *aMinDistance* = `5`, int *aMaxCorners* = `4096`, int *aUseHarrisDetector* = `0`, int *aHarrisK* = `10`, int *aHarrisThreshold* = `0`)

Set Parameters for GFTT corners detector.

For this release, Sobel filter size are fixed to 3. Box filter size are fixed to 7. NMS filter size are fixed to 5.

Parameters

aQualityLevel	Quality Level used for GFTT threshold. The parameter is applied if GFTT detector is being used (useHarrisDetector = 0). Threshold = (maxEigenValue ∗ qualityLevel/256).
aMinDistance	Minimum distance between 2 corners. For this release, minimum distance cannot be higher than 5.
aMaxCorners	Maximum number of total corners to detect. If number of corners are higher than 4096, ARM will handle corners sorting and extraction and performance will be impacted.

Parameters

aUseHarrisDetector	Indicate whether to use Harris detector or GFTT detector. If useHarrisDetector = 0, GFTT detector is used. If useHarrisDetector = 1, Harris detector is used.
aHarrisK	Fixed point Harris Corner Coefficent (k). Range 0 - 255. Fixed_point_K = Floating_point_K $*$ 256. Floating point k is described in [4]. The parameter is applied if Harris detector is being used (useHarrisDetector = 1).
aHarrisThreshold	User defined Harris Threshold. Range MIN_INT16_T - MAX_INT16_T. The parameter is applied if Harris detector is being used (useHarrisDetector = 1).

24.39.2.12 int apexcv::GFTTCorners::SetShiftValue (int *aCovarianceScaleFactor,* int *aBoxSize*)

Set custom rescale shift value for covariance and box filter.

Use this method to override the default rescale shift value for covariance and box filter.

default covarianceScaleFactor is 6 default boxSize is 7 (out of 3, 5, 7)

Parameters

aCovarianceScaleFactor	Rescale shift value for covariance filter
aBoxSize	box filter size

24.40 apexcv::Histogram Class Reference

Histogram.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.40.1 Detailed Description

Histogram.

Object of this class computes the histogram of the image.
Output dimensions are 256x1 VSDK_CV_32SC1.

Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.40.2 Member Function Documentation

24.40.2.1 APEXCV_LIB_RESULT apexcv::Histogram::Initialize (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer 256x1 (VSDK_CV_32SC1).

24.40.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.40.2.3 APEXCV_LIB_RESULT apexcv::Histogram::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer 256x1 (VSDK_CV_32SC1).

24.40.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.41 apexcv::HistogramEqualization Class Reference

Host-ACF interface for histogram equalization.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialize the process.
- APEXCV_LIB_RESULT Process ()

  Execute the process.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect input and output buffers.

24.41.1 Detailed Description

Host-ACF interface for histogram equalization.

This template class is an interface for creating, initializing, processing and releasing the APU implementation of histogram on the host.

24.41.2 Member Function Documentation

24.41.2.1 APEXCV_LIB_RESULT apexcv::HistogramEqualization::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Initialize the process.

Returns

Error code for the initialization (zero on success).

We initialize the process on the host by initializing the ACF process

Parameters

aSrc	8-bit grayscale source image.
aDst	8-bit grayscale destination image.

24.41.2.2 APEXCV_LIB_RESULT apexcv::HistogramEqualization::Process ( )

Execute the process.

Returns

Error code for the execution (zero on success).

24.41.2.3 APEXCV_LIB_RESULT apexcv::HistogramEqualization::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect input and output buffers.

Returns

Error code for the execution (zero on success).

Parameters

aSrc	8-bit grayscale source image.
aDst	8-bit grayscale destination image.

24.42 apexcv::Hog Class Reference

Apex HOG class.

Classes

- struct Config

  The HOG parameters for the linear SVM classifier.
- class SVM

  Class to compute SVM detector from Hog blocks.

Public Types

Public Member Functions

- APEXCV_LIB_RESULT SetConfig (const Config &aConfig)

  Sets the configuration for the Hog class, used only if in full detector mode.
- void GetConfig (Config &aConfig) const

Returns the HOG parameters for the linear SVM classifier.

- APEXCV_LIB_RESULT Initialize (const vsdk::SUMat &aSrc, const vsdk::SUMat &aSVM, vsdk::SUMat &aDst)

  Initialization of the Hog object for detection mode.
- APEXCV_LIB_RESULT Initialize (const vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialization of the Hog object for block mode.
- APEXCV_LIB_RESULT ReconnectIO (const vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

Static Public Member Functions

- static APEXCV_LIB_RESULT GetDescriptors (const Config &aConfig, const vsdk::SUMat &aBlocksHist, vsdk::↩ SUMat &aDescriptor)

  Concatenate blocks to form HOG descriptor.

24.42.1 Detailed Description

Apex HOG class.

apexcv::Hog is the host interface for HOG detection running on Apexcore. HOG Object Detector

The process takes an 8-bit grayscale image and computes the 8-bin normalized block histogram every 4x4 pixels. If SVM decision function is provided in the initialization, it multiplies the blocks histograms with SVM weights for each detection window and returns the scores for the object detection. Otherwise it returns the computed block histograms.

24.42.2 Member Enumeration Documentation

24.42.2.1 enum **apexcv::Hog::SVMTransformMode** `[strong]`

Specifies how the weights of SVM decision function will be applied in the multiplication with block histograms of the detection window.

Enumerator

NONE Apply SVM as it is

HORIZONTAL_SYMMETRY Apply SVM once as it is and once mirrored horizontally

HORIZONTAL_MIRRORING Apply SVM with horizontally mirrored weights

24.42.3 Member Function Documentation

24.42.3.1 void apexcv::Hog::GetConfig (**Config &** ***aConfig***) const

Returns the HOG parameters for the linear SVM classifier.

24.42.3.2 static APEXCV_LIB_RESULT apexcv::Hog::GetDescriptors (const Config & *aConfig*, const vsdk::SUMat & *aBlocksHist*, vsdk::SUMat & *aDescriptor*) [static]

Concatenate blocks to form HOG descriptor.

Returns

Error code for the ACF execution (APEXCV_SUCCESS on success).

The HOG descriptor is a column-wise concatenation of the block histograms in a detection window. If the input image size is not an integer mutiple of the window size and stride, partial blocks at the right and bottom edges are not computed. The length-Nd descriptors are packed as an 8-bit unsigned Nd-channel image. For a (WinxHin) sized image and wxh sized block window the output size is: (Wout, Hout) = (floor((Win-w)/4)∗Nd, floor((Hin-h)/4)) Note that the memory requirement of the output descriptor image is large: Nd/16 = 64 times the input image.

Parameters

in	*aConfig*	HOG configuration
in	*aBlocksHist*	Blocks histograms, output of Hog::Process(), VSDK_CV_8UC1.
out	*aDescriptor*	HOG descriptor, populated on successful return, VSDK_CV_8UC1.

24.42.3.3 APEXCV_LIB_RESULT apexcv::Hog::Initialize (const vsdk::SUMat & *aSrc*, const vsdk::SUMat & *aSVM*, vsdk::SUMat & *aDst*)

Initialization of the Hog object for detection mode.

Returns

Error code for ACF process initialization, APEXCV_SUCCESS on success.

Prepares the object for execution. Initialization is done at this stage. For the size of the arguments, please refer to ReconnectIO()

Parameters

in	*aSrc*	8-bit grayscale source image, VSDK_CV_8UC1.
in	*aSVM*	Vector of descriptorSize + 1 elements of type double, VSDK_CV_64FC1.
in,out	*aDst*	Scores from HOG object detection, VSDK_CV_16SC1.

24.42.3.4 APEXCV_LIB_RESULT apexcv::Hog::Initialize (const vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Initialization of the Hog object for block mode.

Returns

Error code for ACF process initialization, APEXCV_SUCCESS on success.

Prepares the object for execution. Initialization is done at this stage. For the size of the arguments, please refer to ReconnectIO()

Parameters

in	*aSrc*	8-bit grayscale source image, VSDK_CV_8UC1.
in,out	*aDst*	Blocks histograms, VSDK_CV_8UC1.

24.42.3.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.42.3.6 APEXCV_LIB_RESULT apexcv::Hog::ReconnectIO (const vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO.

Use this to reconnect the input and output buffers and allocate the output. This only needs to be done if the connected Input/Outputs are changed. If only the data within changes (no size, data pointer, or type changes), then this does not need to be called. This function will allocation aDst if it has not been pre-allocated with the right dimensions.

The width of the source image should be multiple 64. The height of the source image should be multiple 4. Detection mode: The output is the scores matrix of VSDK_CV_16SC1. For a (WinxHin) sized image and wxh sized detection window the output size is: (Wout, Hout) = (floor((Win-w)/4) + 1, floor((Hin-h)/4) + 1) Block mode: The output is the block histogram matrix of VSDK_CV_8UC1. For a (WinxHin) sized image and window stride 4x4 the output size is: (Wout, Hout) = (floor((Win-4)/4)∗8, floor((Hin-4)/4))

Parameters

in	*aSrc*	8-bit grayscale source image, VSDK_CV_8UC1.
in,out	*aDst*	Detect mode: Scores from HOG object detection, VSDK_CV_16SC1, Block mode: Blocks histograms, VSDK_CV_8UC1.

24.42.3.7 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.42.3.8 APEXCV_LIB_RESULT apexcv::Hog::SetConfig (const Config & *aConfig*)

Sets the configuration for the Hog class, used only if in full detector mode.

Set the HOG parameters for the linear SVM classifier. It is used when changing from default setting only. It should be called before Initialize().

The size of the detection window should not be bigger than 128x128 pixels. If some dimension is not integer multiple of the HOG block size, it is rounded down per the block size.

24.43 apexcv::HoughLineDetector Class Reference

Apex Hough Line Detector.

Classes

- struct Line

  Line data structure associated with the Hough Line Detector.

Public Types

- typedef uint32_t PackedLine

  Packed line format for Hough Line Detector.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aImage, int aPixelThreshold=127, int aAccThreshold=100, int aThetaCount=180, float ∗apTheta=NULL, int aNonMaxSupp=(NMS_RHO|NMS_THETA))

  Initialize parameters and allocate resources.
- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aImage, int aPixelThreshold, int aAccThreshold, int aTheta↩Count, double aThetaStart, double aThetaStep, int aNonMaxSupp=(NMS_RHO|NMS_THETA))

  Initialize parameters and allocate resources.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aImage, int aPixelThreshold, int aAccThreshold, int a↩ThetaCount, float ∗apTheta, int aNonMaxSupp=(NMS_RHO|NMS_THETA))

  Reinitializes the ACF process graf connections.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aImage, int aPixelThreshold, int aAccThreshold, int a↩ThetaCount, double aThetaStart, double aThetaStep, int aNonMaxSupp=(NMS_RHO|NMS_THETA))

  Reinitializes the ACF process graf connections.
- APEXCV_LIB_RESULT SelectApuConfiguration (ACF_APU_CFG aApuConfig=ACF_APU_CFG__DEFAULT, int32_t aApexId=0)

  APEX hardware configuration.
- APEXCV_LIB_RESULT Release ()

Release resources and resets parameters.

- APEXCV_LIB_RESULT SetPixelThreshold (int aPixelThreshold)

  Set the pixel threshold for line detection.

- APEXCV_LIB_RESULT SetAccumThreshold (int aAccThreshold)

  Set the Hough accumulator threshold for line detection.

- APEXCV_LIB_RESULT SetTheta (int aThetaCount, float $*$apThetaData=NULL, int aNonMaxSupp=(NMS_RHO|NMS_THETA))

  Specify the number of angles to be detected.

- APEXCV_LIB_RESULT SetTheta (int aThetaCount, double aThetaStart, double aThetaStep, int aNonMaxSupp)

  Specify the number of angles to be detected.

- APEXCV_LIB_RESULT Process ()

  Detect lines on the image. Lines are detected based on the parameters specified by Initialize, SetPixelThreshold, SetAccumThreshold and SetTheta. The number of detected lines is accessed by GetLineCount. Packed lines are accessed by GetPackedLineData. Line coordinates are accessed by Line.

- int GetRhoCount ()

  Get the number of rho values. This is the size of the Hough accumulator in CMEM It is given by $round\left(\sqrt{w^2+h^2}\right)+1$, where (w,h) is the image width and height.

- int GetRhoStart ()

  Get the starting rho value for the Hough accumulator. This is the offset that must be applied to rhoID "rho index" to obtain the true rho value. That is, rho = rhoID + rhoStart.

- int GetThetaCount ()

  Get the number of angles to detect.

- float $*$ GetThetaData ()

  Get a pointer to angles to detect.

- int GetNmsFlag ()

  Get the non-maxima suppression flag.

- int GetLineCount ()

  Get the number of detected lines.

- PackedLine $*$ GetPackedLineData ()

  Get a pointer to packed line data.

- Line GetLine (int aIndex)

  Get the line coordinates for a detected line.

- Line GetLine (PackedLine aPackedLine)

  Get the line coordinates for a detected line.

- APEXCV_LIB_RESULT CheckParameters (int aImageCols, int aImageRows, int aPixelThreshold, int aAccThreshold, int aThetaCount)

  Check the validity of Hough initialization parameters.

Static Public Member Functions

- static int GetAccumulator (PackedLine aLine)

  Get the Hough accumulator value from the PackedLine.

- static int GetRhoId (PackedLine aLine)

  Get the rho index from the PackedLine.

- static int GetThetaId (PackedLine aLine)

  Get the angle index from the PackedLine.

Static Public Attributes

- static int mCuCnt

  The number of CUs.
- static int mMaxRhoCnt

  Maximum range of rho values.
- static int mMaxLinesPerIt

  Max number of detected lines.
- static double mcDeg2Rad

  Conv. from degrees to radians.
- static double mcRad2Deg

  Conv, from radians to degrees.
- static double mcPi

  Universal constant.

24.43.1 Detailed Description

Apex Hough Line Detector.

apexcv::HoughLineDetector is the host-ACF interface for creating, initializing, executing and releasing the Hough Line Detector on Apex.

24.43.2 Member Typedef Documentation

24.43.2.1 typedef uint32_t apexcv::HoughLineDetector::PackedLine

Packed line format for Hough Line Detector.

Detected lines are stored in 32 bits. The first 8 bits are the angle index. The next 12 bits are the rho index. The last 12 bits are the Hough accumulator value.

24.43.3 Member Enumeration Documentation

24.43.3.1 enum apexcv::HoughLineDetector::NonMaxSupFlag

Non-maxima suppression flag. To disable non-maxima suppression (nms), use NMS_NONE. To use nms on rho only, use NMS_RHO. To use nms on theta only, use NMS_THETA. To use nms on both, use NMS_RHO | NMS_THETA.

