TOPPAN

senSPure™ SDK

TOPPAN ToF SDK PostFilter Library Reference Manual

TOPPAN 3D ToF Camera

TOPPAN Holdings Inc.

Version 1.12

June 24, 2025

Contents

1. Overview

1-1. Purpose of this guide

This manual describes the application program interface (API) specifications of the PostFilter library that is included in the TOPPAN ToF SDK.

Currently, the following cameras are operational targets:

Table 1. Supported camera model

Model	Product code	Camera firmware version
C11U	TPSC1AS1Z	3.1.0 or higher

1-2. Definitions of terms, abbreviations

Table 2. Definitions of terms, abbreviations

Terms, Abbreviations	Definition
SDK	Software Development Kit
ToF	Time of Flight

1-3. Related documents

When referring to this document, please also refer to the following related latest documents.

Table 3. Related Documents

Related Documents	Contents
TOPPAN ToF SDK Library Development Environment Setup Guide	Setup guide of the TOPPAN ToF SDK environment
TOPPAN ToF SDK Library API Reference Manual	API specifications of the TOPPAN ToF SDK library For details about methods and classes marked with an asterisk (*), see this document.

1-4. SDK structure

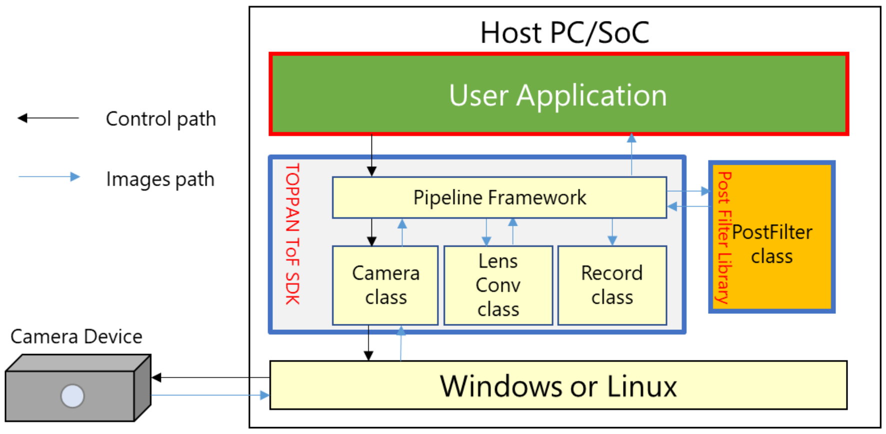

Figure 1. Software structure diagram

Table 4. Block description

Block		Description
User Application		Application that controls this SDK and/or a sample application provided in this SDK
TOPPAN ToF SDK (Library)		Library provided as the SDK *For details about the API of this library, see "TOPPAN ToF SDK Library API Reference Manual."
	Pipeline Framework	A framework that manages the Camera class, Record class, and LensConv class together, and can construct Pipeline processing including arbitrary processing other than these (OSS-linked processing and user-specific processing)
	Camera Class	C++ class for controlling camera devices and/or playback video stored files
	Lens Conv Class	C++ class for lens-related conversion of ToF camera output images
	Record Class	C++ class for saving camera device output images to a file
PostFilter (Library)		This Library that cooperates with the TOPPAN ToF SDK C++ class that provides image filtering
	PostFilter class	C++ class that performs filtering on the image output from the camera device

1-5. System requirements

The PostFilter library has been tested in the following environments.

Table 5. System Requirements

Environment		OS Type	Version
PC		Windows	Windows 11/10 64bit
		Linux	Ubuntu 20.04LTS 64bit
SoC	NVIDIA Jetson AGX Orin	Linux	Ubuntu 20.04LTS 64bit, Jetpack 5.0.1

1-6. Recommended environment on the host PC

The recommended environment on the host PC to run the PostFilter library is described below.

Table 6. Recommended environment

Hardware	Recommended environment
CPU	4-core, 2 GHz or higher, 64 bit CPU
Memory	8GB or higher
Physical Interface	USB3.1 (Gen1) port

Note: If the Pipeline Framework requires multi-threaded configuration and high frame rate processing, a host PC with higher specifications in terms of CPU core count and frequency will be needed.

1-6-1. Programming language

This PostFilter library is developed in C++ (C++17 standard). However, MISRA-C++ and CERT-C++ are not supported.

