Application Report
– May 2012

SCD Algorithms Integration in DVR RDK

Video Surveillance Applications

ABSTRACT

The DVR RDK is a multi-processor software development framework for TI81xx platform and is optimized for multi-channel applications like Surveillance DVR, NVR, Hybrid-DVR, HD-DVR. With the increasing interest in video analytics from the customers of the video surveillance end equipment, Scene change detection module has been provided with DVR solution. The scene detection has capabilities to detect tamper in the camera and motion in the camera region as selected by user. User can choose any of these features as per their requirement. This document talks about how to integrate SCD algorithm into DVR application and also covers various APIs supported to handle/control operation of SCD algorithm.

1 Overview

Scene Change Detection algorithm runs on DSP in TI814x and TI816x along with Software On Screen Display (SWOSD). SCD has following two capabilities. These algorithm runs on iMX (i.e. SIMCOP) in TI810x platform.

a. ***Tamper detection***: Event notification when camera is blocked, covered or disconnected.
b. ***Motion detection*** : Notification of any movement in certain region of frame. Frame is represented by fixed size smaller blocks and Motion detection can be enabled in one or more blocks..

For further details please refer SCD API guide document "***SCD_TI_API_UsersGuide_v00.50.pdf*** " present in the ***<DVR_RDK>/docs/*** folder.

In TI810x platform, only Tamper detection is supported. Motion detection will not be supported on TI810x platform.

2 Scene Change Detection (SCD) setup

In the dvr_rdk system, DSP/SIMCOP is available to schedule any algorithmic task. Currently, SWOSD and SCD algorithm run on DSP/SIMCOP. Apart from these two tasks, user can schedule/map their algorithm on DSP/SIMCOP. In the current implementation, these two algorithms are part of ALGLink.

ALGLink is used to apply SWOSD and SCD algorithms on channel data. User can independently enable or disable them while configuring ALGLink. The create time parameter ***AlgLink_CreateParams*** of ALGLink, defined in ***algLink.h,*** has separate create time parameter for SWOSD and SCD. User can add new task, which can be scheduled on DSP/SIMCOP, into ALGLink and can be handled as it is done for SWOSD and SCD.

```
typedef struct
{
   Int32   enableOSDAlg;
   Int32   enableSCDAlg;
   System_LinkInQueParams  inQueParams;
   System_LinkOutQueParams outQueParams;
   AlgLink_OsdChCreateParams  osdChCreateParams[ALG_LINK_OSD_MAX_CH];
   AlgLink_ScdCreateParams    scdCreateParams;
} AlgLink_CreateParams;
```

SCD can be activated by setting ***enableSCDAlg*** flag to 1. If the SCD is enabled, user can set/initialize SCD related parameters that are part of ***AlgLink_ScdCreateParams*** structure.

```
typedef struct
{
   UInt32 maxWidth;
   UInt32 maxHeight;
   UInt32 maxStride;
   UInt32 numSecs2WaitB4Init;
   UInt32 numSecs2WaitB4FrmAlert;
   UInt32 numSecs2WaitAfterFrmAlert;
   UInt32 inputFrameRate;
   UInt32 outputFrameRate;
   AlgLink_ScdChParams chDefaultParams[ALG_LINK_SCD_MAX_CH];
   UInt32 numBufPerCh;
   Bool enableMotionNotify;
   Bool enableTamperNotify;
   UInt32 numValidChForSCD;
} AlgLink_ScdCreateParams;
```

Maximum 16 channels can be processed by SCD algorithm and the configuration of same can be done via ***chDefaultParams*** array of structure ***AlgLink_ScdChParams*** that is part of ***AlgLink_ScdCreateParams.***

In the dvr_rdk release version ***02.00.00.23,*** motion detection has to be enabled with tamper detection. SCD motion detection only mode i.e. **ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS**, is not supported. Motion detection works fine when it is enabled with tamper detection i.e. SCD mode **ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS_AND_FRAME.**

