Scene Change Detection

Application Programming Interface

and

User's Guide (Beta Version 0.5)

For TMS320DM81xx-based Reference Design Kits

01/31/2012

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Contents

1 Introduction

Texas Instruments (TI) is equipping its video surveillance solutions with intelligent, high-performance video analytics (VA) to meet the ever-increasing requirements of top-tier security equipment manufacturers. Two VA functions have become mandatory capabilities fulfilled by state-of-the art video surveillance devices: *camera tamper detection* and *motion detection*. Camera tamper detection provides automatic notification of events that compromise the usefulness and integrity of video. Motion detection automatically identifies regions in the video that are changing due to movement in the camera's field of view or scene. TI has developed an efficient, real-time VA algorithm called *Scene Change Detection* (SCD) to enable both tamper detection and motion detection. This document will explain SCD's application program interface (API) as well as provide additional guidance for understanding and deploying the algorithm.

1.1 Scene Change Detection Overview

Determining that something or someone is moving in video or that video has been affected by tampering is generally trivial for human observers. However, it can be very difficult for unsupervised systems to exercising this kind of judgment, especially because there are many benign, unintended conditions that can easily be confused as movement or tampering. To solve these problems, a growing class of algorithms is being developed to recognize various digital video signatures that can indicate the presence of movement as well as known tamper conditions. One of these algorithms is *Scene Change Detection* or SCD.

At a high level, SCD is a body of methods that work together to perform two primary operations:

a) monitor video for differences between video frames, and
b) monitor video for differences between a learned model of the scene and incoming frames as well as the presence of blockage in front of a camera.

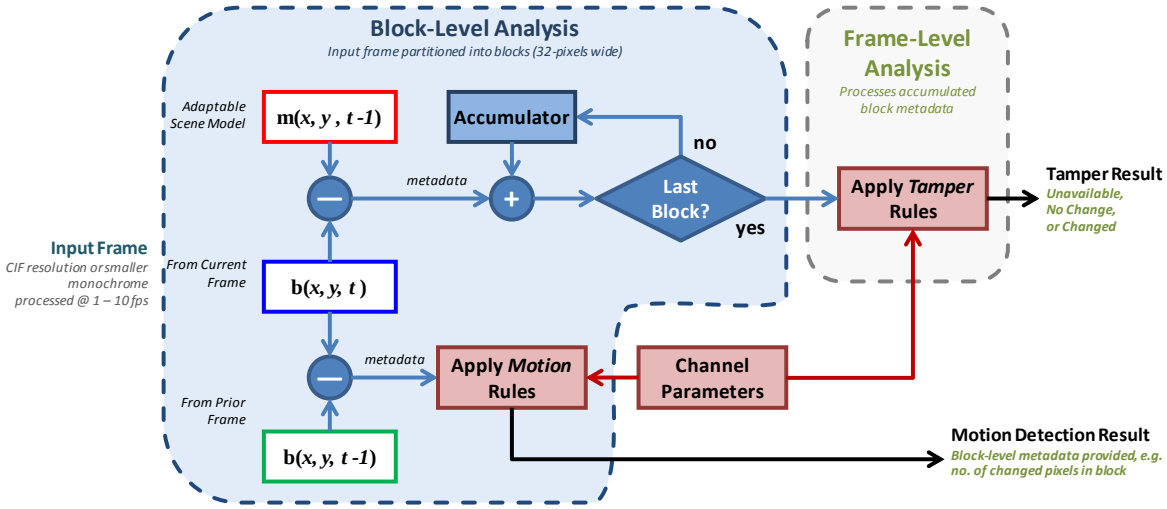
To enable motion detection, SCD works by measuring the changes between two video frames. For most surveillance applications, it's natural for a camera's field-of-view to contain areas where change due to motion can be ignored as well as areas where knowledge of activity is informative. To satisfy this use case, SCD partitions a frame into two-dimensional blocks, processing each one independently. Block-based processing provides a means to isolate changes in video to a specific region-of-interest, e.g., a 32x12 pixel block, as opposed to the entire frame.

To enable tamper detection, SCD works by measuring the global differences between the current video frame and a learned model of the scene, or field of view. SCD adapts the scene model over time to account for natural changes due to lighting variations, moving foreground objects, slight camera vibrations, etc. It assumes that events associated with tamper will affect the entire field of view; as such, SCD monitors the input for unusually large deviations from the adaptive scene model. When configured with the appropriate sensitivity setting, the algorithm is designed to provide notification when the scene experiences gross camera movement or changes due to objects that block or cover a majority of the field of view. SCD has built-in counter measures to help avoid false detections due to automatic gain control, slight camera movement, and the presence of large moving objects.