2. PostFilter API usage

2-1. Provided files

When using the API of this PostFilter library, include the following header file and link the library file according to the class to be used.
The following files are stored in the "lib/include" directory generated after the build. For the configuration of the provided directory, see "TOPPAN ToF SDK Library Development Environment Setup Guide" described in "***1-3. Related documents***".

Table 7. Provided files

Provided file	Provided file	
Header file	PostFilterThread class	PostFilterThread.h
	PostFilter class	PostFilter.h
Library file	Windows	PostFilter.dll
	Linux	libPostFilter.so

2-2. Related library files

The following library files are referenced inside this PostFilter library. A user program employing this PostFilter library requires the above library files.

Table 8. Related library

OS type	Related library
Common	OpenCV, boost, TOPPAN ToF SDK(TptofSdk)

EWCLIB (http://insubaru.g1.xrea.com/ewclib/) is included and used in the Windows version.

3. Provided API for PostFilter

3-1. List of classes

The list of classes provided as API by this PostFilter library is shown below.

Table 9. List of classes

Class name	Description
PostFilterThread	C++ class that is a processing thread on a Pipeline using the PostFilter class. By incorporating it into the Pipeline Framework, performs filtering in the Pipeline.
PostFilter	C++ class that performs filtering on the camera output image *This class is prohibited when using the PipelineFramework.

3-2. PostFilterThread class

3-2-1. Overview

Table 10. PostFilterThread class overview

Provided header files	PostFilterThread.h	PostFilterThread class definition
	PostFilterThreadType.h	Event type definition for PostFilterThread class
Member namespace	krm	
Description	Thread that performs filtering processing *For details about the methods that inherit EvtThread, see the description of the EvtThread class.	
Additional buffer	Unused	

3-2-2. Provided functions

A summary of the functions provided by the PostFilterThread class is as follows.

Table *11*. Functional overview of PostFilterThread class

Class name	Description
Median filter function **Bilateral filter function** **Flying pixel filter function**	Median, bilateral, and flying pixel filtering using the PostFilter class. For this function, refer to the following chapter. (See ***3-2-2. Provided functions***)

3-2-3. Control sequence

This section describes the camera control sequence using the *PipelineFramework class. In the following sequence, the judgment of the abnormal system such as the return value judgment of the function is omitted.

For the initialization sequence, information acquisition & operation mode switching sequence, and finalization sequence, refer to the TOPPAN ToF SDK API Reference Manual described in "***1-3. Related documents***".

3-2-3-1. Filtering processing sequence

The following shows the setting of parameters for filtering processing and the filtering processing sequence using *PlFw:: notifyEvent(). Since the same method is called for all threads registered as Pipeline, it is omitted in this sequence.

The notification of the parameter setting event for filtering processing (EV_POSTFILT_PRM) must be performed before receiving the image.

EV_POSTFILT_PRM is also reported to PostFilterThread and subsequent threads, but no processing is performed.