Enumerator

NMS_NONE No non-maxima suppression.

NMS_RHO Non-maxima suppression on rho. Since the rho step is 1 pixel, this flag should always be used.

NMS_THETA Non-maxima suppression on theta. This flag should be used when the angle resolution small.

24.43.3.2 enum apexcv::HoughLineDetector::Process

The available Hough ACF processes.

Enumerator

PROC_NONE No ACF process.

PROC_40X1 ACF process with block size 40x1.

PROC_20X2 ACF process with block size 20x2.

PROC_10X4 ACF process with block size 10x4.

PROC_6X4 ACF process with block size 6x4.

24.43.4 Member Function Documentation

24.43.4.1 APEXCV_LIB_RESULT apexcv::HoughLineDetector::CheckParameters (int *aImageCols,* int *aImageRows,* int *aPixelThreshold,* int *aAccThreshold,* int *aThetaCount*)

Check the validity of Hough initialization parameters.

Parameters

in	*aImageCols*	Image columns
in	*aImageRows*	Image rows
in	*aPixelThreshold*	Threshold to qualify a pixel for the Accm
in	*aAccThreshold*	Min number of collinear pixels for a line
in	*aThetaCount*	Number of angles to detect

Returns

Error code for initialization (zero on success).

24.43.4.2 static int apexcv::HoughLineDetector::GetAccumulator (PackedLine *aLine*) [static]

Get the Hough accumulator value from the PackedLine.

Parameters

in	*aLine*	Line to get information for

Returns

Hough accumulator value. The number of pixels inside a given line

24.43.4.3 Line apexcv::HoughLineDetector::GetLine (int *aIndex*)

Get the line coordinates for a detected line.

Parameters

in	*aIndex*	Index of the detected line. This value must be within [0, GetLineCount]

Returns

The line coordinates for a detected line.

24.43.4.4 Line apexcv::HoughLineDetector::GetLine (PackedLine *aPackedLine*)

Get the line coordinates for a detected line.

Parameters

in	*aPackedLine*	Packed info about the detected line

Returns

The line coordinates for a detected line.

24.43.4.5 int apexcv::HoughLineDetector::GetLineCount ( )

Get the number of detected lines.

Returns

The number of detected lines.

24.43.4.6 int apexcv::HoughLineDetector::GetNmsFlag ( )

Get the non-maxima suppression flag.

Returns

The non-maxima suppression flag.

24.43.4.7 PackedLine∗ apexcv::HoughLineDetector::GetPackedLineData ( )

Get a pointer to packed line data.

Returns

A pointer to packed line data.

24.43.4.8 int apexcv::HoughLineDetector::GetRhoCount ( )

Get the number of rho values. This is the size of the Hough accumulator in CMEM It is given by $\mathrm{round}\left(\sqrt{w^2+h^2}\right)+1$, where (w,h) is the image width and height.

Returns

The number of rho values.

24.43.4.9 static int apexcv::HoughLineDetector::GetRhoId (PackedLine *aLine*) [static]

Get the rho index from the PackedLine.

Parameters

in	*aLine*	Line to get information for

Returns

The rho index. The rho index $r >= 0$ is the index of Hough transform for the line.

24.43.4.10 int apexcv::HoughLineDetector::GetRhoStart ( )

Get the starting rho value for the Hough accumulator. This is the offset that must be applied to rhoID "rho index" to obtain the true rho value. That is, rho = rhoID + rhoStart.

Returns

The starting rho value for the Hough accumulator.

24.43.4.11 int apexcv::HoughLineDetector::GetThetaCount ( )

Get the number of angles to detect.

Returns

The number of angles to detect.

24.43.4.12 float$*$ apexcv::HoughLineDetector::GetThetaData ( )

Get a pointer to angles to detect.

Returns

A pointer to angles to detect.

24.43.4.13 static int apexcv::HoughLineDetector::GetThetaId (PackedLine *aLine*) [static]

Get the angle index from the PackedLine.

Parameters

in	*aLine*	Line to get information for

Returns

The angle index. The angle index i in [0, 255] corresponds to i^{th} angle specified by SetTheta.

24.43.4.14 APEXCV_LIB_RESULT apexcv::HoughLineDetector::Initialize (vsdk::SUMat & *aImage*, int *aPixelThreshold* = 127, int *aAccThreshold* = 100, int *aThetaCount* = 180, float ∗ *apTheta* = NULL, int *aNonMaxSupp* = (NMS_RHO|NMS_THETA))

Initialize parameters and allocate resources.

Parameters

in	*aImage*	Input image
in	*aPixelThreshold*	Threshold to qualify as pixel from Accm
in	*aAccThreshold*	Hough accumulator threshold. Min number of collinear pixels from a line
in	*aThetaCount*	Number of angles to detect
in	*apTheta*	Angles to detect
in	*aNonMaxSupp*	Non-maxima suppression flag

Returns

Please check apexcv_error_codes.hpp

24.43.4.15 APEXCV_LIB_RESULT apexcv::HoughLineDetector::Initialize (vsdk::SUMat & *aImage*, int *aPixelThreshold*, int *aAccThreshold*, int *aThetaCount*, double *aThetaStart*, double *aThetaStep*, int *aNonMaxSupp* = (NMS_RHO|NMS_THETA))

Initialize parameters and allocate resources.

Parameters

in	*aImage*	Input image
in	*aPixelThreshold*	Threshold to qualify as pixel from Accm
in	*aAccThreshold*	Hough accumulator threshold. Min number of collinear pixels from a line
in	*aThetaCount*	Number of angles to detect
in	*aThetaStart*	Starting angle for detection
in	*aThetaStep*	Incremental angle step
in	*aNonMaxSupp*	Non-maxima suppression flag

Returns

Please check apexcv_error_codes.hpp

24.43.4.16 APEXCV_LIB_RESULT apexcv::HoughLineDetector::Process ( )

Detect lines on the image. Lines are detected based on the parameters specified by Initialize, SetPixelThreshold, SetAccumThreshold and SetTheta. The number of detected lines is accessed by GetLineCount. Packed lines are accessed by GetPackedLineData. Line coordinates are accessed by Line.

Returns

Please check apexcv_error_codes.hpp

24.43.4.17 APEXCV_LIB_RESULT apexcv::HoughLineDetector::ReconnectIO (vsdk::SUMat & *aImage,* int *aPixelThreshold,* int *aAccThreshold,* int *aThetaCount,* float ∗ *apTheta,* int *aNonMaxSupp =* (NMS_RHO|NMS_THETA))

Reinitializes the ACF process graf connections.

Parameters

in	*aImage*	Input image
in	*aPixelThreshold*	Threshold to qualify as pixel from Accm
in	*aAccThreshold*	Hough accumulator threshold. Min number of collinear pixels from a line
in	*aThetaCount*	Number of angles to detect
in	*apTheta*	Angles to detect
in	*aNonMaxSupp*	Non-maxima suppression flag

Returns

Please check apexcv_error_codes.hpp

24.43.4.18 APEXCV_LIB_RESULT apexcv::HoughLineDetector::ReconnectIO (vsdk::SUMat & *aImage,* int *aPixelThreshold,* int *aAccThreshold,* int *aThetaCount,* double *aThetaStart,* double *aThetaStep,* int *aNonMaxSupp =* (NMS_RHO|NMS_THETA))

Reinitializes the ACF process graf connections.

Parameters

in	*aImage*	Input image
in	*aPixelThreshold*	Threshold to qualify as pixel from Accm
in	*aAccThreshold*	Hough accumulator threshold. Min number of collinear pixels from a line
in	*aThetaCount*	Number of angles to detect
in	*aThetaStart*	Starting angle for detection
in	*aThetaStep*	Incremental angle step
in	*aNonMaxSupp*	Non-maxima suppression flag

Returns

Please check apexcv_error_codes.hpp

24.43.4.19 APEXCV_LIB_RESULT apexcv::HoughLineDetector::Release ( )

Release resources and resets parameters.

Returns

Please check apexcv_error_codes.hpp

24.43.4.20 APEXCV_LIB_RESULT apexcv::HoughLineDetector::SelectApuConfiguration (ACF_APU_CFG *aApuConfig =* ACF_APU_CFG__DEFAULT, int32_t *aApexId =* 0)

APEX hardware configuration.

Parameters

in	*aApexId*	Apex id where the code will execute
in	*aApuConfig*	Apu CU size

Returns

Please check apexcv_error_codes.hpp

24.43.4.21 APEXCV_LIB_RESULT apexcv::HoughLineDetector::SetAccumThreshold (int *aAccThreshold*)

Set the Hough accumulator threshold for line detection.

Parameters

in	*aAccThreshold*	Minimum number of collinear pixels needed for line detection

Returns

Error code for initialization (zero on success).

24.43.4.22 APEXCV_LIB_RESULT apexcv::HoughLineDetector::SetPixelThreshold (int *aPixelThreshold*)

Set the pixel threshold for line detection.

Parameters

in	*aPixelThreshold*	Pixel greather than aPixelThreshold are used

Returns

Error code for initialization (zero on success).

24.43.4.23 APEXCV_LIB_RESULT apexcv::HoughLineDetector::SetTheta (int *aThetaCount,* float ∗ *apThetaData =* NULL, int *aNonMaxSupp =* (**NMS_RHO**|**NMS_THETA**))

Specify the number of angles to be detected.

Parameters

in	*aThetaCount*	Number of angles to detect
in	*apThetaData*	Angles to detect. If the pointer is null, the full range of angles is equally partitioned by aThetaCount
in	*aNonMaxSupp*	Non maximum supression flag

Returns

Error code for initialization (zero on success).

24.43.4.24 APEXCV_LIB_RESULT apexcv::HoughLineDetector::SetTheta (int *aThetaCount,* double *aThetaStart,* double *aThetaStep,* int *aNonMaxSupp*)

Specify the number of angles to be detected.

Parameters

in	*aThetaCount*	Number of angles to detect
in	*aThetaStart*	Starting angle for detection
in	*aThetaStep*	Incremental angle step
in	*aNonMaxSupp*	Non maximum supression flag

Returns

Error code for initialization (zero on success).

24.44 apexcv::InsertChannel Class Reference

Channel insert class containing support for inserting a single channel in a multi-channel image.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aChannelIndex, vsdk::SUMat &aDst)

  Channel Insert function.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.44.1 Detailed Description

Channel insert class containing support for inserting a single channel in a multi-channel image.

This class is an interface for using channel insert functions on the host.

24.44.2 Member Function Documentation

24.44.2.1 APEXCV_LIB_RESULT apexcv::InsertChannel::Initialize (vsdk::SUMat & *aSrc,* int *aChannelIndex,* vsdk::SUMat & *aDst*)

Channel Insert function.

Inserts a channel from a multiple channel image based on its index.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC2, VSDK_CV_8UC3, VSDK_CV_8UC4
`in`	*aChannelIndex*	Index of the channel to insert. Starts at 1.
`in,out`	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.44.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.44.2.3 APEXCV_LIB_RESULT apexcv::InsertChannel::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC2, VSDK_CV_8UC3, VSDK_CV_8UC4
`in,out`	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.44.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.45 apexcv::IntegralImage Class Reference

Integral Image value.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.45.1 Detailed Description

Integral Image value.

Object of this class computes the sum of the pixel values located above and to the left of a given pixel.
Output dimensions are same as input.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.45.2 Member Function Documentation

24.45.2.1 APEXCV_LIB_RESULT apexcv::IntegralImage::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_32SC1).

24.45.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.45.2.3 APEXCV_LIB_RESULT apexcv::IntegralImage::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_32SC1).

24.45.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.46 apexcv::InterpolationBicubicGrayscale Class Reference

Bicubic Grayscale Interpolation.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aOffsetX, vsdk::SUMat &aOffsetY, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aOffsetX, vsdk::SUMat &aOffsetY, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.46.1 Detailed Description

Bicubic Grayscale Interpolation.

Object of this class computes the horizontal cubic interpolation, followed by the vertical cubic interpolation on 4x4 patches.
Output dimensions are same as input dimensions.
Supported input type: VSDK_CV_8UC1, output is of identical type VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.46.2 Member Function Documentation

24.46.2.1 APEXCV_LIB_RESULT apexcv::InterpolationBicubicGrayscale::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aOffsetX,* vsdk::SUMat & *aOffsetY,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in	*aOffsetX*	Delta image buffer (VSDK_CV_8UC1).
in	*aOffsetY*	Delta image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.46.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.46.2.3 APEXCV_LIB_RESULT apexcv::InterpolationBicubicGrayscale::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aOffsetX,* vsdk::SUMat & *aOffsetY,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in`	*aOffsetX*	Delta image buffer (VSDK_CV_8UC1).
`in`	*aOffsetY*	Delta image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.46.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ *ApexId*	ID of the APEX core used for performing the processing (0 or 1).

24.47 apexcv::InterpolationBilinearGrayscale Class Reference

Bilinear Grayscale Interpolation.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDelta, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDelta, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.47.1 Detailed Description

Bilinear Grayscale Interpolation.

Object of this class computes the horizontal linear interpolation, followed by the vertical linear interpolation.
Dst'(x,y) = Src(x,y) + ((Src(x+1,y) - Src(x,y)) ∗ Delta[0](x,y) + 128)/256
Dst(x,y) = Dst'(x,y) + ((Dst'(x,y+1) - Dst'(x,y)) ∗ Delta[1](x,y) + 128)/256
Output dimensions are same as input dimensions.
Supported input type: VSDK_CV_8UC1, output is of identical type VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.47.2 Member Function Documentation

24.47.2.1 APEXCV_LIB_RESULT apexcv::InterpolationBilinearGrayscale::Initialize (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDelta*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in`	*aDelta*	Delta image buffer (VSDK_CV_8UC2).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.47.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.47.2.3 APEXCV_LIB_RESULT apexcv::InterpolationBilinearGrayscale::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDelta*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in	*aDelta*	Delta image buffer (VSDK_CV_8UC2).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.47.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.48 apexcv::InterpolationLinearGrayscale Class Reference

Linear Grayscale Interpolation.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDeltaX, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDeltaX, vsdk::SUMat &aDst)

  Reconnect IO (optional).

- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.48.1 Detailed Description

Linear Grayscale Interpolation.

