芯仑科技
CeleX™ 相机套件
SDK 使用手册

芯仑科技（上海）有限公司

目录

修改记录

Version	Date	Section	Description	Author
1.0	2017.11.07	All	New	Guoping He
1.1	2017.12.11	All	修改文档格式	Xiaoqin Hu
1.2	2017.12.15	1.2	新增 CeleX™ Sensor 的工作模式 章节	Xiaoqin Hu
1.2	2017.12.15	1.3	新增 如何使用 CeleX™ Sensor 章节	Xiaoqin Hu
1.2	2017.12.15	1.4	新增 生成 PFN 文件的方法 章节	Xiaoqin Hu
1.2	2017.12.15	2.2.6	删除 SetCallBack 接口 新增 getPixelData 接口	
1.2	2017.12.15	2.2.8	修改 getEventPicBuffer 接口参数	Xiaoqin Hu
1.2	2017.12.15	2.2.9	新增 generateFPN 接口	Xiaoqin Hu
1.2	2017.12.15	2.2.10	新增 setFpnFile 接口	Xiaoqin Hu
1.2	2017.12.15	2.2.11	新增 setSensorMode 接口	Xiaoqin Hu
1.2	2017.12.18	2.2.12	新增 getSensorMode 接口	Xiaoqin Hu
1.2	2017.12.18	2.2.12	新增 isSensorReady 接口	Xiaoqin Hu
1.2	2017.12.18	3	新增 使用示例 章节	Xiaoqin Hu
1.3	2018.01.19	2.2.14 2.2.15	新增 setTimeSlice & getTimeSlice 接口	Xiaoqin Hu
1.3	2018.01.19	2.2.16 2.2.17	新增 setThreshold & getThreshol 接口	Xiaoqin Hu
1.3	2018.01.19	2.2.18 2.2.19	新增 setContrast & getContrast 接口	Xiaoqin Hu
1.3	2018.01.19	2.2.20 2.2.21	新增 setBrightness & getBrightness 接口	Xiaoqin Hu
1.3	2018.01.19	2.2.22	新增 openSensor 接口	Xiaoqin Hu
1.3	2018.01.19	2.2.23 2.2.24	新增 setDisplayMode & getDisplayMode 接口	Xiaoqin Hu
1.3	2018.01.19	2.2.25	增加获取 Optical Flow Buffer 接口	Xiaoqin Hu
1.3	2018.02.09	All	整个文档内容和结构修改	Xiaoqin Hu
1.3.1	2018.03.13	2.2.7	修改 getEventPicBuffer 的使用说明	Xiaoqin Hu
1.4	2018.03.20	1.2.1	新增 FullPicture_Event 模式的描述	Xiaoqin Hu
1.4	2018.03.20	2.2.29	新增 setMotionTime 接口	Xiaoqin Hu
1.4	2018.03.20	2.2.30	新增 getMotionTime 接口	Xiaoqin Hu
2.1	2018.05.22	1.2.2	修改 T 的位数（19 bits →17 bits）	Xiaoqin Hu
2.1	2018.05.22	1.2.4	新增 Optical Flow 章节	Xiaoqin Hu
2.1	2018.05.22	2.2.40~2.2.45	新增 Optical Flow 相关接口	Xiaoqin Hu
2.1	2018.05.22	2.2.37~2.2.38	新增 设置 Sensor 时钟频率的相关接口	Xiaoqin Hu
2.1	2018.05.26	2.2.31~2.2.36	新增各种数据模式下设置和获取 Frame Time 的接口	Xiaoqin Hu
2.1	2018.05.26	2.2.29 2.2.30	删除了 setTimeSlice 接口 删除了 getTimeSlice 接口	Xiaoqin Hu
2.1	2018.05.30	All	内容描述以及图示修改	Xiaoqin Hu
2.1 Update	2018.06.12	1.2.1	修改 CeleX™ Sensor 的工作原理描述	Xiaoqin Hu

2.2	2018.08.02	2.2.12 2.2.14	新增 getFullPicMat & getEventPicMat 接口	Hua Ren
2.2	2018.08.02	2.2.15	新增 getEventDataVector 接口	Hua Ren
2.2	2018.08.02	2.2.40	新增 setEventFrameParameters 接口	Hua Ren
2.2	2018.08.02	2.2.42 2.2.43	新增 setTimeScale & setEventCountStepSize 接口	Hua Ren
2.2	2018.08.02	2.2.49 2.2.51 2.2.53	新增 getOpticalFlowPicMat & getOpticalFlowDirectionPicMat & getOpticalFlowSpeedPicMat 接口	Hua Ren
2.2	2018.08.02	2.2.58	新增 getAttributes 接口	Hua Ren
2.2	2018.08.02	2.2.59	新增 convertBinToAVI 接口	Hua Ren
2.2	2018.08.03	2.2.60	新增 startRecordingVideo 接口	Xiaoqin Hu
2.2	2018.08.03	2.2.61	新增 stopRecordingVideo 接口	Xiaoqin Hu
2.2	2018.08.03	2.2.62	新增 enableAutoAdjustBrightness 接口	Xiaoqin Hu

1 概述

1.1 CeleX™ Sensor 芯片相机套件的基本介绍

1.1.1 CeleX™ 相机套件工作原理

下图 1-1 给出了 CeleX™ Sensor 芯片相机套件的基本工作原理。

CeleX™ Sensor 被激发后，将数据传输至 FPGA。PC 端的应用程序可以通过 FPGA 来读取数据，同样，PC 端的应用程序也可以配置 FPGA，并通过 FPGA 板提供的 API 配置传感器。本相机套件（CeleX™ 04-S）使用的是 Opal Kelly 的 FPGA 处理板 XEM6310-LX45。

关于 Opal Kelly XEM6310-LX45 详细信息，请参考 Opal Kelly 官网，地址如下:
https://library.opalkelly.com/library/FrontPanelAPI/classokCFrontPanel.html#a01d90a0ac728cf88b9637ab2b78546b5

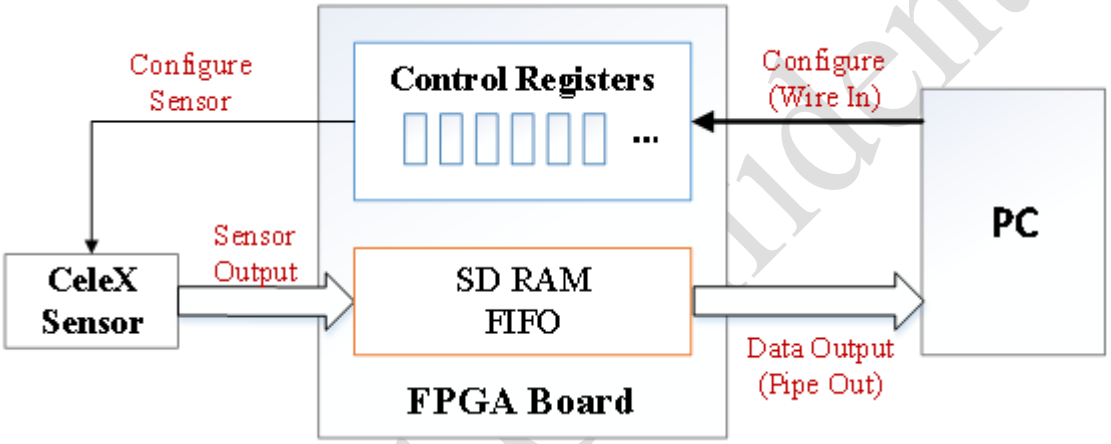


图 1-1 CeleX™ Sensor 芯片相机套件的基本工作原理

关于 CeleX™ Sensor 的工作原理以及数据格式，请参考章节 1.2。
关于控制寄存器的配置，请参考章节 4：附录。

1.1.2 术语

下表中列出了在本文档中出现的一些术语以及它们的解释。

No.	Terminology	Description
1	Full-Picture Mode	Celex™ Sensor 的一种工作模式，在该模式下，Sensor 会在一定的时间段里按照顺序输出每个像素点信息（从上到下，从左到右）。
2	Event Mode	Celex™ Sensor 的一种工作模式，在该模式下，Sensor 只检测亮度值发生变化像素点，然后把它标记成激活的像素点并输出。
3	FullPic-Event Mode	Celex™ Sensor 的一种工作模式，在该模式下，Sensor 会交替输出 Full-Picture 和 Event 的数据。
4	Optical-Flow Mode	Celex™ Sensor 的一种工作模式，在该模式下，Sensor 会输出光流信息。
5	Full Frame Transmission Time	Celex™ Sensor 传输一帧 Full-Picture 所需要的时