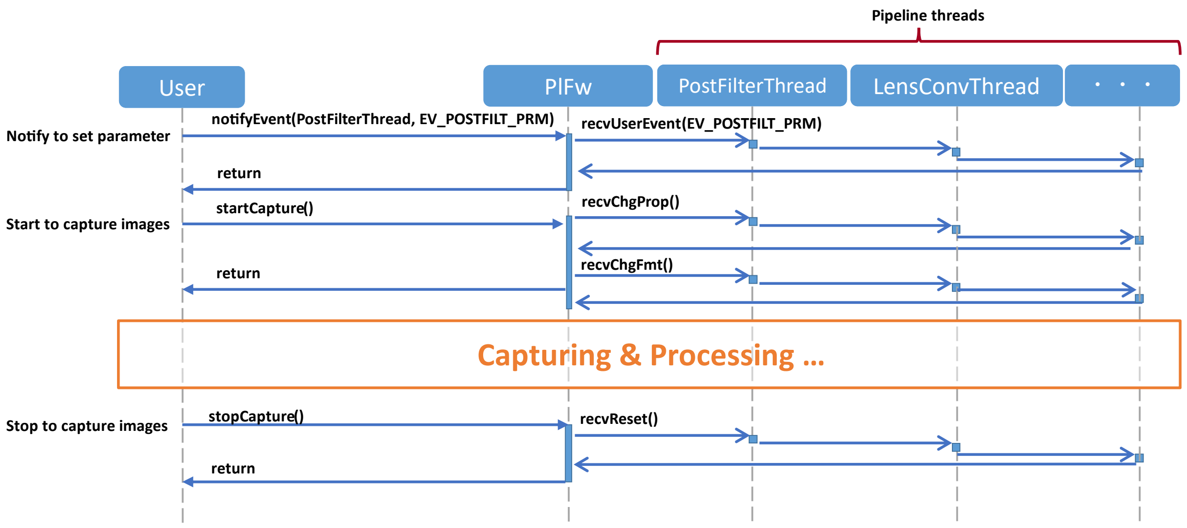

Figure 2. Filtering Processing Sequence

3-2-4. List of methods

Table 12. PostFilterThread class method

Method name	Description
PostFilterThread:: PostFilterThread	Constructor

3-2-4-1. Method details

3-2-4-1-1. PostFilterThread::PostFilterThread

Table 13. PostFilterThread::PostFilterThread Method

Function	Constructor
Definition	PostFilterThread (bool enable_medf = true, bool enable_bilf = true, bool enable_flypf = true)
Description	• Initializes a thread for filtering.

	• If the enable_medf argument is enabled, the depth and IR images in the image frame output by FrameData:: getFrame() will be output with the median filter applied. You can also use PlFw:: notifyEvent (EV_POSTFILT_MEDF) to change the setting. • If the enable_bilf argument is enabled, the depth and IR images in the image frame output by FrameData:: getFrame() will be output with the bilateral filter applied. You can also use PlFw:: notifyEvent (EV_POSTFILT_BILF) to change the setting. • When the enable_flypf argument is set to ON, the flying pixel filtered image is output to the Depth image in the image frame output by FrameData:: getFrame(). You can also use PlFw:: notifyEvent (EV_POSTFILT_FLYPF) to change the setting. • If the input Depth image has the median filter applied, the median filter is disabled regardless of the enable_medf setting. • If the input Depth image has the bilateral filter applied, the bilateral filter is disabled regardless of the enable_bilf setting. • If the input Depth image has the flying pixel filter applied, the flying pixel filter is disabled regardless of the enable_flypf setting.			
Arguments	**Type**	**Name**	**in/out**	**Description**
	bool	enable_medf	in	Enable median filter true : Enabled (initial value) false : Disabled
	bool	enable_bilf	in	Enable Bilateral Filter true : Enabled (initial value) false : Disabled
	bool	enable_flypf	in	Enable Flying Pixel Filter true : Enabled (initial value) false : Disabled
Return value	None			
Sync/Async	Synchronous			

3-2-5. List of Events

The following events can be used with *PlFw:: notifyEvent().

Table 14. PostFilterThread class events

Event name	Description
EV_POSTFILT_PSBL_MEDF	Notification of whether a median filter can be applied
EV_POSTFILT_PSBL_BILF	Notification of whether a bilateral filter can be applied
EV_POSTFILT_PSBL_FLYPF	Notification of whether a flying pixel filter can be applied
EV_POSTFILT_MEDF	ON/OFF of the median filter
EV_POSTFILT_BILF	ON/OFF of the bilateral filter
EV_POSTFILT_FLYPF	ON/OFF of the flying pixel filter
EV_POSTFILT_PRM	Settings for filtering parameters

3-2-5-1-1. EV_POSTFILT_PSBL_MEDF

Table 15. EV_POSTFILT_PSBL_MEDF overview

Information	Acquisition of information on whether a median filter can be applied or not
Argument	bool
Description	・ Acquires information about whether a median filter is applicable. ・ This is false if a median filter is applied in the camera device.

3-2-5-1-2. EV_POSTFILT_PSBL_BILF

Table 16. EV_POSTFILT_PSBL_BILF overview

Information	Acquisition of information on whether a bilateral filter can be applied or not
Argument	bool
Description	・ Acquires information about whether a bilateral filter is applicable. ・ This is false if a bilateral filter is applied in the camera device.