Object of this class computes the horizontal linear interpolation between pixels.
Output dimensions are same as input dimensions.
Dst(x,y) = Src(x,y) + ((Src(x+1,y) - Src(x,y)) ∗ Delta(x,y) + 128)/256
Supported input type: VSDK_CV_8UC1, output is of identical type VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.48.2 Member Function Documentation

24.48.2.1 APEXCV_LIB_RESULT apexcv::InterpolationLinearGrayscale::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDeltaX,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in	*aDeltaX*	Delta image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.48.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.48.2.3 APEXCV_LIB_RESULT apexcv::InterpolationLinearGrayscale::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDeltaX,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in	*aDeltaX*	Delta image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.48.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a← ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.49 apexcv::LaplacianPyramid Class Reference

Pyramid creation class.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDstPyramid, vsdk::SUMat &aDstAux, bool aIsLastLevel)

  Initializes the class.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDstPyramid, vsdk::SUMat &aDstAux)

  Connects the input/outputs to the process.
- APEXCV_LIB_RESULT Process ()

  Run the pyramid creation process.
- APEXCV_LIB_RESULT SelectApexCore (int32_t aApexId)

  SelectApexCore.

24.49.1 Detailed Description

Pyramid creation class.

This class is an interface for using the pyramid creation algorithm.

24.49.2 Member Function Documentation

24.49.2.1 APEXCV_LIB_RESULT apexcv::LaplacianPyramid::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDstPyramid,* vsdk::SUMat & *aDstAux,* bool *aIsLastLevel*)

Initializes the class.

Connects the buffers to the process.

Parameters

aSrc	Unsigned 8-bit Source memory buffer.
aDstPyramid	Signed 16-bit Destination memory buffer for pyramid output.
aDstAux	Auxiliary destination memory buffer for storing the input for the next pyramid level or reconstruction image, depending on aIsLastLevel parameter. Data is Unsigned 8-bit for aIsLastLevel == FALSE and Signed 16-bit for aIsLastLevel == TRUE.
aIsLastLevel	Chooses between returning the reconstruction image or the input for the next pyramid level.

24.49.2.2 APEXCV_LIB_RESULT apexcv::LaplacianPyramid::Process ( )

Run the pyramid creation process.

The Laplacian output for the current pyramid level is computed. Depending on how the class was configured using isLastLevel parameter when initialize() was called, the second buffer will contain the information needed to compute the next pyramid level (isLastLevel==FALSE) or the image that can be used for laplacian image reconstruction (isLastLevel==TRUE). Supported datatypes are:

- unsigned 8 bit to signed 16 bit & signed 16 bit for isLastLevel==TRUE
- unsigned 8 bit to signed 16 bit & unsigned 8 bit for isLastLevel==FALSE

Returns

Error code (zero on success).

24.49.2.3 APEXCV_LIB_RESULT apexcv::LaplacianPyramid::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDstPyramid,* vsdk::SUMat & *aDstAux*)

Connects the input/outputs to the process.

Use this to reconnect the Input and Output Buffers. This only needs to be done if the connected Input/Outputs are changed. If only the data within (no size, or type changes), then this does not need to be called.

Parameters

aSrc	Unsigned 8-bit Source memory buffer.
aDstPyramid	Signed 16-bit Destination memory buffer for pyramid output.
aDstAux	Auxiliary destination memory buffer for storing the input for the next pyramid level or reconstruction image, depending on isLastLevel parameter. Data is Unsigned 8-bit for isLastLevel == FALSE and Signed 16-bit for isLastLevel == TRUE.

24.49.2.4 APEXCV_LIB_RESULT apexcv::LaplacianPyramid::SelectApexCore (int32_t *aApexId*)

SelectApexCore.

Returns

Error code for the ACF execution (zero on success).

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed per frame base.

Parameters

a↩ ApexId	The ID of the desired APEX (e.g if there are 2 APEXs, valid values for lApexId would be 0 and 1).

24.50 apexcv::Lbp Class Reference

LBP.

Public Member Functions

- APEXCV_LIB_RESULT InitializeTrain (vsdk::SUMat &aSrc, int aSrcWidth, int aSrcHeight, int aSrcNum, vsdk::↩ SUMat &aDescriptor)

  Initializes train process: connects the buffer to the process port, and allocates/initializes any internal buffers.
- APEXCV_LIB_RESULT InitializePredict (vsdk::SUMat &aModel, int aModelNum, vsdk::SUMat &aSrc, int aSrc↩ Width, int aSrcHeight, vsdk::SUMat &aDescriptor, vsdk::SUMat &aClosestID, vsdk::SUMat &aDistance)

  Initializes predict process: connects the buffer to the process port, and allocates/initializes any internal buffers.
- APEXCV_LIB_RESULT ReconnectTrainIO (vsdk::SUMat &aSrc, int aSrcWidth, int aSrcHeight, int aSrcNum, vsdk::SUMat &aDescriptor)

  Reconnects the input/outputs to train process.
- APEXCV_LIB_RESULT ReconnectPredictIO (vsdk::SUMat &aModel, int aModelNum, vsdk::SUMat &aSrc, int aSrcWidth, int aSrcHeight, vsdk::SUMat &aDescriptor, vsdk::SUMat &aClosestID, vsdk::SUMat &aDistance)

  Reconnects the input/outputs to predict process.
- APEXCV_LIB_RESULT ProcessTrain ()

  Runs APEX-LBP train process to extract LBP descriptor.
- APEXCV_LIB_RESULT ProcessPredict ()

  Runs APEX-LBP predict process to compare test descriptor to model descriptor.

- APEXCV_LIB_RESULT ProcessPredictExtract ()

  Deprecated. Method will be removed. TODO Remove method.
- APEXCV_LIB_RESULT ProcessPredictCompare ()

  Deprecated. Method will be removed. TODO Remove method.
- APEXCV_LIB_RESULT SelectApexCore (int32_t aApexId)

  SelectApexCore.

24.50.1 Detailed Description

LBP.

apexcv::Lbp is the host-ACF interface for creating, initializing, executing and releasing the APU implementation of Lbp on APEX.

24.50.2 Member Function Documentation

24.50.2.1 APEXCV_LIB_RESULT apexcv::Lbp::InitializePredict (vsdk::SUMat & *aModel,* int *aModelNum,* vsdk::SUMat & *aSrc,* int *aSrcWidth,* int *aSrcHeight,* vsdk::SUMat & *aDescriptor,* vsdk::SUMat & *aClosestID,* vsdk::SUMat & *aDistance*)

Initializes predict process: connects the buffer to the process port, and allocates/initializes any internal buffers.

Parameters

in	*aModel*	Source model descriptor buffer. Datatype is VSDK_CV_8UC1.
in	*aModelNum*	Model descriptor count.
in	*aSrc*	Source test image buffer. Datatype is VSDK_CV_8UC1.
in	*aSrcWidth*	Source image width.
in	*aSrcHeight*	Source image height.
out	*aDescriptor*	Output test descriptor.
out	*aClosestID*	Output closest ID buffer. Datatype is VSDK_CV_16UC1.
out	*aDistance*	Output histogram distance buffer. Datatype is VSDK_CV_32SC1. The buffer width must be 32, and its height should be (modelNum+32-1)/32.

Returns

Error code for the initialization - see apexcv_error_codes.hpp

24.50.2.2 APEXCV_LIB_RESULT apexcv::Lbp::InitializeTrain (vsdk::SUMat & *aSrc,* int *aSrcWidth,* int *aSrcHeight,* int *aSrcNum,* vsdk::SUMat & *aDescriptor*)

Initializes train process: connects the buffer to the process port, and allocates/initializes any internal buffers.

Parameters

in	*aSrc*	Source memory buffer. Datatype is VSDK_CV_8UC1.
in	*aSrcWidth*	Source image width.
in	*aSrcHeight*	Source image height.
in	*aSrcNum*	Number of source image.

Parameters

out	*aDescriptor*	Destination memory buffer. Datatype is VSDK_CV_8UC1.

Returns

Error code for the initialization - see apexcv_error_codes.hpp

24.50.2.3 APEXCV_LIB_RESULT apexcv::Lbp::ProcessPredict ( )

Runs APEX-LBP predict process to compare test descriptor to model descriptor.

Computes LBP descriptor for test image, then compares test descriptor to model descriptor to find the closest descriptor.

Supported datatypes are: INPUT: unsigned 8 bit image and unsigned 8 bit mode descriptor OUTPUT: unsigned 8 bit test descriptor, unsigned 16 bit closest ID and signed 32 bit distance

Returns

Error code for the execution - see apexcv_error_codes.hpp

24.50.2.4 APEXCV_LIB_RESULT apexcv::Lbp::ProcessPredictCompare ( )

Deprecated. Method will be removed. TODO Remove method.

Returns

Error code for the execution - see apexcv_error_codes.hpp

24.50.2.5 APEXCV_LIB_RESULT apexcv::Lbp::ProcessPredictExtract ( )

Deprecated. Method will be removed. TODO Remove method.

Returns

Error code for the execution - see apexcv_error_codes.hpp

24.50.2.6 APEXCV_LIB_RESULT apexcv::Lbp::ProcessTrain ( )

Runs APEX-LBP train process to extract LBP descriptor.

Computes LBP descriptor for each grid cell.

Supported datatypes are:

- unsigned 8 bit image to unsigned 8 bit descriptor

Returns

Error code for the execution - see apexcv_error_codes.hpp

24.50.2.7 APEXCV_LIB_RESULT apexcv::Lbp::ReconnectPredictIO (vsdk::SUMat & *aModel*, int *aModelNum*, vsdk::SUMat & *aSrc*, int *aSrcWidth*, int *aSrcHeight*, vsdk::SUMat & *aDescriptor*, vsdk::SUMat & *aClosestID*, vsdk::SUMat & *aDistance*)

Reconnects the input/outputs to predict process.

Parameters

in	*aModel*	Source model descriptor buffer. Datatype is 08U.
in	*aModelNum*	Model descriptor count.
in	*aSrc*	Source test image buffer. Datatype is 08U.
in	*aSrcWidth*	Source image width.
in	*aSrcHeight*	Source image height.
out	*aDescriptor*	Output test descriptor.
out	*aClosestID*	Output closest ID buffer. Datatype is 16U.
out	*aDistance*	Output histogram distance buffer. Datatype is 32S. The buffer width must be 32, and its height should be (modelNum+32-1)/32.

Returns

Error code for the reinitialization - see apexcv_error_codes.hpp

24.50.2.8 APEXCV_LIB_RESULT apexcv::Lbp::ReconnectTrainIO (vsdk::SUMat & *aSrc*, int *aSrcWidth*, int *aSrcHeight*, int *aSrcNum*, vsdk::SUMat & *aDescriptor*)

Reconnects the input/outputs to train process.

Parameters

in	*aSrc*	Source memory buffer. Datatype is VSDK_CV_8UC1.
in	*aSrcWidth*	Source image width.
in	*aSrcHeight*	Source image height.
in	*aSrcNum*	Number of source image.
out	*aDescriptor*	Destination memory buffer. Datatype is VSDK_CV_8UC1.

Returns

Error code for the reinitialization - see apexcv_error_codes.hpp

24.50.2.9 APEXCV_LIB_RESULT apexcv::Lbp::SelectApexCore (int32_t *aApexId*)

SelectApexCore.

Selects which APEX core (0 or 1) to be selected to run the processing. This function has to be called after the two Initialize methods.

Returns

Error code for updating the APU configuration - see apexcv_error_codes.hpp

24.51 apexcv::HoughLineDetector::Line Struct Reference

Line data structure associated with the Hough Line Detector.

Public Member Functions

- Line (int rho_=0, float theta_=0.f)

  Default constructor.

Public Attributes

- int rho

  Nearest distance of the line to center of the image.
- float theta

  Angle of the line's normal.

24.51.1 Detailed Description

Line data structure associated with the Hough Line Detector.

24.51.2 Constructor & Destructor Documentation

24.51.2.1 apexcv::HoughLineDetector::Line::Line (int *rho_* = 0, float *theta_* = 0.f) [inline]

Default constructor.

Parameters

rho↩ _	Nearest distance of the line to center of the image.
theta↩ _	Angle of the line's normal.

24.52 apexcv::LKPyramidOpticalFlow Class Reference

ApexCV L-K Pyramid Optical Flow.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc0Desc, vsdk::SUMat &aSrc1Desc, icp::Feature32S↩Descriptor &aCoor0Desc, icp::Feature32SDescriptor &aCoor1Desc, icp::Feature32SDescriptor &aCoor1Desc_O, int aMaxCorners, int aW, int aH, int aPyrLayers=1, int aBoxSize=7, int aNumIter=10, int aReqPadding=0)

  Initialization.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc0Desc, vsdk::SUMat &aSrc1Desc, icp::Feature32S↩Descriptor &aCoor0Desc, icp::Feature32SDescriptor &aCoor1Desc, icp::Feature32SDescriptor &aCoor1Desc_↩O)

Reconnect IO.

- APEXCV_LIB_RESULT SetBoxSize (int aBoxSize)

  Set Box filter Size.
- APEXCV_LIB_RESULT SetPyrLayers (int aPyrLayers)

  Set Pyramid Layers.
- APEXCV_LIB_RESULT SetNumIter (int aNumIter)

  Set Pyramid Layers.
- APEXCV_LIB_RESULT Process ()

  Process.
- APEXCV_LIB_RESULT SelectApexCore (int32_t aApexId)

  SelectApexCore.

24.52.1 Detailed Description

ApexCV L-K Pyramid Optical Flow.

apexcv::LKPyramidOpticalFlow is the Host-ACF interface for creating, initializing, executing and releasing the Multi-Scale Lucas-Kanade Optical Flow on Apex.

24.52.2 Member Function Documentation

24.52.2.1 APEXCV_LIB_RESULT apexcv::LKPyramidOpticalFlow::Initialize (vsdk::SUMat & *aSrc0Desc,* vsdk::SUMat & *aSrc1Desc,* icp::Feature32SDescriptor & *aCoor0Desc,* icp::Feature32SDescriptor & *aCoor1Desc,* icp::Feature32SDescriptor & *aCoor1Desc_O,* int *aMaxCorners,* int *aW,* int *aH,* int *aPyrLayers* = 1*,* int *aBoxSize* = 7*,* int *aNumIter* = 10*,* int *aReqPadding* = 0)

Initialization.

Initializes the intermediate buffers needed for the process, and initializes the ACF processes. The size of internal buffers determined by aMaxCorners / aW / aH; pyramid layers cannot exceed 4; Only supported box_size is 7 for now;

Parameters

aSrc0Desc	8-bit grayscale template source image (frame[t-1])
aSrc1Desc	8-bit grayscale patch source image (frame[t])
aCoor0Desc	template features list. X/Y coordinates are signed Q23.8 format
aCoor1Desc	patch features list. X/Y coordinates are signed Q23.8 format
aCoor1Desc_O	output features list. X/Y coordinates are signed Q23.8 format
aMaxCorners	maximum number of corners going to be tracked
aW	image width
aH	image height
aPyrLayers	Number of pyramid layers. 0 means no pyramid. Tracking on original resolution directly. Default is 1. Maximum is 4
aBoxSize	Box size. Only support 7 for now.
aNumIter	Number of iterations within each pyramid level. Default is 10
aReqPadding	Require image padding on each layer. 0: no padding to save cycles. 1: require padding. Default is 0

24.52.2.2 APEXCV_LIB_RESULT apexcv::LKPyramidOpticalFlow::Process ( )

Process.