		间
6	Time Block	FPGA 里面的一个计时周期，用来标记每个 event 的时间戳，它的大小 T = 2^17，时间戳记满后会周而复始。
7	FullPic Frame Time	Full-Picture 数据的建帧时间间隔
8	Event Frame Time	Event 数据的建帧间隔
9	FullPic-Event Frame Time	FullPic-Event 数据的建帧时间间隔
13	Full Pic	本 SDK 中输出的 Full-Picture 图像帧
14	Event Binary Pic	本 SDK 中输出的 Event 的二值图像帧
15	Event Gray Pic	本 SDK 中输出的 Event 的灰度图像帧
16	Event Accumulated Gray Pic	本 SDK 中输出的 Event 的累加灰度图像帧

1.2 CeleX™ Sensor 的工作原理

1.2.1 CeleX™ Sensor 的工作模式

CeleX™ 是针对机器视觉而设计的智能图像传感器系列。传感器中的每一像素点能够独立自主地监测相对光强的变化，并在到达阈值时被激发发出被读出信号。行和列仲裁电路实时处理像素激发信号，并确保即使同时接收到多个请求时能够按有序的方式逐一处理。传感器依据被激发的事件，输出连续的异步数据流，而不是图像帧。CeleX™ 传感器监测的运动物体**速度不再受传统的曝光时间和帧速率限制**。它可以侦测高达万帧/秒昂贵高速相机才能获取到的高速物体运动信息，而且还能大幅降低后端处理量，结合相关配套后端处理软件，已可实现视频回放帧率的高速运动物体信息捕捉，回放帧率可以在 25~10000 帧/秒范围内进行选择。

CeleX™ 能并行输出三种模式数据：传统图像帧，动态数据流和全幅光流数据。这款传感器可以在各种领域中进行应用，如：辅助/自动驾驶，UAV，机器人，监控等。

本 SDK 可提供三种类型的 CeleX™ 传感器数据：*Full-Picture* 数据，*Event* 数据以及 *Optical-Flow* 数据。Full-Picture 与 Event 数据交替输出组成：*FullPic-Event* 数据。

1.2.1.1 Full-Picture 数据模式

当 CeleX™ Sensor 工作在 Full-Picture 数据模式下（如图 1-2 所示），Sensor 会在一定的时间段里按照顺序输出每个像素点信息（从上到下，从左到右），不管这个像素点有没有发生运动。这种工作方式与传统相机类似，我们也会获取到一帧清晰的图像帧。

在 Full-Picture 数据模式下，传输一帧 Full-Picture 的时间（Full Frame Transmission Time）是固定的，如果 Sensor 的工作频率为 25MHz，那么这个时间大约是 16ms。

建帧时间（Frame Time）是可以通过软件调整的（范围：20~320ms），用户可以通过调用 API 接口 setFullPicFrameTime 来修改这个时间。创建 Full-Picture 图像帧的方法见 1.2.3.1 章节。

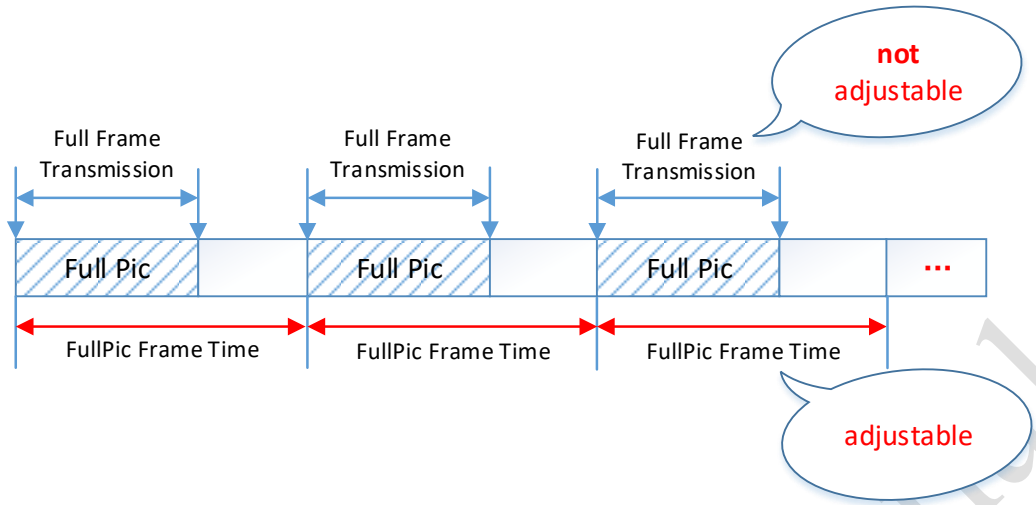


图 1-2 Full-Picture 数据模式

1.2.1.2 Event 数据模式

当 CeleX™ Sensor 工作在 Event 数据模式下（如图 1-3 所示），亮度值发生变化并超过阈值的像素点被激发，形成一个 Event，亮度值不发生变化或在阈值以内的像素点不被激发。每一个 Event 形成后，Sensor 会输出此 Event 被激发后的(X,Y,A,T)信息。

在 Event 模式下，通过 API，可以同时建立 6 种格式的图像帧：累加的图像帧（用最新变化的像素点的灰度值累加出来的灰度图） 和只有变化的像素点的二值图像帧（发生变化的像素点的灰度值标记为 255，没有发生变化的像素点的灰度值被标记为 0）等等。

在 Event 模式下，建帧方式也有两种，即有 Overlap 的方式和没有 Overlap 的方式，见图 1-3。在没有 Overlap 的方式下，每一帧数据都是全新的，帧与帧之间的数据没有重叠；在有 Overlap 的方式下，除了第一帧之外，后面的每一帧数据，都会叠加前一帧中一定时间间隔（Overlap Time）的数据。用户可以通过调用 API setOverlapTime 来设置设个时间，它可以调节的最大值为前一帧的 Frame Time。

在 Event 数据模式下，Time Block 的大小 T 是固定的（它是 FPGA 里面的一个计时周期，T = 2^17）。同样，建帧时间（Frame Time）是可以调整的（范围：1~1000ms），用户可以通过调用 API 接口 setEventFrameTime 来修改这个时间。创建 Event 图像帧的方法见 1.2.3.2 章节。

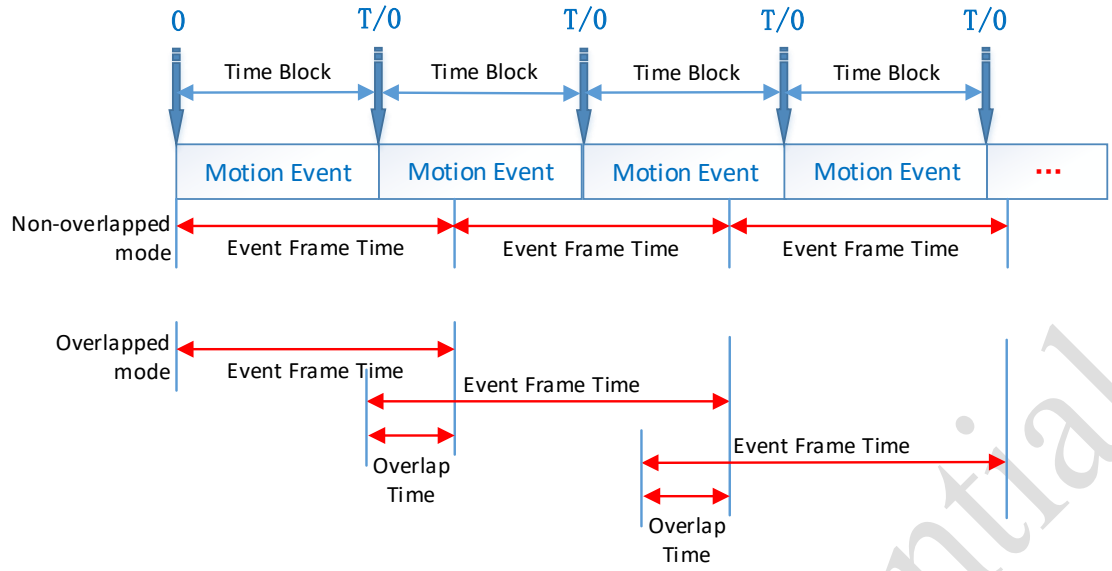


图 1-3 Event 数据模式

1.2.1.3 FullPic-Event 数据模式

当 CeleX™ Sensor 工作在 FullPic-Event 数据模式下（如图 1-4 所示），Sensor 会交替输出 Full-Picture 和 Event 的数据。即在一个 Frame Time 里，Sensor 会先按顺序输出一帧 Full-Picture 数据，紧接着就会输出 Event 数据。

同样，Full Frame Transmission Time 和 Time Block 是不可调整的，但是建帧时间（Frame Time）是可以调整的（范围：40~1000ms），用户可以通过调用 API 接口 setFEFrameTime 来修改这个时间。