3-2-5-1-3. EV_POSTFILT_PSBL_FLYPF

Table 17. EV_POSTFILT_PSBL_FLYPF overview

Information	Acquisition of information on whether a frying pixel filter can be applied or not
Argument	bool
Description	・ Acquisition of information about whether a flying pixel filter is applicable. ・ false if a flying pixel filter is implemented in the camera device.

3-2-5-1-4. EV_POSTFILT_MEDF

Table 18. EV_POSTFILT_MEDF overview

Information	Median filter ON/OFF
Argument	bool
Description	・ When the argument is true, the median filter is ON. When false, the median filter is OFF. ・ When the median filter has been applied to the input Depth image and IR image, the median filter is disabled regardless of the argument. ・ This parameter is initially set to the enable_ medf argument of PostFilterThread::PostFilterThread().

3-2-5-1-5. EV_POSTFILT_BILF

Table 19. EV_POSTFILT_BILF overview

Information	Bilateral filter ON/OFF
Argument	bool
Description	・ If the argument is true, the bilateral filter is turned ON. If false, the bilateral filter is turned OFF. ・ When the input Depth image and IR image are data to which the bilateral filter has been applied, the bilateral filter is disabled regardless of the arguments. ・ This function is initially set to the enable_ bilf argument of PostFilterThread::PostFilterThread().

3-2-5-1-6. EV_POSTFILT_FLYPF

Table 20. EV_POSTFILT_FLYPF overview

Information	Frying pixel filter ON/OFF
Argument	bool
Description	・ If the argument is true, the flying pixel filter is turned ON. If false, the flying pixel filter is turned OFF. ・ If the input Depth image has the flying pixel filter applied, the flying pixel filter is disabled regardless of the argument. ・ This parameter is initially set to the enable_ flypf argument of PostFilterThread::PostFilterThread().

3-2-5-1-7. EV_POSTFILT_PRM

Table 21. EV_POSTFILT_PRM overview

Information	Setting the parameters for filtering processing
Argument	PostFilterPrm
Description	・ Sets parameters for filtering processing.

3-2-6. Status notification

If a status notification occurs during processing by the PostFilterThread, the following values are notified to the PlFw:: getEvent() or PlFw:: addPlProc() callback function.

Table 22. Status notification of PostFilterThread class

Notified value	Description
ERR_INVALID_PTR	Filtering failed due to invalid pointer of notification parameter
ERR_BAD_ARG	Error due to invalid image format or out-of-range filtering parameter setting
ERR_BAD_STATE	Filtering failed due to state transition error
ERR_SYSTEM	Filtering failed due to system error
ERR_NOT_SUPPORT	Filtering failed due to unsupported image

3-2-7. PostFilterThread class definition

The definitions used as event notifications to PostFilterThread, described in PostFilterEvent.h, are shown below.

3-2-7-1. Enumeration definitions

3-2-7-1-1. List of enumeration definitions

Table 23. List of enumeration definitions

Name	Description
PostFilterEvent	Event ID of the event notification for the PostFilterThread

3-2-7-2. EnumerationDefinitionDetails

3-2-7-2-1. PostFilterEvent

Table 24. PostFilterEvent details

Definition	enum PostFilterEvent: uint8_t {
	EV_POSTFILT_PSBL_MEDF,
	EV_POSTFILT_PSBL_BILF,
	EV_POSTFILT_PSBL_FLYPF,
	EV_POSTFILT_MEDF,
	EV_POSTFILT_BILF,
	EV_POSTFILT_FLYPF,
	EV_POSTFILT_PRM,
	};		

Description	・ Indicates the event ID of the event notification for PostFilter.		
Return value	**Name**	**Value**	**Description**
	EV_POSTFILT_PSBL_MEDF	0	Notification of whether a median filter can be applied
	EV_POSTFILT_PSBL_BILF	1	Notification of whether a bilateral filter can be applied
	EV_POSTFILT_PSBL_FLYPF	2	Notification of whether the flying pixel filter can be applied
	EV_POSTFILT_MEDF	3	ON/OFF of the median filter
	EV_POSTFILT_BILF	4	ON/OFF of the bilateral filter
	EV_POSTFILT_FLYPF	5	ON/OFF of the flying pixel filter
	EV_POSTFILT_PRM	6	Settings for filtering parameters
Reference	※PlFw::notifyEvent()		

3-3. PostFilter class

3-3-1. Overview

Table 25. PostFilter class overview

Provided head file	PostFilter.h	PostFilter class definition
	PostFilterType.h	PostFilter class type definition
Member namespace	krm	
Description	C++ class that performs filtering operations on camera device output images.	
Thread Safety	The PostFilter class is not thread-safe. Therefore, user programs that use the PostFilter class must perform exclusive processing as necessary.	