Returns

> Error code for the ACF execution (zero on success). "position" field in output features' descriptor represent the tracked features' X/Y coordinate in signed Q23.8 format; "reverve" field in output features' descriptor represent whether feature is successfully tracked. 0: failed; 1: succeeded; "strength" field in output features' descriptor represent the sum of pixels' grayscale absolute difference between template and tracked patch window, i.e. the lower the better tracked.

24.52.2.3 APEXCV_LIB_RESULT apexcv::LKPyramidOpticalFlow::ReconnectIO (vsdk::SUMat & *aSrc0Desc,* vsdk::SUMat & *aSrc1Desc,* icp::Feature32SDescriptor & *aCoor0Desc,* icp::Feature32SDescriptor & *aCoor1Desc,* icp::Feature32SDescriptor & *aCoor1Desc_O*)

Reconnect IO.

Use this to connect the input, output buffers and features to be tracked. This will perform preprocessing so it is always needed to be done before "process" call. The number of features in Feature Descriptors cannot exceed the max_corners used in Initialize().

Parameters

aSrc0Desc	8-bit grayscale template source image (frame[t-1])
aSrc1Desc	8-bit grayscale patch source image (frame[t])
aCoor0Desc	template features list. X/Y coordinates are signed Q23.8 format
aCoor1Desc	patch features list. X/Y coordinates are signed Q23.8 format
aCoor1Desc↩_O	output features list. X/Y coordinates are signed Q23.8 format

24.52.2.4 APEXCV_LIB_RESULT apexcv::LKPyramidOpticalFlow::SelectApexCore (int32_t *aApexId*)

SelectApexCore.

Returns

> Error code for the ACF execution (zero on success).

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed per frame base.

Parameters

a↩ApexId	The ID of the desired APEX (e.g if there are 2 APEXs, valid values for lApexId would be 0 and 1).

24.52.2.5 APEXCV_LIB_RESULT apexcv::LKPyramidOpticalFlow::SetBoxSize (int *aBoxSize*)

Set Box filter Size.

Change the Box filter size used. only support 7 for now.

Parameters

aBoxSize	Box size. Only support 7 for now.

24.52.2.6 APEXCV_LIB_RESULT apexcv::LKPyramidOpticalFlow::SetNumIter (int *aNumIter*)

Set Pyramid Layers.

Change the number of iterations within each pyramid layer.

Parameters

aNumIter	Number of iterations within each pyramid level.

24.52.2.7 APEXCV_LIB_RESULT apexcv::LKPyramidOpticalFlow::SetPyrLayers (int *aPyrLayers*)

Set Pyramid Layers.

Change the pyramid layers used. Maximum is 4.

Parameters

aPyrLayers	Number of pyramid layers. Maximum is 4

24.53 apexcv::LKTrackerOpticalFlow Class Reference

ApexCV L-K Tracker Optical Flow.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc0Desc, vsdk::SUMat &aSrc1Desc, icp::Feature32SDescriptor &aCoor0Desc, icp::Feature32SDescriptor &aCoor1Desc, icp::Feature32SDescriptor &aCoor1Desc_O, int aMaxCorners, int aW, int aH, int aBoxSize=7, int aNumIter=10, int aSrcPadded=0, int aPadStartWidth=3000)

  Initialization.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc0Desc, vsdk::SUMat &aSrc1Desc, icp::Feature32SDescriptor &aCoor0Desc, icp::Feature32SDescriptor &aCoor1Desc, icp::Feature32SDescriptor &aCoor1Desc_O)

  Reconnect IO.
- APEXCV_LIB_RESULT SetBoxSize (int aBoxSize)

  Set Box filter Size.
- APEXCV_LIB_RESULT SetNumIter (int aNumIter)

Set number of iterations.

- APEXCV_LIB_RESULT Process ()

  Process.

- APEXCV_LIB_RESULT SelectApexCore (int32_t aApexId)

  SelectApexCore.

24.53.1 Detailed Description

ApexCV L-K Tracker Optical Flow.

apexcv::LKTrackerOpticalFlow is the Host-ACF interface for creating, initializing, executing and releasing the LKTracker on Apex.

24.53.2 Member Function Documentation

24.53.2.1 APEXCV_LIB_RESULT apexcv::LKTrackerOpticalFlow::Initialize (vsdk::SUMat & *aSrc0Desc,* vsdk::SUMat & *aSrc1Desc,* icp::Feature32SDescriptor & *aCoor0Desc,* icp::Feature32SDescriptor & *aCoor1Desc,* icp::Feature32SDescriptor & *aCoor1Desc_O,* int *aMaxCorners,* int *aW,* int *aH,* int *aBoxSize* = 7*,* int *aNumIter* = 10*,* int *aSrcPadded* = 0*,* int *aPadStartWidth* = 3000)

Initialization.

Initializes the intermediate buffers needed for the process, and initializes the ACF processes. The size of internal buffers are determined by max_corners / w / h; Supported box_size is only 7 for now;

Parameters

aSrc0Desc	8-bit grayscale template source image (frame[t-1])
aSrc1Desc	8-bit grayscale patch source image (frame[t])
aCoor0Desc	template features list. X/Y coordinates are signed Q23.8 format
aCoor1Desc	patch features list. X/Y coordinates are signed Q23.8 format
aCoor1Desc_O	output features list. X/Y coordinates are signed Q23.8 format
aMaxCorners	maximum number of corners going to be tracked
aW	image width
aH	image height
aBoxSize	Box size. Only support 7 for now
aNumIter	Number of iterations. Default is 10
aSrcPadded	input source images have been padded already: 0:no padding 1:allocated border but no replicated 2:image fully padded
aPadStartWidth	only image width less than aPadStartWidth, we do image padding (for tradeoff purpose between accuracy vs. performance)

24.53.2.2 APEXCV_LIB_RESULT apexcv::LKTrackerOpticalFlow::Process ( )

Process.

Returns

Error code for the ACF execution (zero on success). "position" field in output features' descriptor represent the tracked features' X/Y coordinate in signed Q23.8 format; "reverve" field in output features' descriptor represent whether feature is successfully tracked. 0: failed; 1: succeeded; "strength" field in output features' descriptor represent the sum of pixels' grayscale absolute difference between template and tracked patch window, i.e. the lower the better tracked.

24.53.2.3 APEXCV_LIB_RESULT apexcv::LKTrackerOpticalFlow::ReconnectIO (vsdk::SUMat & *aSrc0Desc,* vsdk::SUMat & *aSrc1Desc,* icp::Feature32SDescriptor & *aCoor0Desc,* icp::Feature32SDescriptor & *aCoor1Desc,* icp::Feature32SDescriptor & *aCoor1Desc_O*)

Reconnect IO.

Use this to connect the input, output buffers and features to be tracked. This will perform preprocessing so it is always needed to be done before "process" call. The number of features in Feature Descriptors cannot exceed the max_corners used in initialize().

Parameters

aSrc0Desc	8-bit grayscale template source image (frame[t-1])
aSrc1Desc	8-bit grayscale patch source image (frame[t])
aCoor0Desc	template features list. X/Y coordinates are signed Q23.8 format
aCoor1Desc	patch features list. X/Y coordinates are signed Q23.8 format
aCoor1Desc↩_O	output features list. X/Y coordinates are signed Q23.8 format

24.53.2.4 APEXCV_LIB_RESULT apexcv::LKTrackerOpticalFlow::SelectApexCore (int32_t *aApexId*)

SelectApexCore.

Returns

Error code for the ACF execution (zero on success).

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed per frame base.

Parameters

a↩ ApexId	The ID of the desired APEX (e.g if there are 2 APEXs, valid values for lApexId would be 0 and 1).

24.53.2.5 APEXCV_LIB_RESULT apexcv::LKTrackerOpticalFlow::SetBoxSize (int *aBoxSize*)

Set Box filter Size.

Change the Box filter size used. only support 7 for now.

Parameters

aBoxSize	Box size. Only support 7 for now

24.53.2.6 APEXCV_LIB_RESULT apexcv::LKTrackerOpticalFlow::SetNumIter (int *aNumIter*)

Set number of iterations.

Change the number of iterations within eacy pyramid layer

Parameters

aNumIter	Number of iterations.

24.54 apexcv::Magnitude Class Reference

Magnitude.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)
  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)
  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)
  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()
  Start processing and return when done.

24.54.1 Detailed Description

Magnitude.

Object of this class computes the magnitude of pixel value of *aSrc1* and *aSrc2* pixel by pixel. Where *aDst* = SQRT(square(*aSrc1*) + square(*aSrc2*))
Supported input type: VSDK_CV_16SC1, output type: VSDK_CV_16UC1
Supported width: 128 to 2048 pixels.

24.54.2 Member Function Documentation

24.54.2.1 APEXCV_LIB_RESULT apexcv::Magnitude::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_16UC1).

24.54.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.54.2.3 APEXCV_LIB_RESULT apexcv::Magnitude::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_16UC1).

24.54.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.55 apexcv::Max Class Reference

Max.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.55.1 Detailed Description

Max.

Object of this class returns the pixel-wise max values of two images.
Supported input type: VSDK_CV_8UC1, VSDK_CV_16SC1, output type: VSDK_CV_8UC1, VSDK_CV_16SC1
Supported width: 128 to 2048 pixels.

24.55.2 Member Function Documentation

24.55.2.1 APEXCV_LIB_RESULT apexcv::Max::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
`in`	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.55.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.55.2.3 APEXCV_LIB_RESULT apexcv::Max::ReconnectIO (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.55.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.56 apexcv::Mean Class Reference

Mean.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT Process (float &aMean)

  Process.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.56.1 Detailed Description

Mean.

Object of this class computes the mean value of the image.
Supported input type: VSDK_CV_8UC1.

24.56.2 Member Function Documentation

24.56.2.1 APEXCV_LIB_RESULT apexcv::Mean::Initialize (vsdk::SUMat & *aSrc*)

Initialize object (required).

This function initializes the object and connect IO. Process() can be called after that to execute the processing in APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).

24.56.2.2 APEXCV_LIB_RESULT apexcv::Mean::Process (float & *aMean*)

Process.

This function start and wait for kernel to complete, then calculate final mean value from output of kernel.

Returns

APEXCV_LIB_RESULT Error code.

24.56.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.56.2.4 APEXCV_LIB_RESULT apexcv::Mean::ReconnectIO (vsdk::SUMat & *aSrc*)

Reconnect IO (optional).

This function allows to change the Input images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).

24.56.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.57 apexcv::MeanStddev Class Reference

MeanStddev.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc)

    Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc)

    Reconnect IO (optional).
- APEXCV_LIB_RESULT Process (float &aMean, float &aStddev)

    Process.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

    Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

    Start processing and return when done.

24.57.1 Detailed Description

MeanStddev.

Object of this class computes the mean and standard deviation value of the image.
Output dimensions are 1x1 VSDK_CV_32SC1.
Supported input type: VSDK_CV_8UC1.

24.57.2 Member Function Documentation

24.57.2.1 APEXCV_LIB_RESULT apexcv::MeanStddev::Initialize (vsdk::SUMat & *aSrc*)

Initialize object (required).

This function initializes the object and connect IO. Process() can be called after that to execute the processing in APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).

24.57.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.57.2.3 APEXCV_LIB_RESULT apexcv::MeanStddev::Process (float & *aMean,* float & *aStddev*)

Process.

This function start and wait for kernel to complete, then calculate final mean and standard deviation value from output of kernel.

Returns

APEXCV_LIB_RESULT Error code.

24.57.2.4 APEXCV_LIB_RESULT apexcv::MeanStddev::ReconnectIO (vsdk::SUMat & *aSrc*)

Reconnect IO (optional).

This function allows to change the Input images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).

24.57.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.58 apexcv::MedianFilter Class Reference

Median filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.58.1 Detailed Description

Median filter.

Object of this class applies a Median filter to *aSrc*. Supported window size: 3 x 3 and 5 x 5 and 7x7
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.58.2 Member Function Documentation

24.58.2.1 APEXCV_LIB_RESULT apexcv::MedianFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3 or 5 or 7
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.58.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.58.2.3 APEXCV_LIB_RESULT apexcv::MedianFilter::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.58.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.59 apexcv::MergeChannel Class Reference

Channel merge class containing support for merging multiple single channels images into a single multi-channel image.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aSrc3, vsdk::SUMat &aSrc4, vsdk::SUMat &aDst)

  Channel Merge function.
- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aSrc3, vsdk::SUMat &aDst)

  Channel Merge function.
- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Channel Merge function.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aSrc3, vsdk↩ ::SUMat &aSrc4, vsdk::SUMat &aDst)

  Reconnect IO.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aSrc3, vsdk↩ ::SUMat &aDst)

Reconnect IO.

- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.59.1 Detailed Description

Channel merge class containing support for merging multiple single channels images into a single multi-channel image.

This class is an interface for using channel merge functions on the host.

24.59.2 Member Function Documentation

24.59.2.1 APEXCV_LIB_RESULT apexcv::MergeChannel::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aSrc3,* vsdk::SUMat & *aSrc4,* vsdk::SUMat & *aDst*)

Channel Merge function.

Merges a channel from a multiple channel image.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc2*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc3*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc4*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in,out	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC4

24.59.2.2 APEXCV_LIB_RESULT apexcv::MergeChannel::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aSrc3,* vsdk::SUMat & *aDst*)

Channel Merge function.

Merges a channel from a multiple channel image.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1

Parameters

in	*aSrc2*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc3*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in,out	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC3

24.59.2.3 APEXCV_LIB_RESULT apexcv::MergeChannel::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Channel Merge function.

Merges a channel from a multiple channel image.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc2*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in,out	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC2

24.59.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.59.2.5 APEXCV_LIB_RESULT apexcv::MergeChannel::ReconnectIO (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aSrc3,* vsdk::SUMat & *aSrc4,* vsdk::SUMat & *aDst*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc2*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1

Parameters

in	*aSrc3*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc4*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in,out	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC4

24.59.2.6 APEXCV_LIB_RESULT apexcv::MergeChannel::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aSrc3*, vsdk::SUMat & *aDst*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc2*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc3*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in,out	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC3

24.59.2.7 APEXCV_LIB_RESULT apexcv::MergeChannel::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in	*aSrc2*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC1
in,out	*aDst*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC2

24.59.2.8 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.60 apexcv::Min Class Reference

Min.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.60.1 Detailed Description

Min.