在该模式下，用户可以同时获取到 Full-Picture 和 Event 图像帧，创建 Full-Picture 图像帧的方法与普通的 Full-Picture 模式一样，详见 1.2.3.1 章节，创建 Event 图像帧的方法与普通的 Event 模式一样，详见 1.2.3.2 章节。

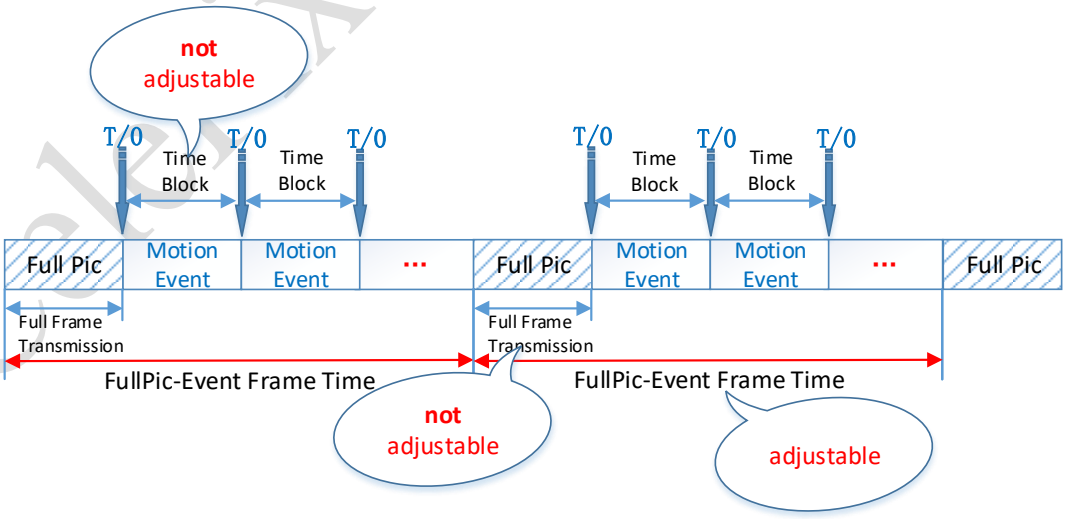


图 1-4 FullPic-Event 数据模式

1.2.1.4 Optical Flow 数据模式

当选择 SDK 中 Optical Flow 数据模式，本 SDK 调用 Sensor Event 数据模式下 Event 数据，进行 Optical Flow 分析。具体分析方式见 1.2.4。

1.2.1.5 数据模式切换

通过调用 setSensorMode 接口可以将 Sensor 在以上三种工作模式下进行切换。

1.2.2 数据格式

CeleX™ Sensor 套件的输出是一连串的像素事件。这些事件包含像素位置/坐标（X，Y），像素点事件触发时刻的绝对亮度的采样值（A）和像素激活时间（T）的信息。如下所示的表 1-1 中给出了符号 X，Y，A 和 T 的详细解释。

表 1-1 像素事件中的参数符号说明

Notation	min	max	Comments
X	0	767	像素列地址
Y	0	639	像素行地址
A	0	511	像素亮度值
T	0	2^17	像素激活时间： （1）时间轴被分成时间块 （2）T 在每个时间块都从 0 开始 （3）T 指示时间块的持续时间

在每一个时间块中，T 都是持续增加直到该时间块结束，然后清零，从零开始下一个时间块计数。图 1-5 给出了 T 的一个更加直观的描述。

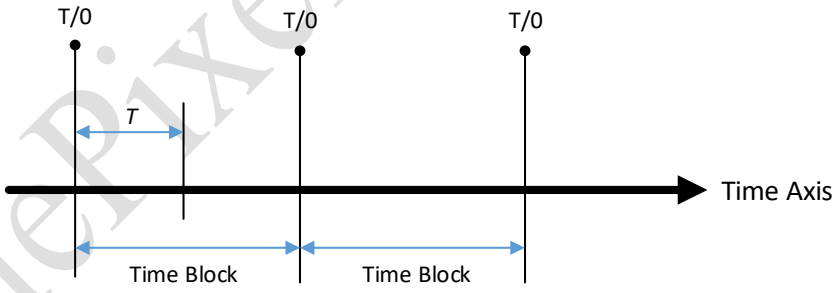


图 1-5 Time Block

CeleX™ Sensor 套件输出时间数据包含三种类型，即行事件，列事件和特殊事件。所有的事件都是 4 个字节，行事件携带关于 Y、T 的数据，列事件携带关于 X、A 的数据，特殊事件 4 bytes 全部为 1，表征时间块的结束。

具体输出方式为：1. Sensor 在所有存在激发像素点的行中随机选定一行，输出行地址和时间，即行事件被输出。2. 行事件输出完成后，Sensor 会按照一定顺序依次输出此行内激发像素点的列地址和 ADC 值，即列事件被依次输出。3. 此行中所有列事件输出完成后，重新开始步骤 1。图 1-6 描述了在 2 个 Special Event 之间的行事件和列事件的输出方式。下表 1-2 说明了事件类型的数据结构和格式。

表 1-2 事件类型的数据结构和格式

Event Type	Size	Description	Comments
行事件	4 bytes (32 bits)	4 bytes Byte 0: {1'b1, Y[6:0]} Byte 1: {1'b1, Y[9,7], T[3:0]} Byte 2: {1'b1, T[10:4]} Byte 3: {2'b10, T[16:11]}	行事件包含该行像素的行地址（Y）和激活时间（T）。
列事件	4 bytes (32 bits)	4 bytes Byte 0: {1'b0, X[6:0]} Byte 1: {1'b0, C[0], X[8,7], A[3:0]} Byte 2: {3'b000, A[8:4]} Byte 3: {8'b00000000}	列事件携带列地址（X），紧跟在行事件之后。它们具有像素点事件触发时刻的绝对亮度的采样值（A）信息。T 信息可以从前面的行事件中获取。
特殊事件	4 bytes (32 bits)	4 bytes Byte 0: {8'b11111111} Byte 1: {8'b11111111} Byte 2: {8'b11111111} Byte 3: {8'b11111111}	特殊事件表示时间块的结尾。此事件发生后，时间计数器重置为 0。时间块的长度是可配置的。

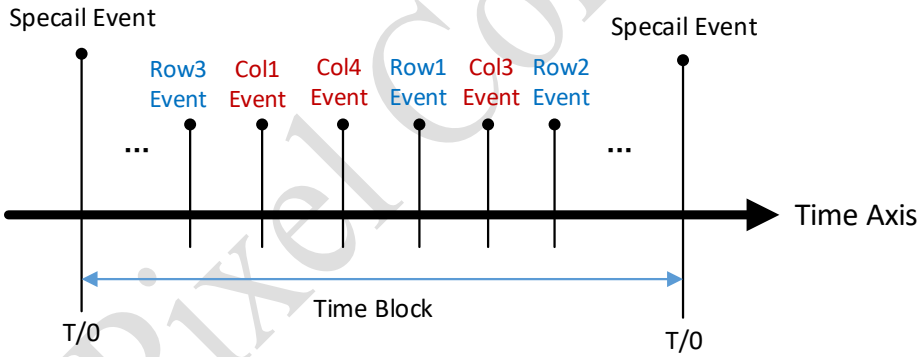


图 1-6 两个 Special Event 之间的行事件和列事件的输出方式

备注：

- 字节 0 表示事件中的第一个字节，字节 1 表示事件中的第二个字节，第三个和第四个字节也是如此。
- 对于行事件，我们将每个字节的第一位设置为“1”，10 位行地址（Y）由第一个字节中的 Y [6:0]和第二个字节中的 Y [9:7]组成。T 是由第二个字节的 T [3:0]，第三个字节的 T [10:4]和第四个字节的 T [16:11]组成。
- 对于列事件，我们将每个字节的第一位设置为“0”，第一个字节中的 X [6:0]和第二个字节中的 Y [8:7]用于形成 9 位列地址（X）。A 由第 2 个字节的 A [3:0]和第 3 个字节的 A [8:4]组成。第 4 个字节中未使用的位标记为“0”。
- 对于 A 值，假设事件触发阈值设为 10mv，某一像素点处光强对应的转换电压正在从 20mv 变化为 100mv。则像素点会在变化至 30mv 的时刻触发事件，且将 30mv 采样得到的 ADC 值输出。

- 除此之外，列事件的字节 1 中有一列左/右半平面指示符 C [0]。当这个控制位是“1”时，实际的列坐标将是（767-X）。当控制位为“0”时，实际列坐标将为 X。
- 对于特殊事件，所有 32 位都标记为“1”。
- 通过读取事件数据的第一位，可以很容易地区分行/列事件。

1.2.3 创建 Full Picture 和 Event 图像帧的方法

CeleX™ Sensor 套件上电之后，就可以输出连续的像素事件。如上一节所述，我们可以从事件中解码 X，Y，A，T 信息。接下来将介绍如何通过（X, Y, A, T）信息来创建图像帧。