A sample code snippet is provided to help user in setting up their configuration.

```
AlgLink_CreateParams  dspAlgPrm;

dspAlgPrm.enableOSDAlg = FALSE;
dspAlgPrm.enableSCDAlg = TRUE;
dspAlgPrm.outQueParams.nextLink     = ipcBitsOutDSPId;

dspAlgPrm.scdCreateParams.maxWidth                = 352;
dspAlgPrm.scdCreateParams.maxHeight               = 288;
dspAlgPrm.scdCreateParams.maxStride               = 352;

dspAlgPrm.scdCreateParams.numValidChForSCD          = 16;
dspAlgPrm.scdCreateParams.numSecs2WaitB4Init        = 3;
dspAlgPrm.scdCreateParams.numSecs2WaitB4FrmAlert  = 1;
dspAlgPrm.scdCreateParams.inputFrameRate            = 30;
dspAlgPrm.scdCreateParams.outputFrameRate           = 5;

 dspAlgPrm.scdCreateParams.enableMotionNotify    = FALSE;
 dspAlgPrm.scdCreateParams.enableTamperNotify    = TRUE;

numHorzBlks    = dspAlgPrm.scdCreateParams.maxWidth / 32;

if((dspAlgPrm.scdCreateParams.maxHeight%10) == 0)
   numVertBlks    = dspAlgPrm.scdCreateParams.maxHeight / 10;
else
   numVertBlks    = dspAlgPrm.scdCreateParams.maxHeight / 12;

numBlksInFrame = numHorzBlks * numVertBlks;

for(chIdx = 0; chIdx < dspAlgPrm.scdCreateParams.numValidChForSCD; chIdx++)
{
  AlgLink_ScdChParams * chPrm = &dspAlgPrm.scdCreateParams.chDefaultParams[chIdx];

  chPrm->blkNumBlksInFrame      = numBlksInFrame;
  chPrm->chId                   = SCDChannelMonitor[chIdx];
  chPrm->mode              =  ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS_AND_FRAME;
   chPrm->frmIgnoreLightsON     = FALSE;
   chPrm->frmIgnoreLightsOFF    = FALSE;
   chPrm->frmSensitivity         = ALG_LINK_SCD_SENSITIVITY_HIGH;
   chPrm->frmEdgeThreshold       = 100;

   /* Setting block configurations */
   i = 0;
   for(y = 0; y < numVertBlks; y++)
   {
      for(x = 0; x < numHorzBlks; x++)
      {
          chPrm->blkConfig[i].sensitivity    = ALG_LINK_SCD_SENSITIVITY_LOW;
          chPrm->blkConfig[i].monitored     = 0;
          i++;
      }
   }
 }
```

3 Application programming Interface (API) for SCD

Once the SCD operation has started; user is allowed to change some of the SCD channel configuration. APIs have been provided to change these parameters dynamically. Following APIs have been provided for this purpose.

Vcap_setDynamicParamChn(VCAP_CHN vcChnId,
VCAP_CHN_DYNAMIC_PARAM_S *psCapChnDynaParam,
VCAP_PARAMS_E paramId)

Where:
vcChnId : Capture channel ID
psCapChnDynaParam : Capture channel Specific Dynamic Parameters
paramId : Capture channel get/set param Id.

User is required to provide SCD channel configuration parameters via (***AlgLink_ScdChParams***) **scdChPrm** or block configuration update via (***AlgLink_ScdChblkUpdate)*** **scdChBlkPrm** which are part of VCAP*_CHN_DYNAMIC_PARAM_S* structure.

```
typedef struct
{
   Int32  contrast;
   Int32  satauration;
   Int32  brightness;
   Int32  hue;
   AlgLink_OsdChWinParams *osdChWinPrm;
   AlgLink_OsdChBlindWinParams *osdChBlindWinPrm;
   AlgLink_ScdChParams scdChPrm;
   AlgLink_ScdChblkUpdate scdChBlkPrm;
   AlgLink_ScdAllChFrameStatus scdAllChFrameStatus;
   CaptureLink_BlindInfo captureBlindInfo;
   VCAP_CHN_DYNAMIC_RESOLUTION chDynamicRes;
} VCAP_CHN_DYNAMIC_PARAM_S;
```

Following different capture param set is used to update SCD parameters at run time.

```
typedef enum
{
   *****          /* Other Capture Param */

   VCAP_SCDMODE,                    /* Set/update SCD mode             */
   VCAP_SCDSENSITIVITY,             /* Set/update SCD frame sensitivity */
   VCAP_SCDGETALLCHFRAMESTATUS/* Get current tamper status of all the channels.*/
   VCAP_IGNORELIGHTSOFF,
   VCAP_IGNORELIGHTSON,
   VCAP_SCDBLOCKCONFIG,          /* Set/update block config */

   ******          /* Other Capture Param */
} VCAP_PARAMS_E;
```

Below code snippets describe how user can update SCD channel parameters.