3-3-2. Provided functions

3-3-2-1. Function overview

The following provides an overview of the functions provided by the PostFilter class.

Table 26. PostFilter class functions

Provided Function	Description
Median filter	This function reduces random noise (point defects) in image data. Applies a median filter to the depth or IR image received as input, and outputs the resulting image. Median filtering uses filtering parameters to perform conversion processing.
Bilateral filter	Removes noise while preserving image edges. Applies a bilateral filter to an input depth image or IR image and outputs the resulting image. Bilateral filtering uses filtering parameters to perform conversion processing.
Frying pixel filter	Removes flying pixels near the boundaries of the subject in a depth image. Applies the flying pixel filter to the received depth image and outputs the resulting image. The flying pixel filter performs a conversion process using filtering parameters.

3-3-2-1-1. Median filter processing function

Removes noise by applying a median filter to the Depth and IR images. The kernel size is 3×3 pixels or 5×5 pixels

3-3-2-1-2. Bilateral filter processing function

Removes noise while preserving edges by applying a bilateral filter to the Depth and IR images. The kernel size is 3×3 pixels or 5×5 pixels. Exception processing is performed for invalid pixels.

3-3-2-1-3. Frying pixel filter processing function

Depth Removes flying pixels in the image. A flying pixel is a noise pixel located near the edge of the Depth image and has a value midway between the foreground and background objects. Removed flying pixels are replaced with invalid pixels.
There are two algorithms for the flying pixel filter: Differential and Ratio. Differential removes more flying pixels but tends to remove more invalid pixels. Ratio removes flying pixels while reducing the chance of invalid pixels replacing valid, non-flying pixels. The flying pixel filter can also be selected for speed or accuracy.

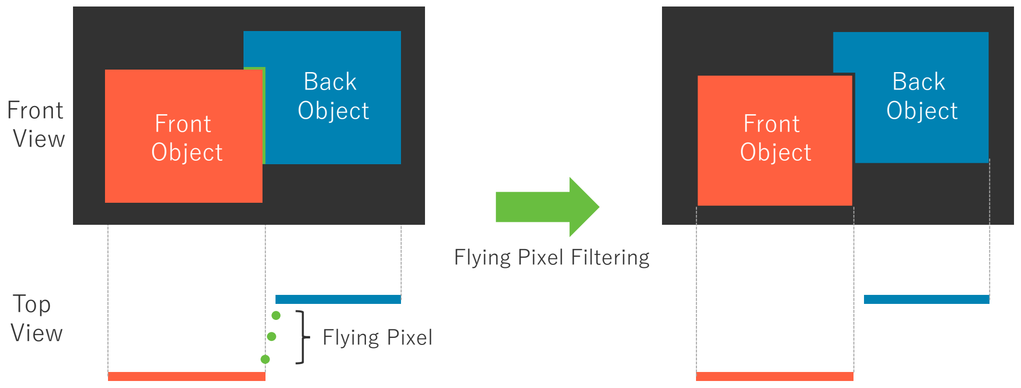

Figure 3. Frying pixel filter

3-3-3. List of methods

Table 27. PostFilter class methods

Method name	Description
PostFilter	Constructor
~PostFilter	Destructor
setPostFilterPrm	Setting parameters for filtering
setFormat	Image format information setting
filterMedian	median filter execution
filterBilateral	bilateral filter execution
filterFlyingPixel	flying pixel filter execution

3-3-3-1. State machine

The PostFilter class state transitions are as follows.