1.2.3.1 创建 Full Picture 图像帧的方法

1) 调用 API 接口 *setFullPicFrameTime* 设置 Full Picture 建帧时间，指示 Sensor 每经过一定的时间间隔（即用户设置的 FullPic Frame Time），产生一个完整的图像帧（Full Picture）。这是一种兼容传统 Sensor 的工作模式，也即不管像素上有没有发生光线的变化，都能获取到它的灰度值。
2) 把 Sensor 的工作模式切换至 Full-Picture 模式。
3) 构建一个二维数组来表示 Full Pic，假设 M [640] [768]，它有 640 行，每行有 768 个像素将每个像素的亮度值初始化为 0。
4) 从 FPGA 获得的数据流中，解析获得的数据。每个事件 E 被解码为（X,Y,A,T）时，M [Y] [X]上的像素的亮度值为 A；在 Full-Picture 模式下，（T）信息是无效的。
5) 在新的 FullPic Frame Time 中，重复步骤 4。

上述建帧的过程是在 API 内部实现的，用户可以直接调用 getFullPicBuffer 这个函数来获得上述的 Full Picture 图像帧的内容。

1.2.3.2 创建 Event 图像帧的方法

在 Event 模式下，创建图像帧的方法有很多，下面仅介绍一下本 SDK 内部是如何创建 Event 二值图像帧（Event Binary Pic），Event 灰度图像帧（Event Gray Pic）以及 Event 累加灰度图像帧（Event Accumulated Gray Pic）的创建方法

1) 调用 API 接口 *setEventFrameTime* 设置 Event 建帧时间，指示本 SDK 将会把 Sensor 每经过一定的时间间隔（即用户设置的 Event Frame Time）内输出的 Event，组合起来产生一个 Event 二值图像帧。
2) 把 Sensor 的工作模式切换至 Event 模式。
3) 创建 3 个二维数组 M1 [640] [768]，M2 [640] [768]以及 M2 [640] [768]，分别用来表示 Event Binary Pic，Event Gray Pic 以及 Event Accumulated Gray Pic。并将 3 个二维数组中的每个像素的亮度值初始化为 0。

4) 从 FPGA 获得的数据流中，解析获得的数据。每个事件 E 被解码为（X,Y,A,T）时，M 1[Y] [X]上的像素的亮度值为 255，M 2[Y] [X]上的像素的亮度值为 A，M 3[Y] [X]上的像素的亮度值为 A。

5) 在新的 Event Frame Time 中，首先将二维数组 M1 和 M2 中的每个像素的亮度值为设置为 0，M3 保持不变；然后再重复步骤 4。也就是说，在每个 Event Frame Time 中，M1 和 M2 都是每次清零的，而 M3 数组则是在上一个 Event Frame Time 建立的数组的基础上更新的。

*上述建帧的过程是在 API 内部实现的，用户可以直接调用 getEventPicBuffer 这个函数来获得上述的各图像帧的内容。*为了帮助用户理解 Event Gray Pic 和 Event Accumulated Gray Pic 的区别，下面用一个 5 * 5 阵列来进一步说明上述过程，本文档仅列出前五个帧作为范例。

在图 1-7 中的每一帧表示 Event 模式下的 Gray Pic，在图 1-8 中的每一帧表示 Event 模式下的 Accumulated Gray Pic。在图 1-7 和图 1-8 里，“x”分别表示 Event 模式下发生变化个的像素的灰度值。红色的“x”表示在第一个 Event Frame Time 中变化的像素，蓝色的“x”表示在第二个 Event Frame Time 中变化的像素，绿色的“x”表示在第三个 Event Frame Time 中变化的像素，紫色的“x”表示在第四个 Event Frame Time 中变化的像素，黑色的“x”表示在第五个 Event Frame Time 中变化的像素。

在图 1-8 中，应该注意的是，当一个像素先前发生了变化，然后它又发生了变化，我们直接用新的灰度值代替旧的灰度值，例如，图示的位置（row0，col0）和位置（row2， col4）。