Object of this class returns the pixel-wise min values of two images.
Supported input type: VSDK_CV_8UC1, VSDK_CV_16SC1, output type: VSDK_CV_8UC1, VSDK_CV_16SC1
Supported width: 128 to 2048 pixels.

24.60.2 Member Function Documentation

24.60.2.1 APEXCV_LIB_RESULT apexcv::Min::Initialize (vsdk::SUMat & *aSrc1,* vsdk::SUMat & *aSrc2,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
`in`	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.60.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.60.2.3 APEXCV_LIB_RESULT apexcv::Min::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
`in`	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.60.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.61 apexcv::Mul Class Reference

Multiplication.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetScale (const uint8_t acScale)

  Set Scale.
- APEXCV_LIB_RESULT GetScale (uint8_t &aScale)

  Get Scale.
- APEXCV_LIB_RESULT SetPolicy (apexcv::eConvertPolicy aPolicy)

  Set Policy type.
- APEXCV_LIB_RESULT GetPolicy (apexcv::eConvertPolicy &aPolicy)

  Get Policy type.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.61.1 Detailed Description

Multiplication.

Object of this class returns element-wise multiplication between two images then divide by 255 or right shift by a value in range of [0,15] (default is 0). *aDst* can be VSDK_CV_8UC1 only if both source images are VSDK_CV_8UC1 and *aDst* is explicitly set to VSDK_CV_8UC1. It is otherwise VSDK_CV_16SC1.
Supported aSrc1 type: VSDK_CV_8UC1, aSrc1 type: VSDK_CV_8UC1, aDst type: VSDK_CV_8UC1 or
Supported aSrc1 type: VSDK_CV_8UC1, aSrc1 type: VSDK_CV_8UC1, aDst type: VSDK_CV_16SC1 or
Supported aSrc1 type: VSDK_CV_8UC1, aSrc1 type: VSDK_CV_16SC1, aDst type: VSDK_CV_16SC1 or
Supported aSrc1 type: VSDK_CV_16SC1, aSrc1 type: VSDK_CV_16SC1, aDst type: VSDK_CV_16SC1
Supported width: 128 to 2048 pixels.

24.61.2 Member Function Documentation

24.61.2.1 APEXCV_LIB_RESULT apexcv::Mul::GetPolicy (apexcv::eConvertPolicy & *aPolicy*)

Get Policy type.

This function allows to read the value of the overflow policy type.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

out	*aPolicy*	Overflow policy type.

24.61.2.2 APEXCV_LIB_RESULT apexcv::Mul::GetScale (uint8_t & *aScale*)

Get Scale.

This function allows to read the scale value.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aScale*	Scale amount (a value in range of [0,15] or 255)

24.61.2.3 APEXCV_LIB_RESULT apexcv::Mul::Initialize (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.61.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.61.2.5 APEXCV_LIB_RESULT apexcv::Mul::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.61.2.6 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.61.2.7 APEXCV_LIB_RESULT apexcv::Mul::SetPolicy (apexcv::eConvertPolicy *aPolicy*)

Set Policy type.

This function allows to change the overflow policy type.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aPolicy*	Overflow policy type

24.61.2.8 APEXCV_LIB_RESULT apexcv::Mul::SetScale (const uint8_t *acScale*)

Set Scale.

This function allows to change the scale value: (aSrc1$*$aSrc2)/255 for 255, (aSrc1$*$aSrc2)$>>$acScale for [0,15]

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*acScale*	Scale amount (a value in range of [0,15] or 255)

24.62 apexcv::Orb Class Reference

Apex Orb class.

Classes

- class Corner

  ORB::Corner.

Public Member Functions

- APEXCV_LIB_RESULT Create (unsigned char aApexId=0, unsigned char aNrOfThreads=2, unsigned int aBorderSize=32, unsigned int aPatchSize=32, unsigned int aRadius=16, unsigned int aDescriptorSizeInBytes=32, unsigned int aFastCircumference=16, unsigned int aFastThreshold=20, float aHarrisK=0.04f, unsigned int aNrOfKeypoints=512)

  Initializes the Orb class. The function begins by allocating all the necessary buffers then initializes the FAST keypoint detector, Harris Corner Score and Orb orientation.
- APEXCV_LIB_RESULT Detect (vsdk::SUMat &aInImg)

  *Starts the processing of the Orb keypoint detector. Three processes will be run on APEX: FAST, Harris and Orb orientation. After APEX finishes, the best **nrOfKeypoints** are selected based on Harris Score. that the algo should look for. The number of detected keypoints could be less than **nrOfKeypoints** if so the value is updated.*
- APEXCV_LIB_RESULT Compute (vsdk::SUMat &aInSmoothedImg, vsdk::SUMat &aOutDescriptors)

  Starts the processing of the Orb rBrief.
- vsdk::SUMat & GetChunkOffsets ()

  Returns the chunk offsets

  .
- vsdk::SUMat & GetPatchSize ()

  Returns the patch size of rBRIEF/IC

  .
- vsdk::SUMat & GetRadius ()

  Returns the radius of the IC calculus

  .
- vsdk::SUMat & GetFastOut ()

  Returns the FAST9 score image

  .
- vsdk::SUMat & GetIcoAngles ()

  Returns the angles for each patch calculated by the ICO

  .
- vsdk::SUMat & GetSerializerOut ()

  Returns the packed coordinates and metric for the FAST9 Img

  .
- int GetNrOfDetectedKeypoints ()

Counter for the keypoints that were detected by FAST

.

- int GetNrOfValidKeypoints ()

  Counter for the keypoints that are inside the border

  .

- bool DataIsValid ()

  Orb keypoints will be processed

  .

- APEXCV_LIB_RESULT SetBriefSamplingPattern (vsdk::SUMat &aInBitPattern)

  Sets the sampling pattern for BRIEF

  .

- std::vector< Orb::Corner > & GetKeypoints ()

  Dumps the keypoints in a standard format
  Usage: std::vector<Orb::Corner> &kpntVec = orb.GetKeypoints(); if(kpntVec.size() > 0) {// Only now the buffer is valid};.

24.62.1 Detailed Description

Apex Orb class.

apexcv::Orb is the host-ACF interface for creating, initializing, executing and releasing the resources for running the Orb algorithm

24.62.2 Member Function Documentation

24.62.2.1 APEXCV_LIB_RESULT apexcv::Orb::Compute (vsdk::SUMat & *aInSmoothedImg,* vsdk::SUMat & *aOutDescriptors*)

Starts the processing of the Orb rBrief.

Parameters

in	*aInSmoothedImg*	8-bit grayscale source image
out	*aOutDescriptors-*	[Output] - 8-bit unsigned, will containt out descr. Expected size: descriptorSizeBytes ∗ nrOfKeypoints

Note

Maximum resolution is **1920 x 1080 pixels** with **64** CUs.

Returns

Check apexcv_error_codes.h to see to possible outcomes

24.62.2.2 APEXCV_LIB_RESULT apexcv::Orb::Create (unsigned char *aApexId* = 0, unsigned char *aNrOfThreads* = 2, unsigned int *aBorderSize* = 32, unsigned int *aPatchSize* = 32, unsigned int *aRadius* = 16, unsigned int *aDescriptorSizeInBytes* = 32, unsigned int *aFastCircumference* = 16, unsigned int *aFastThreshold* = 20, float *aHarrisK* = 0.04f, unsigned int *aNrOfKeypoints* = 512)

Initializes the Orb class. The function begins by allocating all the necessary buffers then initializes the FAST keypoint detector, Harris Corner Score and Orb orientation.

Parameters

in	*aApexId*	Discriminates between the two APEX accelerators
in	*aNrOfThreads*	Indicates how many APUs the algorithm will use
in	*aBorderSize*	Image border that will be trimmed from aInImg
in	*aPatchSize*	A square region that contains the center feature
in	*aRadius*	Hypothetical circle with its center inside the center of the patch
in	*aDescriptorSizeInBytes*	The size of the descriptor in bytes for each patch
in	*aFastCircumference*	The circumference of the circle where the FAST9 algorithm is applied
in	*aFastThreshold*	Fast algorithm detection threshold
in	*aHarrisK*	Harris Corner Coefficent
in	*aNrOfKeypoints*	maximum number of keypoints that the algo should look for

Returns

Check apexcv_error_codes.h to see to possible outcomes

Note

Maximum resolution is **1920 x 1080 pixels** with **64** CUs.

24.62.2.3 bool apexcv::Orb::DataIsValid ( )

Orb keypoints will be processed
.

24.62.2.4 APEXCV_LIB_RESULT apexcv::Orb::Detect (vsdk::SUMat & *aInImg*)

Starts the processing of the Orb keypoint detector. Three processes will be run on APEX: FAST, Harris and Orb orientation. After APEX finishes, the best **nrOfKeypoints** are selected based on Harris Score. that the algo should look for. The number of detected keypoints could be less than **nrOfKeypoints** if so the value is updated.

Parameters

in	*aInImg*	8-bit grayscale source image

Returns

Error code for initialization (zero on success).

Note

This method has the highest latency and the processing is split between APEX and ARM

Returns

Check apexcv_error_codes.h to see to possible outcomes

24.62.2.5 vsdk::SUMat& apexcv::Orb::GetChunkOffsets ( )

Returns the chunk offsets
.

Returns

Reference to an internal pointer that holds the chunk offets. Each chunk is described by a 32 bit value, that represents the offset in bytes from the start of the input image

24.62.2.6 vsdk::SUMat& apexcv::Orb::GetFastOut ( )

Returns the FAST9 score image
.

Returns

Pairs of coordianats <y,x> for every keypoint, each pairs has 32 bits

24.62.2.7 vsdk::SUMat& apexcv::Orb::GetIcoAngles ( )

Returns the angles for each patch calculated by the ICO
.

Returns

An angles for each keypoint, [0-360] degrees are mapped to [0-255]

24.62.2.8 std::vector<Orb::Corner>& apexcv::Orb::GetKeypoints ( )

Dumps the keypoints in a standard format
Usage: std::vector<Orb::Corner> &kpntVec = orb.GetKeypoints(); if(kpntVec.size() > 0) {// Only now the buffer is valid};.

Returns

Array of keypoints, see Orb::Corner type

24.62.2.9 int apexcv::Orb::GetNrOfDetectedKeypoints ( )

Counter for the keypoints that were detected by FAST
.

24.62.2.10 int apexcv::Orb::GetNrOfValidKeypoints ( )

Counter for the keypoints that are inside the border
.

24.62.2.11 vsdk::SUMat& apexcv::Orb::GetPatchSize ( )

Returns the patch size of rBRIEF/IC

.

Returns

Patch size that rBrief will process, 8 bits, signed

24.62.2.12 vsdk::SUMat& apexcv::Orb::GetRadius ( )

Returns the radius of the IC calculus

.

Returns

Radius of the orientation algorithm, 8 bit, signed

24.62.2.13 vsdk::SUMat& apexcv::Orb::GetSerializerOut ( )

Returns the packed coordinates and metric for the FAST9 Img

.

24.62.2.14 APEXCV_LIB_RESULT apexcv::Orb::SetBriefSamplingPattern (vsdk::SUMat & *aInBitPattern*)

Sets the sampling pattern for BRIEF

.

Returns

Check apexcv_error_codes.h to see to possible outcomes

24.63 apexcv::Phase Class Reference

Phase.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.63.1 Detailed Description

Phase.

Object of this class is computes the angles for each pixel and stores this in a VSDK_CV_8UC1 image. Where result is then translated to 0 <= result< 2pi. Each result value is then mapped to the range 0 to 255 inclusive.
Supported input type: VSDK_CV_16SC1, output type: VSDK_CV_8UC1
Supported width: 128 to 2048 pixels.

24.63.2 Member Function Documentation

24.63.2.1 APEXCV_LIB_RESULT apexcv::Phase::Initialize (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Horizontal gradient (VSDK_CV_16SC1).
in	*aSrc2*	Vertical gradient (VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.63.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.63.2.3 APEXCV_LIB_RESULT apexcv::Phase::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc1*	Horizontal gradient (VSDK_CV_16SC1).
`in`	*aSrc2*	Vertical gradient (VSDK_CV_16SC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.63.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a← ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.64 apexcv::PrewittXFilter Class Reference

Prewitt X filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.64.1 Detailed Description

Prewitt X filter.

Object of this class applies a Prewitt X filter to *aSrc*. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.64.2 Member Function Documentation

24.64.2.1 APEXCV_LIB_RESULT apexcv::PrewittXFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size, 3
in,out	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.64.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.64.2.3 APEXCV_LIB_RESULT apexcv::PrewittXFilter::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.64.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.65 apexcv::PrewittYFilter Class Reference

Prewitt Y filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.65.1 Detailed Description

Prewitt Y filter.

Object of this class applies a Prewitt Y filter to *aSrc*. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.65.2 Member Function Documentation

24.65.2.1 APEXCV_LIB_RESULT apexcv::PrewittYFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size, 3
in,out	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.65.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.65.2.3 APEXCV_LIB_RESULT apexcv::PrewittYFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.65.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.66 apexcv::PyramidCreation Class Reference

Pyramid creation class.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initializes the class.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Connects the input/outputs to the process.
- APEXCV_LIB_RESULT Process ()

  Run the pyramid creation process.
- APEXCV_LIB_RESULT SelectApexCore (int32_t aApexId)

  SelectApexCore.

24.66.1 Detailed Description

Pyramid creation class.

This class is an interface for using the pyramid creation algorithm.

24.66.2 Member Function Documentation

24.66.2.1 APEXCV_LIB_RESULT apexcv::PyramidCreation::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Initializes the class.

Connects the buffers to the process.

Parameters

aSrc	Source memory buffer. Datatype is 08U.
aDst	Destination memory buffer. Datatype is 08U.

24.66.2.2 APEXCV_LIB_RESULT apexcv::PyramidCreation::Process ( )

Run the pyramid creation process.

Upscale or downscale *src* buffer and stores the result in *dst* buffer. The process will upscale *src* buffer if *isPyramidUp* = true. The process will downscale *src* buffer if *isPyramidUp* = false. Default is *isPyramidUp* = true.

Supported datatypes are:

- unsigned 8 bit to unsigned 8 bit

Returns

Error code (zero on success).

24.66.2.3 APEXCV_LIB_RESULT apexcv::PyramidCreation::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Connects the input/outputs to the process.

Use this to reconnect the Input and Output Buffers. This only needs to be done if the connected Input/Outputs are changed. If only the data within (no size, or type changes), then this does not need to be called.

Parameters

aSrc	Source memory buffer. Datatype is 08U.
aDst	Destination memory buffer. Datatype is 08U.

24.66.2.4 APEXCV_LIB_RESULT apexcv::PyramidCreation::SelectApexCore (int32_t *aApexId*)

SelectApexCore.