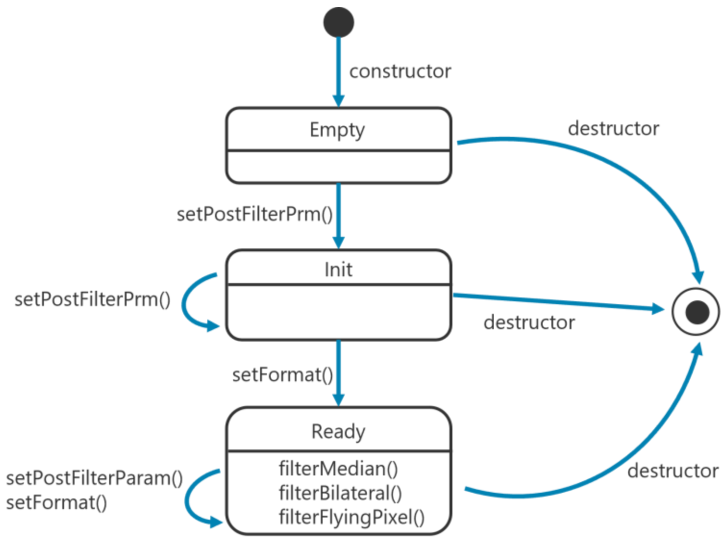

Figure 4. State machine of PostFilter class

3-3-4. Control sequence

This section describes an image conversion sequence using the PostFilter class. In the following sequence, judgment of abnormal system such as function return value judgment is omitted.

Setting of filtering processing parameters (setPostFilterPrm(), setFormat()) must be done before image reception.

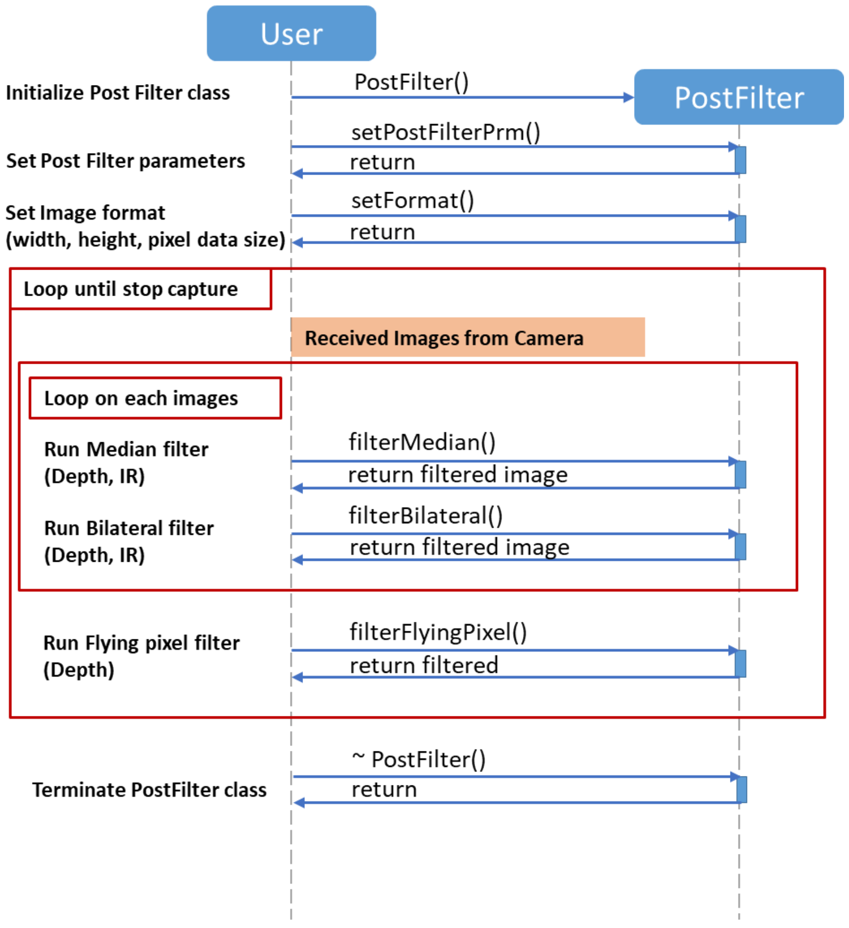
User
PostFilter
Initialize Post Filter class
PostFilter()
Set Post Filter parameters
setPostFilterPrm()
return
Set Image format (width, height, pixel data size)
setFormat()
return
Loop until stop capture
Received Images from Camera
Loop on each images
Run Median filter (Depth, IR)
filterMedian()
return filtered image
Run Bilateral filter (Depth, IR)
filterBilateral()
return filtered image
Run Flying pixel filter (Depth)
filterFlyingPixel()
return filtered
Terminate PostFilter class
~ PostFilter()
return

Figure 5. Image conversion sequence diagram

3-3-5. Method details

3-3-5-1. PostFilter

Table 28. PostFilter::PostFilter method

Function	Constructor			
Definition	PostFilter (void)			
Description	· Initializes filtering processing.			
Arguments	**Type**	**Name**	**in/out**	**Description**
	None			
Return value	None			
Sync/Async	Synchronous			

3-3-5-2. ~PostFilter

Table 29. PostFilter::~PostFilter method

Function	Destructor
Definition	~PostFilter (void)
Description	· Terminates filtering processing.