SCD provides a form of motion detection by assessing block-level changes. By aggregating block-level changes over the entire frame, SCD can also provide awareness of some tamper conditions.
The algorithm was specifically designed to leverage the computational architecture of TI's DM81xx product families, which are equipped with various programmable cores that support efficient software pipelining and SIMD (single instruction, multiple data) operations. It scales in support of multi-channel applications for digital video recorders (DVRs).

1.2 SCD Block Diagram

Figure 1 ***SCD channel processing flow.***



The block diagram above illustrates the basic flow of the algorithm. It is helpful to regard the SCD algorithm as an "engine" that consumes input video frames and produces metadata. Input video frames fed to SCD are first partitioned into blocks. Valid YUV input frames provided by the framework are generally CIF resolution or smaller, but block widths are always fixed to be 32-pixels wide. The x^{th} horizontal and y^{th} vertical block in the partition matrix at time t, i.e. $\mathbf{b}(x,y,t)$, is compared against the co-located block from a prior frame $\mathbf{b}(x,y,t-1)$ *if motion detection for that block is enabled*. If frame-level change, e.g. tamper detection, is enabled, then $\mathbf{b}(x,y,t)$ is compared against a learned model of the scene $\mathbf{m}(x,y,t-1)$. The actual operations to generate block-level metadata are more complex than a simple "subtraction," as depicted in the block diagram.

These metadata are evaluated by logical rules at both a block- and frame-level, depending on whether motion monitoring and/or tamper monitoring is in effect, respectively. Rules for interpreting the metadata are housed inside the algorithm; however, rules and how they influence decisions can be manipulated by channel-specific parameters selected by the application.

1.3 Release Notes about SCD Beta Release, Version 00.50

Having completed the initial Alpha phase testing and evaluation, version 00.50 is the first beta release of the SCD algorithm. All features provided in this release are functional except where explicitly noted. Version 00.50 was designed for the C674x on DM81xx devices, but not been fully optimized to take full advantage of the DSP's architecture. This beta release has survived a second round of internal testing for stability but may not be free of all defects. If encountered, please document the use case, all channel and SCD instance settings as well as collect sample test video so that the condition can be reproduced.

Items addressed in release 00.50 are as follows:

- *Fixed bugs in the block-based analysis engine that affected detection outcome.*
- *Added support for block-based motion detection.*
- *Added two sensitivity levels for broader control of detection responsiveness.*
- *SCD code base partially optimized to reduce C674x DSP load.*
- *Algorithm tuned for wider set of environmental conditions and use cases.*

The following insufficiencies with the SCD algorithm are present in this version:

- *Only 5 frames per second processing has been extensively tested. The impact to processing at other frame rates is currently under investigation.*
- *Detection filtering for lights ON or OFF conditions is currently a work-in-progress and may not perform as expected.*
- *Portions of the algorithm code base have not been optimized to achieve full performance entitlement on DM81xx devices.*

2 Application Programming Interface for Scene Change Detection

This section describes SCD's *enumerated types*, *data structures*, and *function prototypes* that define the application programming interface. Application developers can find reference these elements in **algLink.h**.

2.1 SCD Enumeration Declarations

The following enumerated types are defined and supported in the algorithm header file **algLink.h** in version 0.5, except where noted.

2.1.1 AlgLink_ScdMode

AlgLink_ScdMode specifies the SCD channel's operating mode. This setting allows SCD to *enable* (monitor) or *disable* change detection of a video channel. In addition to functioning as SCD's effective on/off switch for the channel, the mode setting is also used to establish the output configuration, i.e., full-frame (*tamper*), block-based (*motion detection)*, or both. Mode settings can be issued dynamically as required. For example, if video input is supplied by a PTZ camera, the application can disable SCD monitoring of the channel prior to engaging PTZ servos motors by setting the mode to `ALG_LINK_SCD_DETECTMODE_DISABLE` to prevent possible false tamper or motion alarms.

```
typedef enum
{
    ALG_LINK_SCD_DETECTMODE_DISABLE                     = 0,
    ALG_LINK_SCD_DETECTMODE_MONITOR_FULL_FRAME          = 1,
    ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS              = 2,
    ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS_AND_FRAME    = 3
} AlgLink_ScdMode;
```