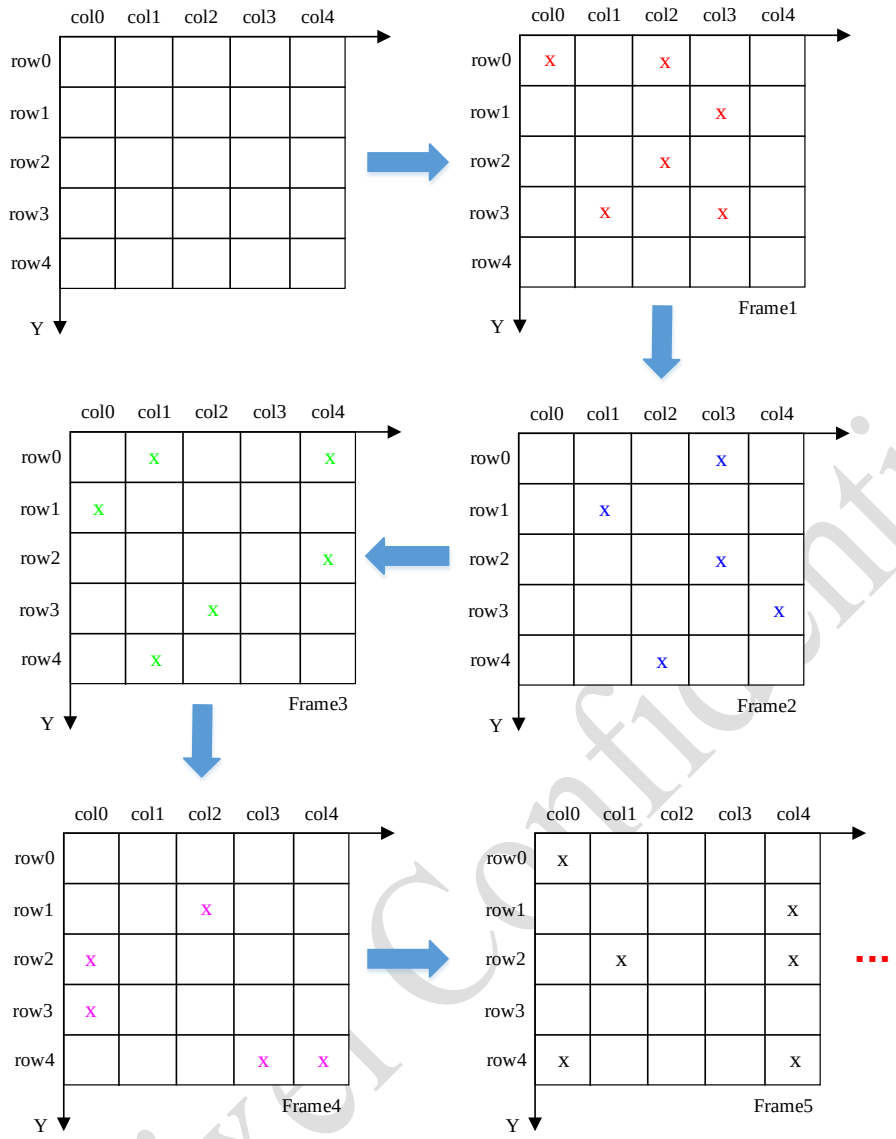


图 1-7 Event 模式下的灰度图像帧（Event Gray Pic）

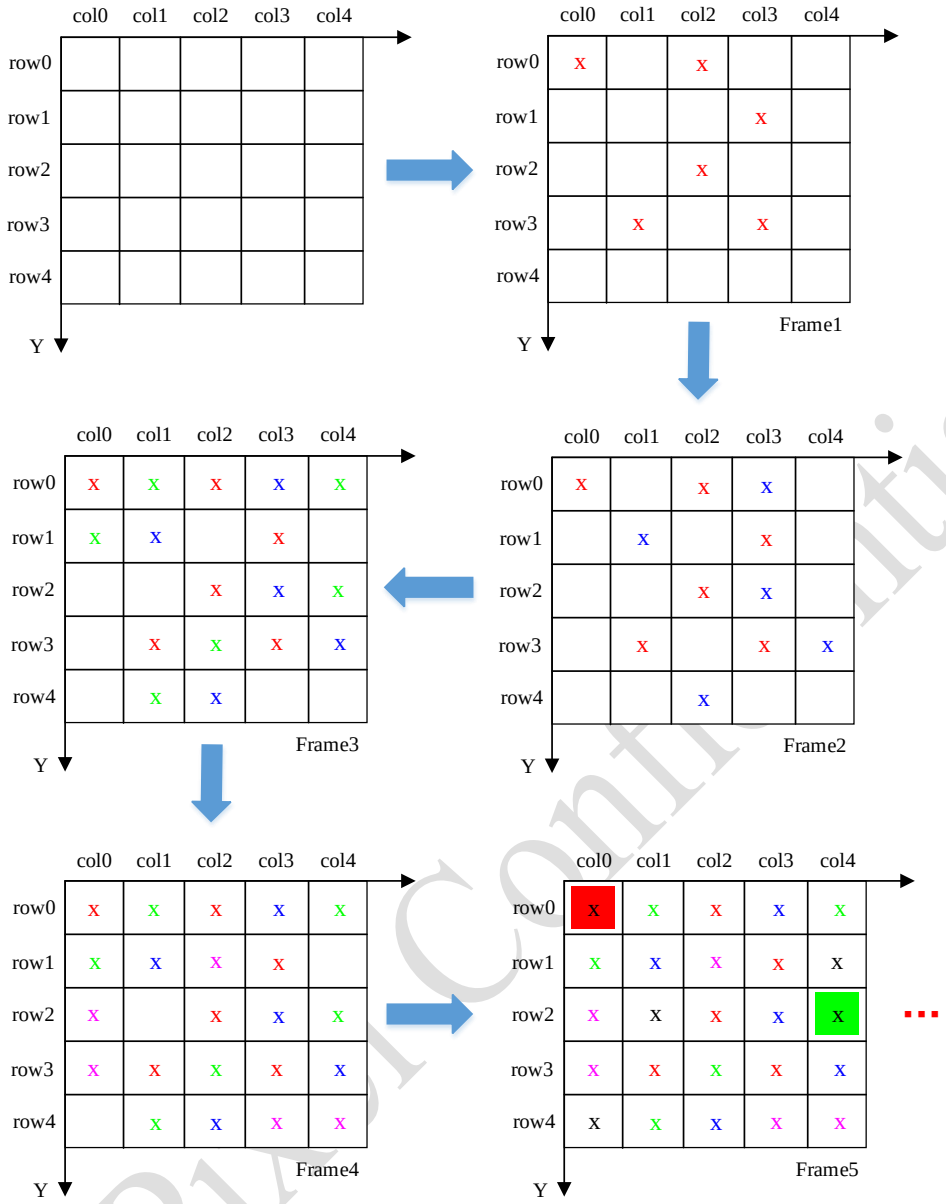


图 1-8 Event 模式下的累加灰度图像帧（Event Accumulated Gray Pic）

1.2.4 Optical Flow 数据

Optical Flow 是把发生在一段时间内（Latency Time）的像素点，按照时间先后顺序切成 N 个时间片（Slice），然后再给每个切片设置一个权重值（对应到图像上，即是一个灰度值），得到的带像素权重值的图像帧。

在 API 接口中调用 *enableOpticalFlow* 接口，启用 Optical Flow 功能，默认是关闭该功能的。用户通过调用 *getOpticalFlowDirectionPicBuffer* 和 *getOpticalFlowSpeedPicBuffer* 接口，可以获得 Optical Flow 数据，即每个像素的运动速度大小和运动方向。

具体建帧步骤为下：

1) 在时间轴上对 Event 模式的数据进行切帧，帧与帧之间可以有一定的时间域上的重叠，如下图 Optical-Flow Frame1，Optical-Flow Frame2；

2) 在每帧的持续时间（Latency）内按时间进行分片（Slice），各 Slice 之间不重合，本 SDK 中每个 Latency 有 255 个 Slice；

3) 对落在不同时间片内的数据点进行赋值，按照时间先后，赋予从小到大的数值，属于同一个 Slice 的数据点赋值相同，如：落在 Slice1 区间的像素们给与权重都为 1，落在 Slice2 区间的像素们给与权重都为 2，以此类推，落在 Slice255 区间的像素们给与权重都为 255，；

4) 把这些数据点赋值映射到 M [640] [768]数组中，获取一个 Optical Flow 图像权重帧，权重值使用灰度值来给与像素赋值，其中灰度值越大表示该像素点发生变化的时刻越新，越小越旧。

5) Optical Flow 在空间标识如下图 1-10，随着物体的运动轨迹，会形成一个坡；这个坡的倾斜方向跟物体的运动方向相同，而坡度跟运动速度有关，物体速度越快，坡度越小。我们内建了一个提取坡度和坡向的算法，分别是 *getOpticalFlowDirectionPicBuffer* 和 *getOpticalFlowSpeedPicBuffer*，用户也可以开发自己的算法来获取速度和方向信息。

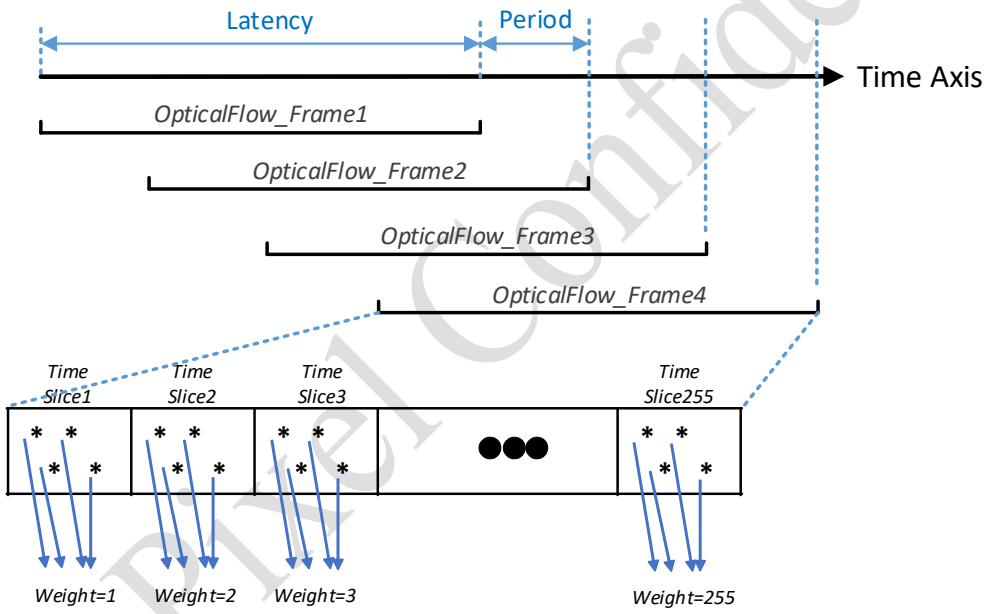


图 1-9 Optical Flow 的基本原理

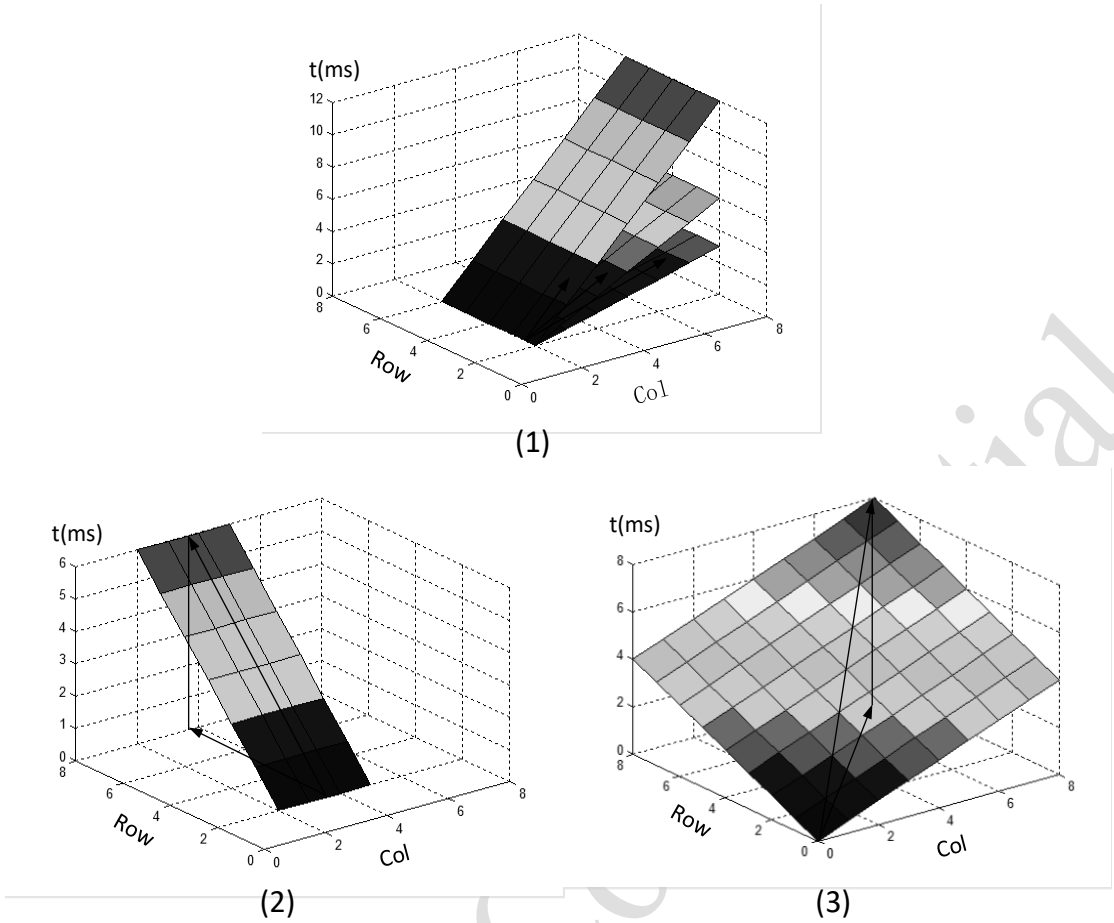


图 1-10 Optical Flow 坡度与坡向

2 API 描述

2.1 概述

CeleX API 提供了与 CeleX™ 传感器通信的 C ++接口。要使用此 API 构建应用程序，您需要将 include 目录中的文件拷贝到您的项目中，这些头文件中包含了我们提供的所有接口。同时，还需指定 CeleX.dll 和 CeleX.lib（Windows）或 libCeleX.so（Linux）（例如/ usr / local / lib）的路径。