Returns

Error code for the ACF execution (zero on success).

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed per frame base.

Parameters

a↩ ApexId	The ID of the desired APEX (e.g if there are 2 APEXs, valid values for lApexId would be 0 and 1).

24.67 apexcv::PyramidMultiCreation Class Reference

Pyramid_multi creation class.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat(&aDstArray)[PYRAMID_LEVELS])
  Initializes the class: connects the buffer to the process port, and allocates/initializes any internal buffers.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat(&aDstArray)[PYRAMID_LEVELS])
  Connects the input/outputs to the process.
- APEXCV_LIB_RESULT Process ()
  Run the pyramid_multi creation process.
- APEXCV_LIB_RESULT SelectApexCore (int32_t aApexId)
  SelectApexCore.

Static Public Attributes

- static const int32_t PYRAMID_LEVELS = 4

24.67.1 Detailed Description

Pyramid_multi creation class.

This class is an interface for using the pyramid_multi creation algorithm.

24.67.2 Member Function Documentation

24.67.2.1 APEXCV_LIB_RESULT apexcv::PyramidMultiCreation::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat(&) *aDstArray[PYRAMID_LEVELS]*)

Initializes the class: connects the buffer to the process port, and allocates/initializes any internal buffers.

Parameters

in	*aSrc*	Source memory buffer. Datatype is VSDK_CV_8UC1.
out	*aDstArray[0]*	Destination memory buffer for scale pyramid 1/2. Datatype is VSDK_CV_8UC1.
out	*aDstArray[1]*	Destination memory buffer for scale pyramid 1/4. Datatype is VSDK_CV_8UC1.
out	*aDstArray[2]*	Destination memory buffer for scale pyramid 1/8. Datatype is VSDK_CV_8UC1.
out	*aDstArray[3]*	Destination memory buffer for scale pyramid 1/16. Datatype is VSDK_CV_8UC1.

Returns

Error code for the initialization - see apexcv_error_codes.hpp

24.67.2.2 APEXCV_LIB_RESULT apexcv::PyramidMultiCreation::Process ( )

Run the pyramid_multi creation process.

Returns

Error code (zero on success).

24.67.2.3 APEXCV_LIB_RESULT apexcv::PyramidMultiCreation::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat(&) *aDstArray[PYRAMID_LEVELS]*)

Connects the input/outputs to the process.

Use this to reconnect the Input and Output Buffers. This only needs to be done if the connected Input/Outputs are changed. If only the data within (no size, or type changes), then this does not need to be called.

Parameters

in	*aSrc*	Source memory buffer. Datatype is VSDK_CV_8UC1.
out	*aDstArray[0]*	Destination memory buffer for scale pyramid 1/2. Datatype is VSDK_CV_8UC1.
out	*aDstArray[1]*	Destination memory buffer for scale pyramid 1/4. Datatype is VSDK_CV_8UC1.
out	*aDstArray[2]*	Destination memory buffer for scale pyramid 1/8. Datatype is VSDK_CV_8UC1.
out	*aDstArray[3]*	Destination memory buffer for scale pyramid 1/16. Datatype is VSDK_CV_8UC1.

Returns

Error code for the initialization - see apexcv_error_codes.hpp

24.67.2.4 APEXCV_LIB_RESULT apexcv::PyramidMultiCreation::SelectApexCore (int32_t *aApexId*)

SelectApexCore.

Returns

Error code for the ACF execution (zero on success).

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed per frame base.

Parameters

a↩ ApexId	The ID of the desired APEX (e.g if there are 2 APEXs, valid values for lApexId would be 0 and 1).

24.67.3 Member Data Documentation

24.67.3.1 const int32_t apexcv::PyramidMultiCreation::PYRAMID_LEVELS = 4 [static]

This is static constant variable that defined pyramid multi level.

24.68 apexcv::Remap Class Reference

Apex Remap.

Public Types

Public Member Functions

- APEXCV_LIB_RESULT Initialize (float ∗aMap, vsdk::SUMat aSrc, vsdk::SUMat aDst, INTER_TYPE aInterp, B↩ORDER_TYPE aBorder, uint32_t aBorderValue)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT Process ()

  Start processing and return when done.
- APEXCV_LIB_RESULT GenerateLUTFromCalibLoader (const char ∗apFilename, uint32_t aDstWidth, uint32↩_t aDstHeight, uint32_t aSrcWidth, uint32_t aSrcHeight, uint32_t aDestBlockWidth, uint32_t aDestBlockHeight, uint32_t aRefSrcBlockWidth, uint32_t aRefSrcBlockHeight)

  Generate the look-up table for top view perspective.
- int RetLUTs (vsdk::SUMat &aDeltaDesc, vsdk::SUMat &aLocalOffsetDesc, vsdk::SUMat &aBlockOffsetDesc)

  Returns the LUTs needed for the remap ACF process.

Static Public Member Functions

- static void GenerateFloatMap (vsdk::SUMat &aInput, vsdk::SUMat &aOutput, float ∗apMap, float aMaxOffsetPerDim, int aSeed)

  Generate random float map.

24.68.1 Detailed Description

Apex Remap.

apexcv::Remap is the host-ACF interface for creating, initializing, executing and releasing image remap on Apex.

24.68.2 Member Enumeration Documentation

24.68.2.1 enum **apexcv::Remap::BORDER_TYPE**

Border options.

Enumerator

BORDER_CONSTANT Set the border to a specified value.

BORDER_REPLICATE Replicate the border.

BORDER_REFLECT Reflect the border.

BORDER_WRAP Wrap the border.

24.68.2.2 enum **apexcv::Remap::INTER_TYPE**

Interpolation options.

Enumerator

INTER_NN Nearest neighbour interpolation.

INTER_LINEAR Bilinear interpolation.

INTER_CUBIC Bicubic interpolation over 4x4 pixel neighborhood.

INTER_AREA Resampling using pixel area relation. It may be a preferred method for image decimation, as it gives moire-free results. But when the image is zoomed, it is similar to the INTER_NEAREST method.

INTER_LANCZOS4 Lanczos interpolation over 8x8 pixel neighborhood.

24.68.3 Member Function Documentation

24.68.3.1 static void apexcv::Remap::GenerateFloatMap (vsdk::SUMat & *aInput,* vsdk::SUMat & *aOutput,* float ∗ *apMap,* float *aMaxOffsetPerDim,* int *aSeed*) `[static]`

Generate random float map.

This method generates a random float map based off the input parameter's dimensions. map must be a pointer to an appropriate sized buffer. max_offset_per_dim specifies the maximum radius of the dst pixel from the source, ie. a value of 3 means the dst pixel can be chosen from a 3x3 window centered at the x, y location of the current pixel.

Parameters

in	*aInput*	8-bit grayscale source image, VSDK_CV_8UC1.
in	*aOutput*	8-bit grayscale destination image, VSDK_CV_8UC1, used for dimensions only.
out	*apMap,X,Y*	pair of the location in the input image for each pixel in the output image.
in	*aMaxOffsetPerDim*	Allowed variation in the float map from a direct mapping input to output.
in	*aSeed*	Random number generator seed.

24.68.3.2 APEXCV_LIB_RESULT apexcv::Remap::GenerateLUTFromCalibLoader (const char ∗ *apFilename,* uint32_t *aDstWidth,* uint32_t *aDstHeight,* uint32_t *aSrcWidth,* uint32_t *aSrcHeight,* uint32_t *aDestBlockWidth,* uint32_t *aDestBlockHeight,* uint32_t *aRefSrcBlockWidth,* uint32_t *aRefSrcBlockHeight*)

Generate the look-up table for top view perspective.

Returns

Error code for the execution (APEXCV_SUCCESS on success).

24.68.3.3 APEXCV_LIB_RESULT apexcv::Remap::Initialize (float ∗ *aMap,* vsdk::SUMat *aSrc,* vsdk::SUMat *aDst,* INTER_TYPE *aInterp,* BORDER_TYPE *aBorder,* uint32_t *aBorderValue*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code (APEXCV_SUCCESS on success).

Parameters

in	*aMap*	X, Y pair of the location in the input image for each pixel in the output image.
in	*aSrc*	8-bit grayscale source image, VSDK_CV_8UC1.
in,out	*aDst*	8-bit grayscale destination image, VSDK_CV_8UC1.
in	*aInterp*	Interpolation mode, only INTER_LINEAR is supported.
in	*aBorder*	Border management scheme, only BORDER_REPLICATE is supported.
in	*aBorderValue*	Border value, not supported.

24.68.3.4 APEXCV_LIB_RESULT apexcv::Remap::Process ( )

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV_LIB_RESULT Error code (APEXCV_SUCCESS on success).

24.68.3.5 APEXCV_LIB_RESULT apexcv::Remap::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code (APEXCV_SUCCESS on success).

Parameters

`in`	*aSrc*	8-bit grayscale source image, VSDK_CV_8UC1.
`in,out`	*aDst*	8-bit grayscale destination image, VSDK_CV_8UC1.

24.68.3.6 int apexcv::Remap::RetLUTs (vsdk::SUMat & *aDeltaDesc*, vsdk::SUMat & *aLocalOffsetDesc*, vsdk::SUMat & *aBlockOffsetDesc*)

Returns the LUTs needed for the remap ACF process.

This method returns the LUTs which was configured by initialize().

Returns

Error code for the execution (zero on success).

24.69 apexcv::Resize Class Reference

Apex Resize.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrcImage, vsdk::SUMat &aDestImage)

  Initialize the resize ACF process. This function will initialize all objects of Resize and it will only be called once.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrcImage, vsdk::SUMat &aDestImage)

  Reconnect IO Reconnects the input and output ports to the object. This is how to update the image containers the object uses. This function support full type reconnect meaning that changes in size and type (8 bit, 16 bit) are possible if within allowed range. Note: the object needs to be initialized before calling reconnectIO.
- APEXCV_LIB_RESULT Process ()

  Execute the ACF resize process.

24.69.1 Detailed Description

Apex Resize.

apexcv::Resize is the host-ACF interface for creating, initializing, executing and releasing image resize on Apex. Apex Resize uses memcpy when source image width/height is equal to destination image width/height. for performance consideration, resized result image buffer needs be allocated DMA friendly. for destination size of width and height, the dst buffer should be allocated to a SUMat with size of (width+127)/128∗128+32, height+32, and use SUMat ROI Rect(0, 0, width, height) to specify actual dst image size.

24.69.2 Member Function Documentation

24.69.2.1 APEXCV_LIB_RESULT apexcv::Resize::Initialize (vsdk::SUMat & *aSrcImage*, vsdk::SUMat & *aDestImage*)

Initialize the resize ACF process. This function will initialize all objects of Resize and it will only be called once.

Returns

APEXCV_LIB_RESULT Error code (APEXCV_SUCCESS on success).

Parameters

aSrcImage	Source memory buffer.
aDestImage	Destination memory buffer.

24.69.2.2 APEXCV_LIB_RESULT apexcv::Resize::Process ( )

Execute the ACF resize process.

Returns

APEXCV_LIB_RESULT Error code (APEXCV_SUCCESS on success).

24.69.2.3 APEXCV_LIB_RESULT apexcv::Resize::ReconnectIO (vsdk::SUMat & *aSrcImage*, vsdk::SUMat & *aDestImage*)

Reconnect IO Reconnects the input and output ports to the object. This is how to update the image containers the object uses. This function support full type reconnect meaning that changes in size and type (8 bit, 16 bit) are possible if within allowed range. Note: the object needs to be initialized before calling reconnectIO.

Returns

APEXCV_LIB_RESULT Error code (APEXCV_SUCCESS on success).

Parameters

aSrcImage	Source memory buffer.
aDestImage	Destination memory buffer.

24.70 apexcv::SaturateFilterHT Class Reference

Saturate filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.70.1 Detailed Description

Saturate filter, Hand Tuned (optimized).

Object of this class applies a Saturate filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. *aDst* and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_16SC1, output type: VSDK_CV_8SC1.
Supported width: 128 to 2048 pixels.

24.70.2 Member Function Documentation

24.70.2.1 APEXCV_LIB_RESULT apexcv::SaturateFilterHT::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8SC1).

24.70.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.70.2.3 APEXCV_LIB_RESULT apexcv::SaturateFilterHT::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8SC1).

24.70.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.71 apexcv::ScharrFilter Class Reference

Scharr filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.

- APEXCV_LIB_RESULT **Process** ()
  Start processing and return when done.

24.71.1 Detailed Description

Scharr filter.

Object of this class applies a Scharr filter to *aSrc*. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported output type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.71.2 Member Function Documentation

24.71.2.1 APEXCV_LIB_RESULT apexcv::ScharrFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3 or 5
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.71.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.71.2.3 APEXCV_LIB_RESULT apexcv::ScharrFilter::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.71.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.72 apexcv::ScharrXFilter Class Reference

Scharr X filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.72.1 Detailed Description

Scharr X filter.

Object of this class applies a Scharr X filter to *aSrc*. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.

Supported input type: VSDK_CV_8UC1.
Supported output type: VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.72.2 Member Function Documentation

24.72.2.1 APEXCV_LIB_RESULT apexcv::ScharrXFilter::Initialize (vsdk::SUMat & *aSrc*, int *aWindowSize*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3 or 5
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_16SC1).

24.72.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.72.2.3 APEXCV_LIB_RESULT apexcv::ScharrXFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.72.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.73 apexcv::ScharrXYFilter Class Reference

Scharr XY filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDstX, vsdk::SUMat &a↩DstY)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDstX, vsdk::SUMat &aDstY)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.73.1 Detailed Description

Scharr XY filter.

Object of this class applies a Scharr X and Y filter to *aSrc*. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported output type: VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.73.2 Member Function Documentation

24.73.2.1 APEXCV_LIB_RESULT apexcv::ScharrXYFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDstX,* vsdk::SUMat & *aDstY*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size, 3 or 5
in,out	*aDstX*	Destination Image buffer, X (VSDK_CV_16SC1).
in,out	*aDstY*	Destination Image buffer, Y (VSDK_CV_16SC1).

24.73.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.73.2.3 APEXCV_LIB_RESULT apexcv::ScharrXYFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDstX*, vsdk::SUMat & *aDstY*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDstX*	Destination image buffer, X (VSDK_CV_16SC1).
in,out	*aDstY*	Destination image buffer, Y (VSDK_CV_16SC1).

24.73.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a← ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.74 apexcv::ScharrYFilter Class Reference

Scharr Y filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.74.1 Detailed Description

Scharr Y filter.

Object of this class applies a Scharr Y filter to *aSrc*. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported output type: VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.74.2 Member Function Documentation

24.74.2.1 APEXCV_LIB_RESULT apexcv::ScharrYFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size, 3 or 5
in,out	*aDst*	Destination Image buffer (VSDK_CV_16SC1).