2.1.2 AlgLink_ScdSensitivity

AlgLink_ScdSensitivity specifies the sensitivity setting of the scene change detection algorithm. There are *seven* levels of sensitivity to provide a coarse level of control over the way SCD behaves. This enumerated setting is used to establish detection thresholds for both frame-level differences, e.g., tamper detection, as well as block-level changes. There is a direct correlation between *the percentage of overall change that SCD must sense to detect activity* and *the sensitivity settings*. For example, setting frame-level sensitivity to `ALG_LINK_SCD_SENSITIVITY_VERYHIGH` establishes the threshold low enough to make SCD responsive to more subtle tamper threats in the scene. As a trade off for the increased responsiveness, however, undesired tamper events could be more likely. In contrast, lower sensitivity values require more substantial change before a threshold is reached, which makes SCD less responsive to potential tamper activity in the scene and more likely to result in some missed events.

```
typedef enum {
    ALG_LINK_SCD_SENSITIVITY_VERYLOW   =  0,
    ALG_LINK_SCD_SENSITIVITY_LOW       =  1,
    ALG_LINK_SCD_SENSITIVITY_MIDLO     =  2,
    ALG_LINK_SCD_SENSITIVITY_MID       =  3,
    ALG_LINK_SCD_SENSITIVITY_MIDHI     =  4,
    ALG_LINK_SCD_SENSITIVITY_HIGH      =  5,
    ALG_LINK_SCD_SENSITIVITY_VERYHIGH  =  6
} AlgLink_ScdSensitivity;
```

2.1.3 AlgLink_ScdOutput

`AlgLink_ScdOutput` indicates the frame-level result of the SCD algorithm after processing a channel's input video. If frame-level analysis of the scene indicates that the current video has changed significantly from the learned, adaptive scene model, the SCD prototype `SCD_TI_process` responds with `ALG_LINK_SCD_DETECTOR_CHANGE` for the respective channel. If no meaningful change is assessed, `ALG_LINK_SCD_DETECTOR_NO_CHANGE` is returned. If SCD is unable to assess frame-level changes, `ALG_LINK_SCD_DETECTOR_UNAVAILABLE` is returned. The latter condition may result from video input that produces metadata than can not be properly interpreted. For example, if the camera scene contains little-to-no visual texture, SCD's internal rule logic may not be able to distinguish a tamper event, e.g., a camera blocked by a sheet of white paper, from a benign condition, e.g., a camera pointed towards a blank white wall.

```
typedef enum
{
    ALG_LINK_SCD_DETECTOR_UNAVAILABLE =-1,
    ALG_LINK_SCD_DETECTOR_NO_CHANGE   = 0,
    ALG_LINK_SCD_DETECTOR_CHANGE      = 1
} AlgLink_ScdOutput;
```

2.2 SCD Structures

The following structures are defined and supported in SCD version 00.50, except where noted. The multi-channel framework's algorithm link configuration, i.e., *AlgLink*, imposes seemingly redundant structures. Structures bearing the **SCD**_ prefix are used internally by the algorithm versus those beginning with **AlgLink**_, which are used by the Link-side application.

2.2.1 AlgLink_ScdCreateParams

AlgLink_ScdCreateParams defines the set of parameters used at creation time to configure the SCD Link instance.

```
typedef struct
{
    UInt32 maxWidth;
    UInt32 maxHeight;
    UInt32 maxStride;
    UInt32 numCh;
    UInt32 numSecs2WaitB4Init;
    UInt32 numSecs2WaitB4FrmAlert;
    UInt32 inputFrameRate;
    UInt32 outputFrameRate;
    UInt32 scdChEnableFlag[ALG_LINK_SCD_MAX_CH];
    AlgLink_ScdChParams chDefaultParams[ALG_LINK_SCD_MAX_CH];
    UInt32 numBufPerCh;
    UInt32 numValidChForSCD;
} AlgLink_ScdCreateParams;
```

Variable Name	Type	Range	Description
maxWidth	U32	[160, 360]	The maximum possible width of an input frame video channel supplied to SCD. *The current release supports **maxWidth** values up to 360.*
maxHeight	U32	[192, 288]	The maximum possible width of an input frame video channel supplied to SCD. *The current release supports **maxHeight** values up to 288.*
maxStride	U32	---	The maximum possible pitch/stride of an input frame video channel supplied to SCD.
numCh	U32	[1, 16]	*This release supports up to 16 SCD channels.*
numSecs2WaitB4Init	U32	---	*The number of seconds to wait before initializing SCD monitoring after create-time. This wait time occurs when the algorithm instance first starts and allows video input to stabilize.*
numSecs2WaitB4FrmAlert	U32	---	*The number of seconds to wait before signaling a frame-level scene change event. The delay can be used to filter spurious events.*