CeleX4 封装了与 CeleX™ 传感器相关的接口，所以要使用该库，需要创建一个 CeleX4 实例，然后调用 *openSensor* 打开连接到 PC 上的传感器，再调用 *pipeOutFPGAData* 来读取 FPGA FIFO 中的数据，最后可以调用 *getFullPicBuffer* 和 *getEventPicBuffer* 来获取使用原始数据计算得到的各种图像帧。

CeleX4DataManager 是一个数据通知类，重载 *onFrameDataUpdated* 方法以接收各种图像帧已经建立完成的通知。

在 2.2 版中添加了 **dvs** 命名空间以放置 Event 处理的类和函数。需要注意的是，之前版中的类和函数并不在该命名空间之内。

Here are enums and structs with brief descriptions:

num emSensorMode {
FullPictureMode = 0,
EventMode = 1,
FullPic_Event_Mode = 2
};
	FullPictureMode - 在该模式下 Sensor 会在一定的时间段里按照顺序输出每个像素点信息（从上到下，从左到右）。
	EventMode - 在该模式下 Sensor 只检测亮度值发生变化像素点，然后把它标记成激活的像素点并输出。
	FullPic_Event_Mode - 在该模式下 Sensor 会交替输出 Full-Picture 和 Event 的数据。

enum emEventPicMode {
EventBinaryPic = 0,
EventAccumulatedPic = 1,
EventGrayPic = 2,
EventSuperimposedPic = 3,
EventDenoisedBinaryPic = 4,
EventDenoisedGrayPic = 5,
EventCountPic = 6,
};
	EventBinaryPic - Event 二值图像帧
	EventAccumulatedPic - Event 累加灰度图像帧
	EventGrayPic - Event 灰度图像帧
	EventSuperimposedPic - Event 叠加图像帧，即将 Event 二值图像帧叠加到 Full Pic 或者 Event 累加灰度图像帧上形成的图像帧
	EventDenoisedBinaryPic - 去噪后的 Event 二值图像
	EventDenoisedGrayPic - 去噪后的 Event 灰度图像
	EventCountPic - Event次数图像帧

typedef struct BinFileAttributes { int hour; int minute; int second; emSensorMode mode;	**hour, minute, second** - 录制的 bin 文件的时间长度 **mode** - 录制的 bin 文件的数据模式 **length** - 录制的 bin 文件大小

long length; }BinFileAttributes;	
typedef struct EventData { uint16_t col; uint16_t row; uint16_t brightness; uint32_t t; }EventData;	**col -** 像素所在的列 **row -** 像素所在的行 **brightness -** 像素所在的亮度值 **t -** 像素变化的时间

2.2 CeleX4 Class Reference

CeleX4 类封装跟 CeleX 第 4 代传感器有关的所有接口，如获取传感器输出的数据，切换传感器的工作模式和配置寄存器参数以控制传感器等功能。

Public Member Functions:

API Name	Description
openSensor	打开 Sensor，调用该接口可直接打开 Sensor
isSensorReady	判断 CeleX™ Sensor 初始化是否成功
isSdramFull	判断 SD RAM 中的数据是否阻塞
setSensorMode	设置 CeleX™ Sensor 的工作模式
getSensorMode	获取 Sensor 当前的工作模式
setFpnFile	设置创建图像时要使用的 FPN
generateFPN	生成 FPN
pipeOutFPGAData	把 FPGA 中的数据 Pop 出来，并存入 API 内部的数据缓存中
getFPGADataSize	获取 FPGA FIFO 中的已有的数据长度（字节数）
readDataFromFPGA	从 FPGA 中读取数据
getFullPicBuffer	获取 Full-Picture 模式下的灰度图像 buffer
getFullPicMat	获取 Full-Picture 模式下的灰度图像 mat
getEventPicBuffer	获取 Event 模式下的各种图像 buffer
getEventPicMat	获取 Event 模式下的各种图像 mat
getEventDataVector	获取每帧 Event 数据的数组,每个像素点的信息包括（X，Y，A，T）
setThreshold	设置触发 Event 的阈值（当像素点的光强变化大于此阈值时，像素点才能被标记为有效像素点/一个 Event）
getThreshold	获取当前触发 Event 的阈值
setContrast	设置对比度
getContrast	获取对比度
setBrightness	设置亮度
getBrightness	获取亮度
setLowerADC	设置可变亮度范围的下限
getLowerADC	获取可变亮度范围的下限

setUpperADC	设置可变亮度范围的上限
getUpperADC	获取可变亮度范围的上限
resetFPGA	复位 FPGA
resetSensorAndFPGA	复位 Sensor 以及 FPGA
enableADC	启用或禁用灰度值输出
trigFullPic	在 Event 模式生成一帧 Full Pic
setClockRate	设置 Sensor 的时钟频率
getClockRate	获取 Sensor 的时钟频率
setFullPicFrameTime	设置 CeleX™ Sensor 工作在 Full-Picture 模式下的建帧时间（Frame Time）
getFullPicFrameTime	获取 CeleX™ Sensor 工作在 Full-Picture 模式下的建帧时间（FullPic Frame Time）
setEventFrameTime	设置 CeleX™ Sensor 工作在 Event 模式下的建帧时间（Event Frame Time）
getEventFrameTime	获取 CeleX™ Sensor 工作在 Event 模式下的建帧时间（Frame Time）
setFEFrameTime	设置 CeleX™ Sensor 工作在 FullPic-Event 模式下的建帧时间（FullPic-Event Frame Time）
getFEFrameTime	获取 CeleX™ Sensor 工作在 FullPic-Event 模式下的建帧时间（Frame Time）
setOverlapTime	在创建 Event 图像帧时，设置帧与帧之间的叠加时间
getOverlapTime	获取 Overlap Time
setEventFrameParameters	设置在 Event 数据的建帧时间与建帧间隔
setFrameTimeRange	设置 FullPic-Event 模式下 Event 数据的范围
setTimeScale	设置时间尺度
setEventCountStepSize	设置 Event 模式下计数的步长
enableOpticalFlow	开启或关闭 Optical Flow 功能
isOpticalFlowEnabled	判断 Optical Flow 功能是否开启
setOpticalFlowLatencyTime	设置一个 Optical Flow Frame 的 Latency 时间，默认值为 100ms
setOpticalFlowSliceCount	设置一个 Optical Flow Frame 中的切片个数，默认值为 255
getOpticalFlowPicBuffer	获取 Optical Flow 数据图像 buffer
getOpticalFlowPicMat	获取 Optical Flow 数据图像 mat
getOpticalFlowDrirectionPicBuffer	获取在 Optical Flow 图像帧上计算出来的每个像素的方向 buffer
getOpticalFlowDirectionPicMat	获取在 Optical Flow 图像帧上计算出来的每个像素的方向 mat
getOpticalFlowSpeedPicBuffer	获取在 Optical Flow 图像帧上计算出来的每个像素的速度 buffer
getOpticalFlowSpeedPicMat	获取在 Optical Flow 图像帧上计算出来的每个像素的速度 mat
startRecording	开始录制 bin 数据
stopRecording	停止录制 bin 数据

openPlaybackFile	打开指定路径下的 bin 文件
readPlayBackData	从 bin 文件中读取指定字节长度的数据
getAttributes	获取bin文件的相关属性
convertBinToAVI	将 bin 文件转换为 AVI 视频
startRecordingVideo	开始录制 Sensor 数据（视频格式）
stopRecordingVideo	停止录制视频
enableAutoAdjustBrightness	开启或禁用自动调节亮度功能

2.2.1 openSensor

CeleX4::ErrorCode CeleX4::openSensor(string str)