Step 1. Populate (AlgLink_ScdChParams scdChPrm) structure .

```
VCAP_CHN_DYNAMIC_PARAM_S params;
 params.scdChPrm.chId = 0;   /* Updating channel 0 */
 params.scdChPrm.mode = (UInt32)  ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS_AND_FRAME;
```

Step 2. Call VCap API

```
Vcap_setDynamicParamChn(chId, &params, VCAP_SCDMODE);
```

Step 3: VCap API in turn send command to ALGLink to update appropriate parameter.

```
status = System_linkControl(
        scdAlgLinkId,                          /* ALGLink ID assigned at create time */
        ALG_LINK_SCD_CMD_SET_CHANNEL_MODE,     /* Link Command to update SCD mode */
        &( params ->scdChPrm),                 /* pointer to the structure */
        sizeof(params->scdChPrm),              /* size to the structure */
        TRUE
        );
```

Follow below steps to update block configurations

Step 1. Populate (AlgLink_ScdChParams scdChPrm) structure .

```
VCAP_CHN_DYNAMIC_PARAM_S params;
 UInt32 startX, startY, endX, endY;

        startX = 0;
        startY = 0;
        endX = startX + 4;
        endY = startY + 8;
        for(i = startY; i < endY; i++)
        {
           for(j = startX; j < endX; j++)
           {
              AlgLink_ScdChBlkConfig * blkConfig;
              blkConfig = &params.scdChBlkPrm.blkConfig[params.scdChBlkPrm.numValidBlock];

              blkConfig->blockId     =  j + (i * 11)   ;
              blkConfig->sensitivity  = sensitivity;
              blkConfig->monitorBlock = flag;
              params.scdChBlkPrm.numValidBlock++;
           }
        }
```

Step 2. Call VCap API

```
Vcap_setDynamicParamChn(chId, &params, VCAP_SCDBLOCKCONFIG);
```

Step 3: VCap API in turn send command to ALGLink to update appropriate parameter.

```
status = System_linkControl(
        scdAlgLinkId,                                  /* ALGLink ID assigned at create time */
        ALG_LINK_SCD_CMD_SET_CHANNEL_BLOCKCONFIG,      /* Link Command to update SCD mode */
        &( params ->scdChBlkPrm),                      /* pointer to the structure */
        sizeof(params->scdChBlkPrm),                   /* size to the structure */
        TRUE
        );
```

4 Use Case Scenarios

SCD has been added in following use cases of 816x and 814x. The details of current default settings are given in below table.

SCD parameter	816x Use case		814x Use case		
	Progressive DVR App	SD Encode Only App	4D1 DVR	8D1 DVR	16 CIF DVR
Resolution	CIF	CIF	CIF	QCIF	QCIF
numValidChForSCD	16	16	4	8	16
Operation FPS	2	2	5	5	5
SCD mode	ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS_AND_FRAME				
numSecs2WaitB4Init	3				
numSecs2WaitB4FrmAlert	1				
frmIgnoreLightsON	FALSE				
frmIgnoreLightsOFF	FALSE				
frmSensitivity	ALG_LINK_SCD_SENSITIVITY_MID				
frmEdgeThreshold	100				
Block sensitivity *	ALG_LINK_SCD_SENSITIVITY_MID				
Block monitor flag *	0				

* Default settings of all the blocks within a frame.

4 Getting current Tamper status

From DVR_RDK release 2.80.xx.xx onwards, a provision is given to get current Tamper status of all the channels at once. User can use following API for the same. Please follow below instruction to use this API.

Vcap_getDynamicParamChn (VCAP_CHN vcChnId,
VCAP_CHN_DYNAMIC_PARAM_S *psCapChnDynaParam,
VCAP_PARAMS_E paramId)

Where:
vcChnId : Capture channel ID. Don't care. Can be set at 0.
psCapChnDynaParam : Capture channel Specific Dynamic Parameters
paramId : Capture channel get/set param Id.

User is required to call above API with (*paramId = VCAP_SCDGETALLCHFRAMESTATUS)* and once the call is completed user would get current Tamper status of all the channels in the (***AlgLink_ScdAllChFrameStatus)*** **scdAllChFrameStatus** which is part of VCAP_*CHN_DYNAMIC_PARAM_S* structure.

```
typedef struct
{
   UInt32 numCh;
   /**< Channel number, 0..ALG_LINK_SCD_MAX_CH-1 */

   AlgLink_ScdChStatus chanFrameResult[ALG_LINK_SCD_MAX_CH];
   /**< SCD frame level state. For valid values see, AlgLink_ScdOutput */
} AlgLink_ScdAllChFrameStatus;
```

Below is the sample code snippet to get current tamper status.