Arguments	**Type**	**Name**	**in/out**	**Description**
	None			
Return value	None			
Sync/Async	Synchronous			

3-3-5-3. setPostFilterPrm

Table 30. PostFilter::setPostFilterPrm method

Function	Setting parameters for filtering			
Definition	Result setPostFilterPrm (const PostFilterPrm& filter_prm)			
Description	・ Sets parameter information to be used for filtering.			
Arguments	**Type**	**Name**	**in/out**	**Description**
	const PostFilterPrm&	filter_prm	in	Parameters for filtering
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	An invalid argument was set.	
	ERR_BAD_STATE	6	Status transition error	
Sync/Async	Synchronous			

3-3-5-4. setFormat

Table 31. PostFilter::setFormat method

Function	Image format information setting			
Definition	Result setFormat (const ImageFormat& format)			
Description	・ This method sets the image format information handled by the PostFilter class. ・ When using filterFlyingPixel(), set the image format of the Depth image. ・ When performing various filtering functions on multiple types of images, they must have the same image format (Image Width, Image Height). ・ ERR_BAD_STATE is returned when this method is called without any filtering parameters set by setPostFilterPrm().			
Arguments	**Type**	**Name**	**in/out**	**Description**
	const ImageFormat&	format	in	Image format
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	Image format information is empty.	
	ERR_BAD_STATE	6	Status transition error	
Sync/Async	Synchronous			

3-3-5-5. filterMedian

Table 32. PostFilter::filterMedian method

Function	Median filter execution
Definition	Result filterMedian (const ImageData& org_img,

	ImageData& aft_img)			
Description	・ Applies a median filter to the input image. ・ Allocate an area in the aft_img argument that has the same number of pixels as the org_img argument. ・ ERR_BAD_STATE is returned if this method is called without having executed setPostFilterPrm() or setFormat().			
Arguments	**Type**	**Name**	**in/out**	**Description**
	const ImageData&	org_img	in	Input image
	ImageData&	aft_img	out	Filtered image
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	An image format different from setFormat () is set in the org_img and aft_img arguments.	
	ERR_BAD_STATE	6	State transition error	
Sync/Async	Synchronous			

3-3-5-6. filterBilateral

Table 33. PostFilter:: filterBilateral method

Function	Bilateral filter execution			
Definition	Result filterBilateral (const ImageData& org_img, ImageData& aft_img, bool is_depth)			
Description	・ Applies a bilateral filter to the input image. ・ Allocate an area with the same number of pixels as org_img in the aft_img argument. ・ For the is_depth argument, set true if org_img is a depth image and false if it is an IR image. ・ ERR_BAD_STATE is returned when this method is called without executing setPostFilterPrm() or setFormat().			
Arguments	**Type**	**Name**	**in/out**	**Description**
	const ImageData&	org_img	in	Input image
	ImageData&	aft_img	out	Filtered image
	bool	is_depth	in	Whether the input image is depth image true : Depth image false : IR image
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	An image format different from setFormat() is set for the org_img and aft_img arguments.	
	ERR_BAD_STATE	6	State transition error	
Sync/Async	Synchronous			

3-3-5-7. filterFlyingPixel

Table 34. PostFilter:: filterFlyingPixel method

Function	Frying pixel filter execution			
Definition	Result filterFlyingPixel (const ImageData& org_img, ImageData& aft_img)			
Description	・ Depth Applies the flying pixel filter to the image. ・ Allocate an area with the same number of pixels as org_img in the aft_img argument. ・ ERR_BAD_STATE is returned when this method is called without executing setPostFilterPrm () or setFormat().			
Arguments	**Type**	**Name**	**in/out**	**Description**
	const ImageData&	org_img	in	Input image
	ImageData&	aft_img	out	Filtered image
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	An image format different from setFormat () is set in the org_img and aft_img arguments.	
	ERR_BAD_STATE	6	State transition error	
Sync/Async	Synchronous			

3-3-6. Definition for PostFilter class

The definitions used in the PostFilter class described in PostFilterType.h are as follows.