Parameters

[in] **str** 打开 Sensor 需要的 top.bit 的路径。

Returns

NoError - Sensor 启动成功

InitializeFPGAFailed - 创始化 Sensor 失败

PowerUpFailed - 给 Sensor 上电失败

ConfigureFailed - 配置 Sensor 的工作模式失败

该方法用于启动 CeleX™ Sensor，当有多个设备时，需要指定每个 Sensor 启动所需的 top.bit 的路径；当只有一个 Sensor 时，可以指定该路径，也可以传空值，默认会使用可执行文件所在的目录下 top.bit 文件。示例代码如下：

```
//create a CeleX object
CeleX *pCelex = new CeleX;
//now, open the sensor device
if (pCelex != NULL)
    pCelex->openSensor("");
```

See also

isSensorReady

2.2.2 isSensorReady

bool CeleX4::isSensorReady()

Returns

CeleX™ Sensor 的初始化状态

该方法用于判断 CeleX™ Sensor 初始化是否成功。返回 true 表示 Sensor 已初始化成功，否则返回 false。

See also

openSensor

2.2.3 isSdramFull

bool CeleX4::isSdramFull()

Returns

FPGA 中 SDRAM 的状态

该方法用于判断 SDRAM 中的数据是否阻塞，如果数据阻塞，那么从 FPGA 中读取数据会失败。返回 true 表示 SDRAM 中的数据已阻塞，否则返回 false。

2.2.4 setSensorMode

void CeleX4::setSensorMode(emSensorMode mode)

Parameters

[in] **mode** Sensor 的工作模式

该方法用于设置 CeleX™ Sensor 的工作模式，关于工作模式的具体描述，参见 *enSensorMode* 的描述。

See also

getSensorMode

2.2.5 getSensorMode

emSensorMode CeleX4::getSensorMode()

Returns

Sensor 的工作模式

该方法用于设置 CeleX™ Sensor 的工作模式.

See also

setSensorMode

2.2.6 setFpnFile

bool CeleX4::setFpnFile(const string &fpnFile)

Parameters

[in] **fpnFile** 要使用的 FPN 文件的路径+名称

Returns

加载 FPN 文件成功与否的状态

该方法用于设置创建图像时要使用的 FPN 文件，返回 true 表示成功读取 FPN 文件中的数据，否则返回 false。

See also

generateFPN

2.2.7 generateFPN

void CeleX4::generateFPN(std::string fpnFile)

Parameters

[in] **fpnFile** 生成 FPN 文件的路径+名称

该方法用于生成 FPN 文件，固定模式噪声（FPN, Fixed Pattern Noise）是数字图像传感器上的特定噪声模式的术语，在较长的曝光镜头中经常可见，其中特定像素易于在一般背景噪声之上提供较亮的强度。

See also

setFpnFile

2.2.8 pipeOutFPGAData

void CeleX4::pipeOutFPGAData()

该方法用于从 FPGA 中读取数据，由于图像数据会实时写入 FPGA 中，所以使用者需要及时把 FPGA 中的数据 Pipe 出来，以免 FPGA 被阻塞。

See also

getFPGADataSize

readDataFromFPGA

2.2.9 getFPGADataSize

long CeleX4::getFPGADataSize()

Returns

FPGA FIFO 的数据长度（单位：字节）

该方法用于获取 FPGA FIFO 中的已有的数据长度（字节数）。

See also

pipeOutFPGAData

readDataFromFPGA

2.2.10 readDataFromFPGA

long CeleX4::readDataFromFPGA(long length, unsigned char *data)

Parameters

[in] **length** 要从 FPGA FIFO 中读取的数据长度

[out] **data** 存储数据的 buffer

Returns

读取到的数据长度或-1（读取数据失败）

该方法用于从 FPGA FIFO 中读取指定字节长度的数据，如果读取数据失败，则会返回-1。

See also

pipeOutFPGAData

getFPGADataSize

2.2.11 getFullPicBuffer

unsigned char *CeleX4::getFullPicBuffer()

Returns

Full-Picture 模式下的全幅图像帧

该方法用于获取Full-Picture模式下的一帧的Full Pic。实例代码如下：

```
//now, open the sensor device
if (pCelex != NULL)
    pCelex->openSensor("");
pCelex->setSensorMode(FullPictureMode);//set sensor mode
pCelex->setFpnFile("FPN.txt");//set FPN file
while (true)
{
    pCelex->pipeOutFPGAData();
    //get the fullpic buffer
    unsigned char* pFullPicBuffer = pCelex->getFullPicBuffer();
}
```

See also

getFullPicMat

2.2.12 getFullPicMat

cv::Mat CeleX4:: getFullPicMat()

Returns

Full-Picture 模式下 cv::Mat 格式的图像帧。

调用该方法可以获取Full-Picture模式下的一帧cv::Mat格式的Full Pic。示例代码如下：

```
//now, open the sensor device
if (pCelex != NULL)
    pCelex->openSensor("");
pCelex->setSensorMode(FullPictureMode);//set sensor mode
pCelex->setFpnFile("FPN.txt");//set FPN file
while (true)
```

```
{
    pCelex->pipeOutFPGAData();
    //get the fullpic mat
    cv::Mat fullPicMat = pCelex->getFullPicMat();
}
```

See also

getFullPicBuffer

2.2.13 getEventPicBuffer

unsigned char *CeleX4::getEventPicBuffer(emEventPicMode mode)

Parameters

[in] **mode** Event 模式下可输出的图像类型

Returns

Event 模式下指定图像类型的图像帧

该方法用于获取指定的 Event 图像类型的图像帧，它可以获取七种不同类型的图像，包括二值图、灰度图等等。更多图像类型可见 *emEventPicMode*。示例代码如下：

```
//now, open the sensor device
if (pCelex != NULL)
    pCelex->openSensor("");

pCelex->setFpnFile("FPN.txt");//set FPN file

while (true)

{

    pCelex->pipeOutFPGAData();

    //get the event buffer in the form of binary pic
    unsigned char* pEventBuffer = pCelex->getEventPicBuffer(EventBinaryPic);

}
```

See also

getEventPicMat

2.2.14 getEventPicMat

cv::Mat **CeleX4::** getEventPicMat(emEventPicMode **picMode**)

Parameters

[in] **picMode** Event模式下可输出的图像类型

Returns

Event 模式下指定图像类型的 cv::Mat 格式的图像帧。

该方法用于指定输出的 Event 图像类型，获取到相应图像类型 cv::Mat 格式的图像帧。该方

法可以获取七种不同类型的图像，包括二值图、灰度图等等。更多图像类型可见 *emEventPicMode*。

See also

getEventPicBuffer

2.2.15 getEventDataVector

bool CeleX4::getEventDataVector(std::vector<EventData>& data)

Parameters

[out] **data** 用于存储一帧Event数据的vector容器

Returns

如果 data 不为空则返回 true，否则返回 false。

该方法用于获取到每一帧 Event 数据的数组。默认的帧长为 60 毫秒。每个 Event 数据包含行、列、亮度以及时间信息。更多解释可见 *EventData*。该方法在实时获取数据或是离线读取 bin 文件时皆可用。示例参考代码如下：

```
//now, open the sensor device
if (pCelex != NULL)
    pCelex->openSensor("");
while (true)
{
    pCelex->pipeOutFPGAData(); //get event data in real-time
    std::vector<EventData> v;    //vector to store the event data
    pCelex->getEventDataVector(v);              //get the event data
}
```

2.2.16 setThreshold

void CeleX4::setThreshold(uint32_t value)

Parameters

[in] **value** 触发 Event 数据的阈值

该方法用于设置Event数据触发的阈值，当某个像素的光强度变化超过该阈值时，则该像素可以被标记为一个Event或激活的像素。该阈值越大，则被触发的像素会越少，反之会越多，它支持调节的范围为：25 ~ 200。

该方法只有在 Celex™ Sensor 输出 Event 数据时有效，如果 Seosor 工作在 Full-Picture 模式，不论该值多大，Sensor 都会输出一帧完整的 Full Pic。而且，由于该方法修改的是硬件的参数，所有只有在实时获取 Sensor 数据时有效，在回播 bin 文件时无效。

See also

getThreshold

2.2.17 getThreshold

uint32_t CeleX4::getThreshold()

Returns

触发 Event 数据的阈值

该方法用于获取已设置的触发 Event 数据的阈值大小。

See also

setThreshold

2.2.18 setContrast

void CeleX4::setContrast(uint32_t value)

Parameters

[in] **value** 跟图像对比度有关的配置值

该方法用于设置一个跟图像对比度有关的配置值的大小，它支持调节的范围为 0 ~ 1023。由于该方法修改的是硬件的参数，所有只有在实时获取 Sensor 数据时有效，在回播 bin 文件时无效。

See also

getContrast

2.2.19 getContrast

uint32_t CeleX4::getContrast()

Returns

The register value associated with the contrast of the image.