Variable Name	Type	Range	Description
inputFrameRate	U32	[1, 30]	*Frames per second fed to SCD. Typical rate is 30 fps.*
outputFrameRate	U32	[1,2,3,5,6,10]	*Rate at which channel frames are processed by SCD, in frames per second. This release supports a limited number of integer rates. Rate applies to all channels. Only the fps rates supported here are valid;*
scdChEnableFlag	U32	[1, ALG_LINK_SCD_MAX_CH]	*Boolean flag array SCD references when installing video channels.*
chDefaultParams	AlgLink_ScdChParams		*Array of SCD Channel parameters use to configure SCD. Up to ALG_LINK_SCD_MAX_CH channels can be defined in this release.*
numBufPerCh	U32	[1, 16]	*Number of metadata buffers to allocate per channel.*
numValidChForSCD	U32	[1, 16]	*The maximum channels on which scd will run.* *startChNoForSCD + numValidChForSCD should be less than MAX Input channels* *Still all parameter settings for SCD will be for Ch Num 0 ~ numValidChForSCD.* *startChNoForSCD, numValidChForSCD are used only to separate valid channels from an input queue having other channels which don't require SCD*

2.2.2 AlgLink_ScdChParams

AlgLink_ScdChParams defines individual channel parameters used to configure how SCD processes the input video. This SCD Link parameter set is complements the SCD Algorithm parameter set **SCD_chPrm**, which is used internally.

```
typedef struct AlgLink_ScdChParams
{
    UInt32 chId;
    UInt32 mode;
    UInt32 frmIgnoreLightsON;
    UInt32 frmIgnoreLightsOFF;
    AlgLink_ScdSensitivity frmSensitivity;
    UInt32 frmEdgeThreshold;
    UInt32 blkNumBlksInFrame;
    AlgLink_ScdblkChngConfig blkConfig[ALG_LINK_SCD_MAX_BLOCKS_IN_FRAME];
} AlgLink_ScdChParams;
```

Variable Name	Type	Range	Description
chId	S32	---	**Channel ID:** *A 32-bit number used to uniquely identify the SCD channel. Serves as a internal handle.*
ignoreLightsON	UInt32	[0, 1]	**Detection filter:** *To suppress events when environmental illumination levels suddenly get dramatically brighter, e.g. turning lights on in a dark room, set to 1; Otherwise, set to zero to detect this condition.*
ignoreLightsOFF	UInt32	[0, 1]	**Detection filter:** *To suppress events when environmental illumination levels suddenly get dramatically darker, e.g. turning lights off in a well-lit room, set to 1; Otherwise, set to zero to detect this condition.*
mode	AlgLink_ScdMode	[0, 1, 2, 3]	**Channel configuration mode:** • *Set to 0 to disable all SCD processing on channel* • *Set to 1 to enable change detection on entire frame (only)* • *Set to 2 to enable change detection on frame tiles/blocks (only)* • *Set to 3 to enable change detection on entire frame as well as individual frame tiles/blocks*
frmSensitivity	AlgLink_ ScdSensitivity	[0,7]	**Detection sensitivity:** *internal threshold for scd change tolerance; globally applied to entire frame*
frmEdgeThreshold	S16		*The minimum number of edge pixels required to indicate non-tamper event. No implemented in this version.*
blkNumBlksInFrame	S16	[1, 264]	*The number block/tiles available in a frame after it has been partitioned for block-based processing. Block width is always 32-pixels wide in this version. Block height is typically 10 or 12.*
blkConfig	AlgLink_ ScdblkChngConfig	--	*Array of* AlgLink_ScdblkChngConfig *elements used to configure each block, e.g. enable/disable as well as set sensitivity, when motion detection capabilities are desired. Up to* ALG_LINK_SCD_MAX_BLOCKS_IN_FRAME *blocks can be defined.*

2.2.3 SCD_blkChngConfig

SCD_blkChngConfig is a structure used to configure and individual block for motion detection.

```
typedef struct
{
      SCD_Sensitivity sensitivity;
      U32 monitored;
} SCD_blkChngConfig;
```

Variable Name	Type	Range	Description
sensitivity	SCD_Sensitivity	[0, 6]	*Blocks's sensitivity setting for change detection*
monitored	UInt32	[0, 1]	*Flag indicates whether to monitor block for change detection.*