Step 1. Call VCap API

```
VCAP_CHN_DYNAMIC_PARAM_S    prm;
Vcap_getDynamicParamChn(0, &param, VCAP_SCDGETALLCHFRAMESTATUS);
```

Step 2: VCap API in turn send command to ALGLink to update appropriate parameter.

```
status = System_linkControl(
         scdAlgLinkId,                                  /* ALGLink ID assigned at create time */
        ALG_LINK_SCD_CMD_GET_ALL_CHANNEL_FRAME_STATUS
                                                        /* Link Command to get Tamper state */
        &( params ->scdAllChFrameStatus),               /* pointer to the structure */
        sizeof(params->scdAllChFrameStatus),            /* size to the structure */
        TRUE
        );
```

Step 3: After above call, user will get updated channel status in the ***scdAllChFrameStatus*** structure and the values can be extracted at below.

```
for(chId=0; chId<param.scdAllChFrameStatus.numCh; chId++)
{
   UInt32 chanNum;              /* SCD channel No. */
   UInt scdFrameStatus;         /* Current Tamper status */

   scdFrameStatus = prm.scdAllChFrameStatus.chanFrameResult[chId].frmResult;
   chanNum        = prm.scdAllChFrameStatus.chanFrameResult[chId].chId;

   if(scdFrameStatus==ALG_LINK_SCD_DETECTOR_CHANGE)
   {
      printf(" [TAMPER DETECTED]\n");
   }
   if(scdFrameStatus==ALG_LINK_SCD_DETECTOR_NO_CHANGE)
   {
      printf(" [NO TAMPER]\n");
   }
}
```

5 Extracting Block metaData

When motion detection is enabled i.e. SCD mode is either ***ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS*** or ***ALG_LINK_SCD_DETECTMODE_MONITOR_BLOCKS_AND_FRAME,*** SCD algorithm outputs block metadata that can be used in the application to take appropriate action. Each output buffer is populated by SCD algorithm in the following structure format below. From the exported buffer, user can extract different frame level information as well as block metadata as per the **AlgLink_ScdResult** structure.

In the exported buffer, meta data of each block in the frame is provided. In the block metadata structure, algorithm updates two parameters ***numFrmsBlkChanged*** and ***numPixelsChanged***. User can incorporate their logic of motion detection by using this block metadata.

```
typedef struct
{
   UInt32 chId;
   UInt32 frmResult;
   AlgLink_ScdblkChngConfig blkConfig[ALG_LINK_SCD_MAX_BLOCKS_IN_FRAME];
   AlgLink_ScdblkChngMeta  blkResult[ALG_LINK_SCD_MAX_BLOCKS_IN_FRAME];
} AlgLink_ScdResult;


typedef struct
{
   UInt32 sensitivity;
   UInt32 monitored;
} AlgLink_ScdblkChngConfig;

typedef struct
{
   UInt32 numFrmsBlkChanged;
   UInt32 numPixelsChanged;
} AlgLink_ScdblkChngMeta;
```

6 Memory requirement in algorithms

This section covers memory required by SCD algorithm. Apart from the memory requested by SCD algorithm for its internal operations, separate memory is required to pass on the SCD algorithm result to the next link. As mentioned earlier when motion detection is enabled ALGLink outputs SCD algorithm result that is generated for every frame. This data is given out in an output buffer for each processed frame. Below table list down memory requirement for 16 channels SCD algorithm, where each channel requests 6 buffers (per channel) that would be used to store block metadata of 6 frames.

Alg SCD Link	
Num Channels	16
Width	352
Height	240
Num Frames per Ch	6
Blk MeData PerFrame	4352
Total Bytes	417792
In KB	417

Once result for complete frame is available, buffers are passed on to the next link. After consuming, the buffers are released back to the ALGLink.

7 Motion tracking on display

In Ne-Progressive demo, motion tracking feature has been enabled. It can be seen on On-Chip HDMI display when it is set at 1x1 layout mode. If the block is enabled for motion tracking, the boundaries of blocks are highlighted whenever there is motion detected. User can refer the demo code in "*\dvr_rdk\demos\mcfw_api_demos\mcfw_demo\demo_scd_bits_wr.c"* file and add their own logic of highlighting blocks or add their own mechanism to use block metaData that is exported to A8.

8 References

i) SCD API user guide ***SCD_TI_API_UsersGuide_v00.50.pdf***.
ii) Demo guide "**DM81xx_DVR_RDK_DemoGuide.pdf**".