24.74.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.74.2.3 APEXCV_LIB_RESULT apexcv::ScharrYFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.74.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.75 apexcv::SeparableFilterHT Class Reference

Separable filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, signed char(&aFilterRow)[3], signed char(&aFilterCol)[3], vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, signed char(&aFilterRow)[5], signed char(&aFilterCol)[5], vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetFilterRow (signed char(&aFilterRow)[3])

  set Filter Row.
- APEXCV_LIB_RESULT SetFilterCol (signed char(&aFilterCol)[3])

  set Filter Col.
- APEXCV_LIB_RESULT SetFilterRow (signed char(&aFilterRow)[5])

  set Filter Row.
- APEXCV_LIB_RESULT SetFilterCol (signed char(&aFilterCol)[5])

  set Filter Column.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.75.1 Detailed Description

Separable filter, Hand Tuned (optimized).

Object of this class applies a Separable filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.75.2 Member Function Documentation

24.75.2.1 APEXCV_LIB_RESULT apexcv::SeparableFilterHT::Initialize (vsdk::SUMat & *aSrc,* signed char(&) *aFilterRow[3],* signed char(&) *aFilterCol[3],* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aFilterRow*	3x1 Horizontal filter parameters
in	*aFilterCol*	1x3 Vertical filter parameters
in,out	*aDst*	Destination Image buffer (VSDK_CV_16SC1).

24.75.2.2 APEXCV_LIB_RESULT apexcv::SeparableFilterHT::Initialize (vsdk::SUMat & *aSrc,* signed char(&) *aFilterRow[5],* signed char(&) *aFilterCol[5],* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aFilterRow*	5x1 Horizontal filter parameters
in	*aFilterCol*	1x5 Vertical filter parameters
in,out	*aDst*	Destination Image buffer (VSDK_CV_16SC1).

24.75.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.75.2.4 APEXCV_LIB_RESULT apexcv::SeparableFilterHT::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.75.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.75.2.6 APEXCV_LIB_RESULT apexcv::SeparableFilterHT::SetFilterCol (signed char(&) *aFilterCol[3]*)

set Filter Col.

This function allows to change filter coefficients.

Returns

APEXCV_LIB_RESULT Error code.[in] 1x3 Vertical filter parameters

24.75.2.7 APEXCV_LIB_RESULT apexcv::SeparableFilterHT::SetFilterCol (signed char(&) *aFilterCol[5]*)

set Filter Column.

This function allows to change filter coefficients.

Returns

APEXCV_LIB_RESULT Error code.[in] 1x5 Vertical filter parameters

24.75.2.8 APEXCV_LIB_RESULT apexcv::SeparableFilterHT::SetFilterRow (signed char(&) *aFilterRow[3]*)

set Filter Row.

This function allows to change filter coefficients.

Returns

APEXCV_LIB_RESULT Error code.[in] 3x1 Horizontal filter parameters

24.75.2.9 APEXCV_LIB_RESULT apexcv::SeparableFilterHT::SetFilterRow (signed char(&) *aFilterRow[5]*)

set Filter Row.

This function allows to change filter coefficients.

Returns

APEXCV_LIB_RESULT Error code.[in] 5x1 Horizontal filter parameters

24.76 apexcv::SobelFilter Class Reference

Sobel filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.76.1 Detailed Description

Sobel filter.

Object of this class applies a Sobel filter to *aSrc*. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.76.2 Member Function Documentation

24.76.2.1 APEXCV_LIB_RESULT apexcv::SobelFilter::Initialize (vsdk::SUMat & *aSrc*, int *aWindowSize*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3 or 5
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.76.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.76.2.3 APEXCV_LIB_RESULT apexcv::SobelFilter::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.76.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.77 apexcv::SobelFilterHT Class Reference

Sobel filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

    Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

    Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

    Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

    Start processing and return when done.

24.77.1 Detailed Description

Sobel filter, Hand Tuned (optimized).

Object of this class applies a Sobel filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1.
Supported width: 128 to 2048 pixels.

24.77.2 Member Function Documentation

24.77.2.1 APEXCV_LIB_RESULT apexcv::SobelFilterHT::Initialize (vsdk::SUMat & *aSrc*, int *aWindowSize*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

aSrc	Source Image buffer (VSDK_CV_8UC1).
aWindowSize	Window Size, 3 or 5
aDst	Destination Image buffer (VSDK_CV_16SC1).

24.77.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.77.2.3 APEXCV_LIB_RESULT apexcv::SobelFilterHT::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_16SC1).

24.77.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.78 apexcv::SobelXFilter Class Reference

Sobel X filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.78.1 Detailed Description

Sobel X filter.

Object of this class applies a Sobel X filter to *aSrc*. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.78.2 Member Function Documentation

24.78.2.1 APEXCV_LIB_RESULT apexcv::SobelXFilter::Initialize (vsdk::SUMat & *aSrc*, int *aWindowSize*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3 or 5
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.78.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.78.2.3 APEXCV_LIB_RESULT apexcv::SobelXFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.78.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a← *ApexId*	ID of the APEX core used for performing the processing (0 or 1).

24.79 apexcv::SobelXFilterHT Class Reference

Sobel X filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.79.1 Detailed Description

Sobel X filter, Hand Tuned (optimized).

Object of this class applies a Sobel X filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8SC1.
Supported width: 128 to 2048 pixels.

24.79.2 Member Function Documentation

24.79.2.1 APEXCV_LIB_RESULT apexcv::SobelXFilterHT::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
in	*aWindowSize*	Window Size, 3
in,out	*aDst*	Destination Image buffer (VSDK_CV_8SC1).

24.79.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.79.2.3 APEXCV_LIB_RESULT apexcv::SobelXFilterHT::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8SC1).

24.79.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.80 apexcv::SobelXYFilter Class Reference

Sobel XY filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDstX, vsdk::SUMat &a↩DstY)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDstX, vsdk::SUMat &aDstY)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.80.1 Detailed Description

Sobel XY filter.

Object of this class applies a Sobel X and Y filter to *aSrc*. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.80.2 Member Function Documentation

24.80.2.1 APEXCV_LIB_RESULT apexcv::SobelXYFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDstX,* vsdk::SUMat & *aDstY*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3
`in,out`	*aDstX*	Destination Image buffer, X (VSDK_CV_8UC1).
`in`	*aDstY*	Destination Image buffer, Y (VSDK_CV_8UC1).

24.80.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.80.2.3 APEXCV_LIB_RESULT apexcv::SobelXYFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDstX*, vsdk::SUMat & *aDstY*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDstX*	Destination image buffer (VSDK_CV_8UC1).
`in,out`	*aDstY*	Destination image buffer (VSDK_CV_8UC1).

24.80.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.81 apexcv::SobelYFilter Class Reference

Sobel Y filter.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.81.1 Detailed Description

Sobel Y filter.

Object of this class applies a Sobel Y filter to *aSrc*. Supported window size: 3 x 3 and 5 x 5
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1.
Supported width: 128 to 2048 pixels.

24.81.2 Member Function Documentation

24.81.2.1 APEXCV_LIB_RESULT apexcv::SobelYFilter::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3 or 5
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8UC1).

24.81.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.81.2.3 APEXCV_LIB_RESULT apexcv::SobelYFilter::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.81.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.82 apexcv::SobelYFilterHT Class Reference

Sobel Y filter, Hand Tuned (optimized).

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, int aWindowSize, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.82.1 Detailed Description

Sobel Y filter, Hand Tuned (optimized).

Object of this class applies a Sobel Y filter to *aSrc*. This is a hand tuned (HT) implementation providing faster processing times. Supported window size: 3 x 3
aDst and *aSrc* must have identical dimensions.
Supported input type: VSDK_CV_8UC1, output type: VSDK_CV_8SC1.
Supported width: 128 to 2048 pixels.

24.82.2 Member Function Documentation

24.82.2.1 APEXCV_LIB_RESULT apexcv::SobelYFilterHT::Initialize (vsdk::SUMat & *aSrc,* int *aWindowSize,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source Image buffer (VSDK_CV_8UC1).
`in`	*aWindowSize*	Window Size, 3
`in,out`	*aDst*	Destination Image buffer (VSDK_CV_8SC1).

24.82.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) `[inherited]`

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.82.2.3 APEXCV_LIB_RESULT apexcv::SobelYFilterHT::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source image buffer (VSDK_CV_8UC1).
`in,out`	*aDst*	Destination image buffer (VSDK_CV_8SC1).

24.82.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) `[inherited]`

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.83 apexcv::SplitChannel Class Reference

Channel split class containing support for spliting a single channel from a multi-channel image.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst1, vsdk::SUMat &aDst2, vsdk::SUMat &aDst3, vsdk::SUMat &aDst4)

  Channel Split function.
- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst1, vsdk::SUMat &aDst2, vsdk::SUMat &aDst3)

  Channel Split function.
- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &aDst1, vsdk::SUMat &aDst2)

  Channel Split function.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst1, vsdk::SUMat &aDst2, vsdk::↩ SUMat &aDst3, vsdk::SUMat &aDst4)

Reconnect IO.

- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst1, vsdk::SUMat &aDst2, vsdk::↩ SUMat &aDst3)

  Reconnect IO.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst1, vsdk::SUMat &aDst2)

  Reconnect IO.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.83.1 Detailed Description

Channel split class containing support for spliting a single channel from a multi-channel image.

This class is an interface for using channel split functions on the host.

24.83.2 Member Function Documentation

24.83.2.1 APEXCV_LIB_RESULT apexcv::SplitChannel::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst1,* vsdk::SUMat & *aDst2,* vsdk::SUMat & *aDst3,* vsdk::SUMat & *aDst4*)

Channel Split function.

Splits a channel from a multiple channel image.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC4
`in,out`	*aDst1*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
`in,out`	*aDst2*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
`in,out`	*aDst3*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
`in,out`	*aDst4*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.83.2.2 APEXCV_LIB_RESULT apexcv::SplitChannel::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst1,* vsdk::SUMat & *aDst2,* vsdk::SUMat & *aDst3*)

Channel Split function.

Splits a channel from a multiple channel image.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC3
in,out	*aDst1*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
in,out	*aDst2*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
in,out	*aDst3*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.83.2.3 APEXCV_LIB_RESULT apexcv::SplitChannel::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst1,* vsdk::SUMat & *aDst2*)

Channel Split function.

Splits a channel from a multiple channel image.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC2
in,out	*aDst1*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
	aDst2	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.83.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.83.2.5 APEXCV_LIB_RESULT apexcv::SplitChannel::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst1,* vsdk::SUMat & *aDst2,* vsdk::SUMat & *aDst3,* vsdk::SUMat & *aDst4*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC4

Parameters

in,out	*aDst1*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
in,out	*aDst2*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
in,out	*aDst3*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
in,out	*aDst4*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.83.2.6 APEXCV_LIB_RESULT apexcv::SplitChannel::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst1*, vsdk::SUMat & *aDst2*, vsdk::SUMat & *aDst3*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC3
in,out	*aDst1*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
in,out	*aDst2*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
in,out	*aDst3*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.83.2.7 APEXCV_LIB_RESULT apexcv::SplitChannel::ReconnectIO (vsdk::SUMat & *aSrc*, vsdk::SUMat & *aDst1*, vsdk::SUMat & *aDst2*)

Reconnect IO.

This function allows to change the Input and Output images without re-initializing

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer. Accepted buffer types are VSDK_CV_8UC2
in,out	*aDst1*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1
in,out	*aDst2*	Destination memory buffer. Accepted buffer type is VSDK_CV_8UC1

24.83.2.8 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a← ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.84 apexcv::Subtract Class Reference

Subtract.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc1, vsdk::SUMat &aSrc2, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetPolicy (apexcv::eConvertPolicy aPolicy)

  Set Policy type.
- APEXCV_LIB_RESULT GetPolicy (apexcv::eConvertPolicy &aPolicy)

  Get Policy type.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.84.1 Detailed Description

Subtract.

Object of this class subtracts the pixel value of *aSrc2* from *aSrc1* pixel by pixel. *aDst* can be VSDK_CV_8UC1 only if both source images are VSDK_CV_8UC1 and *aDst* is explicitly set to VSDK_CV_8UC1. It is otherwise VSDK_CV_←16SC1.
Supported input 1 type: VSDK_CV_8UC1, input 2 type: VSDK_CV_8UC1, output type: VSDK_CV_8UC1 or
Supported input 1 type: VSDK_CV_8UC1, input 2 type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1 or
Supported input 1 type: VSDK_CV_8UC1, input 2 type: VSDK_CV_16SC1, output type: VSDK_CV_16SC1 or
Supported input 1 type: VSDK_CV_16SC1, input 2 type: VSDK_CV_8UC1, output type: VSDK_CV_16SC1 or
Supported input 1 type: VSDK_CV_16SC1, input 2 type: VSDK_CV_16SC1, output type: VSDK_CV_16SC1
Supported width: 128 to 2048 pixels.

24.84.2 Member Function Documentation

24.84.2.1 APEXCV_LIB_RESULT apexcv::Subtract::GetPolicy (apexcv::eConvertPolicy & *aPolicy*)

Get Policy type.

This function allows to read the value of the overflow policy type.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

out	*aPolicy*	Overflow policy type.

24.84.2.2 APEXCV_LIB_RESULT apexcv::Subtract::Initialize (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.84.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.84.2.4 APEXCV_LIB_RESULT apexcv::Subtract::ReconnectIO (vsdk::SUMat & *aSrc1*, vsdk::SUMat & *aSrc2*, vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc1*	Source image buffer 1 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in	*aSrc2*	Source image buffer 2 (VSDK_CV_8UC1, VSDK_CV_16SC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1, VSDK_CV_16SC1).

24.84.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.84.2.6 APEXCV_LIB_RESULT apexcv::Subtract::SetPolicy (apexcv::eConvertPolicy *aPolicy*)

Set Policy type.

This function allows to change the overflow policy type.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aPolicy*	Overflow policy type

24.85 apexcv::Hog::SVM Class Reference

Class to compute SVM detector from Hog blocks.

Public Member Functions

- APEXCV_LIB_RESULT SetConfig (const Config &aConfig)

  Sets the configuration for the SVM class.
- void GetConfig (Config &aConfig) const

  Gets the configuration for the class.
- APEXCV_LIB_RESULT Initialize (const vsdk::SUMat &aSrc, const vsdk::SUMat &aSVM, vsdk::SUMat &aDst)

  Initialization of the SVM object.
- APEXCV_LIB_RESULT ReconnectIO (const vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.85.1 Detailed Description

Class to compute SVM detector from Hog blocks.

apexcv::Hog::SVM is used in case of separate detection, when the apexcv::Hog class compute the hog blocks only. HOG Object Detector

The process takes an 8-bin normalized block histogram computed every 4x4 pixels. It multiplies the block histograms with SVM weights for each detection window and returns the scores for the object detection.