2.2.4 SCD_chPrm

SCD_chPrm defines individual channel parameters used to configure how SCD processes the input video. This SCD Link parameter set is complements the SCD Algorithm parameter set **SCD_chPrm**, which is used internally. All parameters must be set with valid values prior to calling **SCD_TI_setPrms**. Parameter sets can not change between calls to **SCD_TI_setPrms**, but can not change after to support dynamic conditions.

```
typedef struct AlgLink_ScdChParams
{
    UInt32 chId;
    UInt32 mode;
    UInt32 width;
    UInt32 height;
    UInt32 stride;
    void   *curFrame;
    SCD_Sensitivity frmSensitivity;
    UInt32 frmIgnoreLightsON;
    UInt32 frmIgnoreLightsOFF;
    UInt32 frmEdgeThreshold;
    SCD_blkChngConfig *blkConfig;
} SCD_chPrm;
```

Variable Name	Type	Range	Description
chId	S32	---	**Channel ID:** *A 32-bit number used to uniquely identify the SCD channel. Serves as a internal handle.*
ignoreLightsON	UInt32	[0, 1]	**Detection filter:** *To suppress events when environmental illumination levels suddenly get dramatically brighter, e.g. turning lights on in a dark room, set to 1; Otherwise, set to zero to detect this condition.*

Variable Name	Type	Range	Description
ignoreLightsOFF	UInt32	[0, 1]	**Detection filter:** *To suppress events when environmental illumination levels suddenly get dramatically darker, e.g. turning lights off in a well-lit room, set to 1; Otherwise, set to zero to detect this condition.*
mode	SCD_Mode	[0, 1, 2, 3]	**Channel configuration mode:** • *Set to 0 to disable all SCD processing on channel* • *Set to 1 to enable change detection on entire frame (only)* • *Set to 2 to enable change detection on frame tiles/blocks (only)* • *Set to 3 to enable change detection on entire frame as well as individual frame tiles/blocks*
frmSensitivity	SCD_Sensitivity	[0,7]	**Detection sensitivity:** *internal threshold for scd change tolerance; globally applied to entire frame*
frmEdgeThreshold	S16		*The minimum number of edge pixels required to indicate non-tamper event. No implemented in this version.*
blkNumBlksInFrame	S16	[1, 264]	*The number block/tiles available in a frame after it has been partitioned for block-based processing. Block width is always 32-pixels wide in this version. Block height is typically 10 or 12.*
stride	S16	[160, 360]	**Input frame stride:** *Should match the input frame width in this version*
blkConfig	AlgLink_ ScdblkChngConfig	--	*Array of block-wise change detection parameters for each channel. Set this pointer to a valid, allocated array to set the block during set-up; Otherwise, set to NULL to preserve the existing configuration arrangement, e.g., you only need to set this as required to change the sensitivity or monitor state. Up to* ALG_LINK_SCD_MAX_BLOCKS_IN_FRAME *blocks can be defined.*

2.3 SCD Function Calls

The prototype declarations defined in the section are supported in SCD version 00.50. SCD algorithm instance is designed to be called by a Link application. Please reference the file **scdLink_alg.c** for an example that shows how to create the SCD instance.

2.3.1 SCD_TI_setPrms

SCD_TI_setPrms passes the channel parameter sets of all channels that should be processed by the SCD instance. The function checks for erroneous values in each parameter set that is submitted, updating those with legitimate, or at least, non-lethal values *(more on this below)*. In a real-time, multi-channel environment, SCD processing rates are generally lower than captured frame rates, so the number of channels to process is typically lower than the maximum number of channels configured, e.g. **numCh < maxChannels**. Because the pointer for the current input frame (a field in the channel parameter set) has to be furnished, *this function must be called prior to every call to **SCD_TI_process***.

The function returns a status value representing the collective status result from the parameter sets submitted. If SCD_NO_ERROR is returned, all sets have valid parameters and the application can proceed with a call to **SCD_TI_process**. Otherwise, the application should query the **status** array to identify the channel(s), i.e., **scdPrm**, with an invalid parameter set. If non-lethal values are submitted for updating channel settings, e.g. incorrect mode or sensitivity, existing valid values are preserved and the application can proceed with a call to **SCD_TI_process**. However, if lethal errors are encountered, e.g. null pointer supplied for current frame pointer, SCD will invalidate the channel, so subsequent attempts to process the channel could fail.

```
SCD_Status
SCD_TI_setPrms(PTR           handle,
               SCD_chPrm    *pScdParam,
               U32           chanID,
               SCD_Status    status);
```

Variable Name	Type	Description
handle	PTR	Handle pointer to the SCD object instance
*pScdChParam	SCD_chPrm	Pointer to SCD channel parameter.
chanID	U32	Unique channel identifier.
*status	SCD_Status	Status values populated after program returns from the function call.