3-3-6-1. Structure definition

3-3-6-1-1. List of structure definitions

Table 35. List of structure definitions

Name	Description
PostFilterPrm	Parameters for filtering

3-3-6-1-2. Structure definition details

3-3-6-1-2-1. PostFilterPrm

Table 36. PostFilterPrm defionition

Definition	struct PostFilterPrm {
	uint8_t median_ksize;
	uint8_t bil_ksize;
	double bil_sigma_depth;
	double bil_sigma_ir;
	double bil_sigma_space;
	uint8_t flyp_ksize;
	bool flyp_log;
	uint16_t flyp_thr;

	bool flyp_fast_proc;
	};			

Description	・ Indicates the parameters used for various types of filtering.			
Arguments	**Type**	**Name**	**Description**	**C11U Default**
	uint8_t	median_ksize	Median filter kernel size 3: 3×3 pixels, 5: 5×5 pixels	3
	uint8_t	bil_ksize	Bilateral filter kernel size 3: 3×3 pixels, 5: 5×5 pixels	3
	double	bil_sigma_depth	Bilateral Filter Depth Smoothing Parameter Range: 0.1~1000.0	500.0
	double	bil_sigma_ir	Bilateral Filter IR Smoothing Parameter Range: 0.1~250.0	100.0
	double	bil_sigma_space	Bilateral Filter Spatial Smoothing Parameter Range: 0.1~10.0	1.0
	uint8_t	flyp_ksize	Flying Pixel Filter Kernel Size 3: 3×3 pixels, 5: 5×5 pixels	3
	bool	flyp_log	Flying Pixel Filter Processing Method true: Ratio, false: Differential(test func.)	true
	uint16_t	flyp_thr	Flying Pixel Filter Threshold A smaller threshold results in greater filtering. Range: 0~8000	130
	Bool	flyp_fast_proc	Flying Pixel Filter Processing Method true: Speed priority, false: Accuracy priority	true
Reference	3-3-5-3. setPostFilterPrm			

4. Terms of Use and Disclaimer

Please refer to the "C11U User's Guide", "TOPPAN ToF SDK API Reference Manual", and other related documents for the terms and conditions of use for products from TOPPAN Holdings Inc. and TOPPAN Inc. (hereinafter referred to as "the Company").

- Unauthorized copying, reproduction, or reprinting of any part or all of this document is strictly prohibited.
- The contents of this document are subject to change without notice.
- While the Company takes every measure to provide accurate information, it does not assume responsibility for errors or omissions. Furthermore, the Company is not liable for any damages resulting from the use of the information contained in this document.
- The Company shall not be liable for any damages, including but not limited to data loss, loss of opportunity, loss of profit, and other incidental, indirect, or consequential damages, arising from the use of this product.
- Product names and company names, and other proper nouns mentioned in this document and related documents, belong to their respective companies or individuals. In this document, the TM (™) and R (®) marks may be omitted. These names are used for identification and explanation purposes within this document only. The Company has no intention to infringe on any of these rights.

5. Document history

Date	Version	Comment
2024/09/6	1.00	Initial release
2025/03/19	1.10	Initial release for C11U ES Version
2025/04/17	1.11	Modified minor typos
2025/06/24	1.12	Modified minor typos 1-5. Operating environment; update (Jetpack version included)

TOPPAN ホールディングス株式会社 TOF 事業推進センター
TOF Business Development Center, TOPPAN Holdings Inc.

TOPPAN 株式会社 エレクトロニクス事業本部
Electronics Division, TOPPAN Inc.

Location
(日本語) 〒108-8539 東京都港区芝浦 3-19-26 トッパン芝浦ビル
(English) 3-19-26, Shibaura, Minato-ku, Tokyo, 108-8539

E-mail	electronics@toppan.co.jp
Website	https://www.toppan.com/ja/electronics/device/tof/ (TOPPAN Inc.)

ToF camera product support center

E-mail	btop_support@toppan.co.jp