24.85.2 Member Function Documentation

24.85.2.1 void apexcv::Hog::SVM::GetConfig (Config & *aConfig*) const

Gets the configuration for the class.

Returns the HOG parameters for the linear SVM classifier.

24.85.2.2 APEXCV_LIB_RESULT apexcv::Hog::SVM::Initialize (const vsdk::SUMat & *aSrc,* const vsdk::SUMat & *aSVM,* vsdk::SUMat & *aDst*)

Initialization of the SVM object.

Returns

Error code for ACF process initialization, APEXCV_SUCCESS on success.

Prepares the object for execution. Initialization is done at this stage. For the size of the arguments, please refer to ReconnectIO() of Hog class

Parameters

`in`	*aSrc*	Blocks histograms, output of Hog::Process(), VSDK_CV_8UC1
`in`	*aSVM*	Vector of descriptorSize + 1 elements of type double, VSDK_CV_64FC1.
`in,out`	*aDst*	Scores from HOG object detection, VSDK_CV_16SC1.

24.85.2.3 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.85.2.4 APEXCV_LIB_RESULT apexcv::Hog::SVM::ReconnectIO (const vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO.

Use this to reconnect the input and output buffers and allocate the output. This only needs to be done if the connected Input/Outputs are changed. If only the data within changes (no size, data pointer, or type changes), then this does not need to be called. This function will allocation aDst if it has not been pre-allocated with the right dimensions. For the size of the arguments, please refer to ReconnectIO() of Hog class

Parameters

`in`	*aSrc*	Blocks histograms, output of Hog::Process(), VSDK_CV_8UC1.
`in,out`	*aDst*	Scores from HOG object detection, VSDK_CV_16SC1.

24.85.2.5 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.85.2.6 APEXCV_LIB_RESULT apexcv::Hog::SVM::SetConfig (const Config & *aConfig*)

Sets the configuration for the SVM class.

Set the HOG parameters for the linear SVM classifier. It is used when changing from default setting only. It should be called before Initialize().

The size of the detection window should not be bigger than 128x128 pixels. If some dimension is not integer multiple of the HOG block size, it is rounded down per the block size.

24.86 apexcv::TableLookup Class Reference

Table Lookup.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, vsdk::SUMat &acLut, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &acLut, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.86.1 Detailed Description

Table Lookup.

Object of this class translates the pixel value of *aSrc* through the lookup table *acLut* pixel by pixel.
Supported input type: VSDK_CV_8UC1
Supported width: 128 to 2048 pixels.

24.86.2 Member Function Documentation

24.86.2.1 APEXCV_LIB_RESULT apexcv::TableLookup::Initialize (vsdk::SUMat & *aSrc,* vsdk::SUMat & *acLut,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in	*acLut*	Look-up table for the transformation (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.86.2.2 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.86.2.3 APEXCV_LIB_RESULT apexcv::TableLookup::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *acLut,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in	*acLut*	Look-up table for the transformation (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.86.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.87 apexcv::Threshold Class Reference

Threshold.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, const uint32_t aThreshold, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).

- APEXCV_LIB_RESULT SetThreshold (const uint32_t acThreshold)

  Set Threshold.
- APEXCV_LIB_RESULT GetThreshold (uint32_t &aThreshold)

  Get Threshold.
- APEXCV_LIB_RESULT SetOutputValues (const uint8_t acTrueVal, const uint8_t acFalseVal)

  Set Output Values.
- APEXCV_LIB_RESULT GetOutputValues (uint8_t &aTrueVal, uint8_t &aFalseVal)

  Get Output Values.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.87.1 Detailed Description

Threshold.

Object of this class thresholds the pixel value of *aSrc* with the value of *aThreshold* pixel by pixel.
True when *aSrc*(x,y) $>$ *aThreshold*, otherwise false.
Default output values are 255 when true and 0 when false.
Supported input type: VSDK_CV_8UC1
Supported width: 128 to 2048 pixels.

24.87.2 Member Function Documentation

24.87.2.1 APEXCV_LIB_RESULT apexcv::Threshold::GetOutputValues (uint8_t & *aTrueVal,* uint8_t & *aFalseVal*)

Get Output Values.

This function allows to read the low and high output values.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

out	*aTrueVal*	true output value.
out	*aFalseVal*	false output value.

24.87.2.2 APEXCV_LIB_RESULT apexcv::Threshold::GetThreshold (uint32_t & *aThreshold*)

Get Threshold.

This function allows to read the value of the threshold.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

out	*aThreshold*	threshold.

24.87.2.3 APEXCV_LIB_RESULT apexcv::Threshold::Initialize (vsdk::SUMat & *aSrc,* const uint32_t *aThreshold,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in	*aThreshold*	Threshold value.
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.87.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.87.2.5 APEXCV_LIB_RESULT apexcv::Threshold::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source image buffer (VSDK_CV_8UC1).
in,out	*aDst*	Destination image buffer (VSDK_CV_8UC1).

24.87.2.6 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↵ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.87.2.7 APEXCV_LIB_RESULT apexcv::Threshold::SetOutputValues (const uint8_t *acTrueVal,* const uint8_t *acFalseVal*)

Set Output Values.

This function allows to change the low and high output values.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*acTrueVal*	true output value.
in	*acFalseVal*	false output value.

24.87.2.8 APEXCV_LIB_RESULT apexcv::Threshold::SetThreshold (const uint32_t *acThreshold*)

Set Threshold.

This function allows to change the value of the threshold.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*acThreshold*	threshold.

24.88 apexcv::ThresholdRange Class Reference

Threshold Range.

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &aSrc, const uint32_t acLowThreshold, const uint32_t acHighThreshold, vsdk::SUMat &aDst)

  Initialize object (required).
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &aSrc, vsdk::SUMat &aDst)

  Reconnect IO (optional).
- APEXCV_LIB_RESULT SetThresholds (const uint32_t acLowThreshold, const uint32_t acHighThreshold)

  Set Thresholds.
- APEXCV_LIB_RESULT GetThresholds (uint32_t &aLowThreshold, uint32_t &aHighThreshold)

  Get Thresholds.
- APEXCV_LIB_RESULT SetOutputValues (const uint8_t acTrueVal, const uint8_t acFalseVal)

  SetOutputValues.
- APEXCV_LIB_RESULT GetOutputValues (uint8_t &aTrueVal, uint8_t &aFalseVal)

  Get Output Values.
- APEXCV_LIB_RESULT **SelectApexCore** (int aApexId)

  Select the APEX Core.
- APEXCV_LIB_RESULT **Process** ()

  Start processing and return when done.

24.88.1 Detailed Description

Threshold Range.

Object of this class thresholds the pixel value of *aSrc* with the following scheme pixel by pixel.
False when *aSrc*(x,y) $>$ *acHighThreshold*, False when *aSrc*(x,y) $<$ *acLowThreshold*, otherwise true.
Default output values are 255 when true and 0 when false.
Supported input type: VSDK_CV_8UC1
Supported width: 128 to 2048 pixels.

24.88.2 Member Function Documentation

24.88.2.1 APEXCV_LIB_RESULT apexcv::ThresholdRange::GetOutputValues (uint8_t & *aTrueVal,* uint8_t & *aFalseVal*)

Get Output Values.

This function allows to read the true and false output values.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

out	*aTrueVal*	true output value.
out	*aFalseVal*	false output value.

24.88.2.2 APEXCV_LIB_RESULT apexcv::ThresholdRange::GetThresholds (uint32_t & *aLowThreshold,* uint32_t & *aHighThreshold*)

Get Thresholds.

This function allows to read the value of the thresholds.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

out	*aLowThreshold*	low threshold.
out	*aHighThreshold*	high threshold.

24.88.2.3 APEXCV_LIB_RESULT apexcv::ThresholdRange::Initialize (vsdk::SUMat & *aSrc,* const uint32_t *acLowThreshold,* const uint32_t *acHighThreshold,* vsdk::SUMat & *aDst*)

Initialize object (required).

This function initializes the object. The function Process() can be called to execute the processing on the APEX core. To process another image buffer, use ReconnectIO(...).

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer.
in	*acLowThreshold*	Low Threshold value.
in	*acHighThreshold*	High Threshold value.
in,out	*aDst*	Destination memory buffer.

24.88.2.4 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::Process ( ) [inherited]

Start processing and return when done.

Execute code on selected APEX core (default is Apex core 0). This function is called after initialize() and is executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

24.88.2.5 APEXCV_LIB_RESULT apexcv::ThresholdRange::ReconnectIO (vsdk::SUMat & *aSrc,* vsdk::SUMat & *aDst*)

Reconnect IO (optional).

This function allows to change the Input and Output images without re-initializing.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*aSrc*	Source memory buffer.
in,out	*aDst*	Destination memory buffer.

24.88.2.6 APEXCV_LIB_RESULT apexcv::ApexcvHostBaseClass::SelectApexCore (int *aApexId*) [inherited]

Select the APEX Core.

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after initialize() and can be executed on a per frame base.

Returns

APEXCV Error code (APEXCV_SUCCESS on success).

Parameters

a↩ ApexId	ID of the APEX core used for performing the processing (0 or 1).

24.88.2.7 APEXCV_LIB_RESULT apexcv::ThresholdRange::SetOutputValues (const uint8_t *acTrueVal,* const uint8_t *acFalseVal*)

SetOutputValues.

This function allows to change the true and false output values.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

in	*acTrueVal*	true output value.
in	*acFalseVal*	false output value.

24.88.2.8 APEXCV_LIB_RESULT apexcv::ThresholdRange::SetThresholds (const uint32_t *acLowThreshold,* const uint32_t *acHighThreshold*)

Set Thresholds.

This function allows to change the value of the thresholds.

Returns

APEXCV_LIB_RESULT Error code.

Parameters

`in`	*acLowThreshold*	low threshold.
`in`	*acHighThreshold*	high threshold.

24.89 apexcv::Tmo Class Reference

TMO class This class is an interface for using the tone mapping algorithm on the APEX.

Public Types

Public Member Functions

- APEXCV_LIB_RESULT Initialize (vsdk::SUMat &arHdrImage, HDR_IMAGE_FORMAT const acHdrImageFormat, vsdk::SUMat &arLdrImage, vsdk::SUMat &arLdrTransformKey)

  Initializes TMO process, Connects the buffer to the process port, and allocates/initializes any internal buffers.
- APEXCV_LIB_RESULT ReconnectIO (vsdk::SUMat &arHdrImage, HDR_IMAGE_FORMAT const acHdrImage↩ Format, vsdk::SUMat &arLdrImage, vsdk::SUMat &arLdrTransformKey)

  Reconnects the input/output to TMO process for RGBE input.
- APEXCV_LIB_RESULT Process ()

  Run APEX-TMO process.
- APEXCV_LIB_RESULT SelectApexCore (int32_t aApexId)

  Select APEX Core for processing.

24.89.1 Detailed Description

TMO class This class is an interface for using the tone mapping algorithm on the APEX.

24.89.2 Member Enumeration Documentation

24.89.2.1 enum **apexcv::Tmo::HDR_IMAGE_FORMAT**

HDR Image Format.

Enumerator

HDR_IMAGE_FORMAT_INVALID Invalid type

HDR_IMAGE_FORMAT_RGBE Datatype is 08U. e0 size is (4, 1). Channel Order is RGBE

HDR_IMAGE_FORMAT_OPENEXR Datatype is 16U. e0 size is (3, 1). Channel Order is RGB

24.89.3 Member Function Documentation

24.89.3.1 APEXCV_LIB_RESULT apexcv::Tmo::Initialize (vsdk::SUMat & *arHdrImage*, HDR_IMAGE_FORMAT const *acHdrImageFormat*, vsdk::SUMat & *arLdrImage*, vsdk::SUMat & *arLdrTransformKey*)

Initializes TMO process, Connects the buffer to the process port, and allocates/initializes any internal buffers.

Parameters

arHdrImage	Input image. Datatype must correspond to "acHdrImageFormat"
acHdrImageFormat	Input image format. "arHdrImage" datatype must correspond to "acHdrImageFormat"
arLdrImage	Output image. Datatype is 08U. e0 size is (3, 1).
arLdrTransformKey	Key value used during LDR transform. Datatype is 08U.

24.89.3.2 APEXCV_LIB_RESULT apexcv::Tmo::Process ( )

Run APEX-TMO process.

Generate LDR image from HDR image.

Supported datatypes are:

- unsigned 8 bit image to unsigned 8 bit image (RGBE input)
- unsigned 16 bit image to unsigned 8 bit image (OpenEXR input)

Returns

Error code (zero on success).

24.89.3.3 APEXCV_LIB_RESULT apexcv::Tmo::ReconnectIO (vsdk::SUMat & *arHdrImage*, HDR_IMAGE_FORMAT const *acHdrImageFormat*, vsdk::SUMat & *arLdrImage*, vsdk::SUMat & *arLdrTransformKey*)

Reconnects the input/output to TMO process for RGBE input.

Parameters

arHdrImage	Input image. Datatype must correspond to "acHdrImageFormat"
acHdrImageFormat	Input image format. "arHdrImage" datatype must correspond to "acHdrImageFormat"
arLdrImage	Output image. Datatype is 08U. e0 size is (3, 1).
arLdrTransformKey	Key value used during LDR transform. Datatype is 08U.

24.89.3.4 APEXCV_LIB_RESULT apexcv::Tmo::SelectApexCore (int32_t *aApexId*)

Select APEX Core for processing.

Returns

Error code for the ACF configuration (APEXCV_SUCCESS on success).

Select which APEX core (0 or 1) to be selected to run the processing. This function has to be called after Initialize() and can be executed per frame base.

Parameters

a↩ ApexId	The ID of the desired APEX (e.g if there are 2 APEXs, valid values for aApexId would be 0 and 1).

Bibliography

[1] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005. 13

[2] Toshiyuki Dobashi, Atsushi Tashiro, Masahiro Iwahashi, and Hitoshi Kiya. A fixed-point implementation of tone mapping operation for hdr images expressed in floating-point format. *APSIPA Transactions on Signal and Information Processing*, 3, 2014. 26

[3] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972. 15

[4] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988. 10, 115, 116, 118

[5] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. 27

[6] Edward Rosten and Tom Drummond. Machine learning for high speed corner detection. In *In 9th European Conference on Computer Vision*, volume 1, pages 430–443, 2006. 18

[7] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, June 1994. 10, 115, 116

[8] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846, Jan 1998. 50

Index