2.3.2 SCD_TI_process

SCD_TI_process peforms scene change detection on video channels identified in the array `pChId`. The detection results for the respective channels is populated in the array **`pScdResult`**. The error status is also returned.

```
SCD_Status
SCD_TI_process(PTR        handle,
               UInt32     chId,
               SCD_Result *pScdResult);
```

Variable Name	Type	Description
handle	PTR	Handle pointer to the SCD_TI_Obj object instance
chanID	U32	Unique channel identifier.
*pScdResult	SCD_Result	Array of status values populated after program pointer returns from the function call. Array values reflect status of respective channel parameter sets submitted.

3 User's Guide for Scene Change Detection

Like all video analytics algorithms, the behavior of SCD is highly dependent on a number of factors, but none more important than the composition and activity in the scene itself. This section describes how to deploy the camera tamper and motion detection capabilities provided by version 00.50 of SCD. General guidelines for setting up cameras that furnish video to SCD are provided as well as descriptions of expected algorithm behavior under various environmental conditions.

3.1 Scene Change Detection Goal

Examples of camera tampering provided in this version include obstructing the entire lens with paint or a foreign object, major adjustments to the lens' focus or aperture settings, and turning off the lights (if indoors with no major change in ambient illumination expected). Depending on the sensitivity setting, SCD will detect these types of deliberate events while also tolerating the effects of automatic camera gain, camera shaking, and dimly lit scenes. SCD is not designed to catch slight lens defocusing or mild adjustments in camera pose or rotation. Attempts to point the camera in a completely different direction may go undetected, especially at lower sensitivity levels.

3.2 Specifications

Specification	Value
Max. Input Frame Width	**360**
Max. Input Frame Height	**288**
Max. Number of Video Channels	**16**

3.3 General Guidelines for Video and Scene Characteristics

To perform optimally, video analytics requires careful consideration of several factors, including each camera's placement, field of view, and illumination under various operating conditions. SCD is designed to analyze video that is acquired from a fixed camera, i.e. the field of view does not change due to panning, tilting, or zooming. Video is processed frame-by-frame. The order and timing of frames are crucial for analysis. SCD algorithms expect frames to be available in sequential order with inter-frame jitter (variability in frame timing) minimized to be no greater than ±100 ms outside of the specified processing rate.

SCD relies on a fairly stable field of view to identify relevant changes caused by moving objects. Tamper events are assumed to affect the majority of the field of view, depending on the sensitivity setting. To prevent false alarms and achieve desired results, installers should be careful to position the camera to satisfy the following constraints:

- The video should be in focus and as sharp as possible.
- Camera mounting should be fixed and stable. Excessive vibration or movement from wind, large vehicles, or other external factors should be avoided.

- Good contrast with strong edges and corners is desirable for optimum performance. Large reflective surfaces, glare and direct illumination (camera pointed at the sun) can result in poor contrast and must be avoided. Tamper detection in scenes without enough visual texture or contrast, e.g. camera pointed at a blank wall, could be ineffective. However, motion detection has no similar requirement, except for sufficient illumination.

- No more than 75% of the scene should experience motion or change in appearance at any given time. The size of any individual moving object should not fill more than 50% of the camera's field of view at any time. If these recommendations are unavoidable and the field of view is easily filled by objects moving into the scene temporarily, e.g. the scene of a camera monitoring traffic is filled by vehicles in traffic, consider lowering the sensitivity to prevent false detections.

- Areas monitored by the camera should be reasonably well lit, e.g. adequate for supporting human eyesight. SCD can work with infrared illuminators to assist in very low-light environments or conditions, but operation under these circumstances can produce undesired effects. Precipitation, e.g. rain drops, snow flakes, ice, sleet, etc., dirt, insects, or other debris on the camera lens can cause SCD algorithms to work improperly.

Adjustments to sensitivity settings will influence detection performance. Lower sensitivity will require a larger degree of high-contrast change for a tamper event to be generated and will result in fewer false events. Higher sensitivity will require a smaller amount of change for an event to be generated and could therefore result in more events, some of them false alarms.