该方法用于设置一个跟图像对比度有关的配置值的大小。

See also

setContrast

2.2.20 setBrightness

void CeleX4::setBrightness(uint32_t value)

Parameters

[in] **value** 跟图像亮度有关的配置值

该方法用于设置一个跟图像亮度有关的配置值的大小，它支持调节的范围为 0 ~ 1023。该值设置的越大，Sensor 输出的图像越亮，越小，图像则越暗。由于该方法修改的是硬件的参数，所有只有在实时获取 Sensor 数据时有效，在回播 bin 文件时无效。

See also

getBrightness

.

2.2.21 getBrightness

void CeleX4::getBrightness(uint32_t value)

Returns

跟图像亮度有关的配置值

该方法用户获取跟图像亮度有关的配置值的大小。

See also

setBrightness

2.2.22 setLowerADC

void CeleX4::setLowerADC(uint32_t value)

Parameters

[in] **value** ADC 值的下限

该方法用于设置 ADC 值（触发像素事件时绝对亮度的样本值）的下限。小于下限值得 ADC 值都会被赋值为 0，而在下限值到上限值之间的 ADC 值将会被线性映射到[0, 255]。

See also

getLowerADC

2.2.23 getLowerADC

uint32_t CeleX4::getLowerADC()

Returns

ADC 值的下限

该方法用于获取 ADC 值（触发像素事件时绝对亮度的样本值）的下限。

See also

setLowerADC

2.2.24 setUpperADC

void CeleX4::setUpperADC(uint32_t value)

Parameters

[in] **value** ADC 值的上限

该方法用于设置 ADC 值（触发像素事件时绝对亮度的样本值）的下限。大于上限值的 ADC 值都会被赋值为 255，而在下限值到上限值之间的 ADC 值将会被线性映射到[0, 255]。

See also

getUpperADC

2.2.25 getUpperADC

uint32_t CeleX4::getUpperADC()

Returns

ADC 值的上限

该方法用于获取 ADC 值（触发像素事件时绝对亮度的样本值）的上限。

See also

setUpperADC

2.2.26 resetFPGA

void CeleX4::resetFPGA()

该方法用于清除 FPGA FIFO 中的数据。

2.2.27 resetSensorAndFPGA

void CeleX4::resetSensorAndFPGA()

该方法用于复位 Sensor 的各种配置参数，但不改变 Sensor 的工作模式、亮度以及对比度等信息，并清除 FPGA FIFO 中的数据。

2.2.28 enableADC

void CeleX4::enableADC(bool enable)

Parameters

[in] **enable** 开启或禁用输出 Event 数据灰度值的状态

该方法用于开启或禁用 Event 数据灰度值的输出，默认该功能是开启。一旦禁用了 ADC 的输出，那么 Event 数据的所有 A 值都为 0。

2.2.29 trigFullPic

void CeleX4::trigFullPic()

该方法用于在 Event 模式下触发 Sensor 输出一帧 Full Pic 数据，之后 Sensor 又继续输出 Event 数据。

2.2.30 setClockRate

void CeleX4::setClockRate(uint32_t value)

Parameters

[in] **value** Celex™ Sensor 的时钟工作频率，单位为 MHz

该方法用于设置的 Celex™ Sensor 的工作频率，Sensor 默认是工作在 25 MHz，它可以调节的范围为：2 ~ 50 MHz。该值越大表示 Sensor 可以检测到像素点的光强变化的速度越快。

See also

getClockRate

2.2.31 getClockRate

uint32_t CeleX4::getClockRate()

Returns

Celex™ Sensor 的时钟工作频率，单位为 MHz

该方法用于获取的 Celex™ Sensor 的工作频率，默认值 25 MHz。

See also

setClockRate

2.2.32 setFullPicFrameTime

void CeleX4::setFullPicFrameTime(uint32_t msec)

Parameters

[in] **msec** Full-Picture 模式下的帧长，单位为 ms

该方法用于设置 CeleX™ Sensor 工作在 Full-Picture 模式时的帧长，详见 1.2.1.1 章节。需要注意的是，该设置改变的是硬件的参数，一旦修改了这个值，那么 Sensor 将会按照这个时间长度输出一帧 Full Pic。

See also

getFullPicFrameTime

2.2.33 getFullPicFameTime

uint32_t CeleX4::getFullPicFrameTime()

Returns

Full-Picture 模式下的帧长，单位为 ms

该方法用于获取 CeleX™ Sensor 工作在 Full-Picture 模式时的帧长。

See also

setFullPicFrameTime

2.2.34 setEventFrameTime

void CeleX4::setEventFrameTime(uint32_t msec)

Parameters

[in] **msec** Event 模式下的帧长，单位为 ms

该方法用于设置 CeleX™ Sensor 工作在 Event 模式时的帧长，详见 1.2.1.2 章节。需要注意的是，该设置并不改变硬件参数，它仅仅是改变了软件用 Event 数据建帧的时间长度。

See also

getEventFrameTime

2.2.35 getEventFrameTime

uint32_t CeleX4::getEventFrameTime()

Returns

Event 模式下的帧长，单位为 ms

该方法用于获取 CeleX™ Sensor 工作在 Event 模式时的帧长。

See also

setEventFrameTime

2.2.36 setFEFrameTime

void CeleX4::setFEFrameTime(uint32_t msec)

Parameters

[in] **msec** FullPic-Event 模式下的帧长，单位为 ms

该方法用于设置 CeleX™ Sensor 工作在 FullPic-Event 模式时的帧长，详见 1.2.1.3 章节。需要注意的是，该设置改变的是硬件的参数，一旦修改了这个值，那么 Sensor 将会按照这个时间长度输出一帧 Full Pic 和一段时间的 Event 数据。由于传输一帧 Full Pic 的时间是固定的，那么修改这个值，真正改变的是 FullPic-Event 模式中输出 Event 数据的时间长度。

See also

getFEFrameTime

2.2.37 getFEFrameTime

uint32_t CeleX4::getFEFrameTime()

Returns

FullPic-Event 模式下的帧长，单位为 ms

该方法用于获取 CeleX™ Sensor 工作在 FullPic-Event 模式时的帧长。

See also

setFEFrameTime

2.2.38 setOverlapTime

void CeleX4::setOverlapTime(uint32_t msec)

Parameters

[in] **msec** 两个 Event 图像帧之间的重叠时间，单位为 ms

该方法用于设置两个 Event 图像帧之间的重叠时间，如果这个时间为 0，则表示两个 Event 图像帧之间没有重叠（即 1.2.1.2 章节中描述的 non-overlapped 建帧）。如果这个时间大于 0，则表示连个 Event 图像帧之间有一段时间的重叠信息（即即 1.2.1.2 章节中描述的 overlapped 建帧）。

See also

getOverlapTime

setEventFrameParameters

2.2.39 getOverlapTime

uint32_t CeleX4::getOverlapTime()

Returns

两个 Event 图像帧之间的重叠时间，单位为 ms

该方法用于获取两个 Event 图像帧之间的重叠时间。

See also

setOverlapTime

2.2.40 setEventFrameParameters

void CeleX4::setEventFrameParameters(uint32_t frameTime,

uint32_t intervalTime

)

Parameters

[in] **frameTime** Event模式下的建帧时间

[in] **intervalTime** 两帧之间的建帧间隔

该方法用于设置 Event 模式下的建帧时长以及建帧间隔时长。两个时长单位都为毫秒。通过调整建帧时长和建帧间隔可以设置两帧的重叠时间。例如，如果建帧时长与建帧间隔相同，则两帧之间没有重叠。反之，建帧时长和建帧间隔时长的差则为两帧之间的重叠时间。

2.2.41 setFrameLengthRange

void CeleX4::setFrameLengthRange(float startRatio,

float endRatio

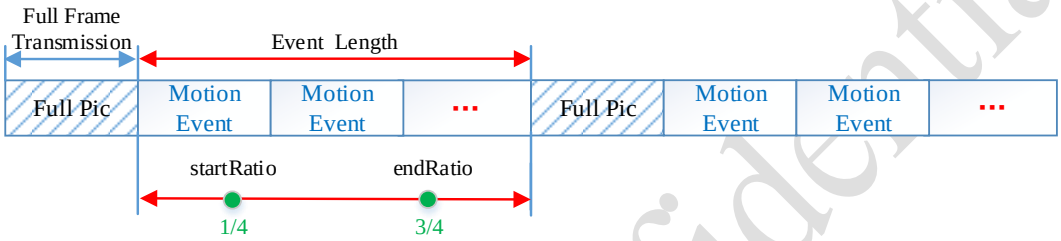
)

Parameters

[in] **startRatio** 帧数据开始的位置比例

[in] **endRatio** 帧数据结束的位置比例

该方法用于设置 Event 数据的帧开始和结束位置比例（仅 FullPic-Event 模式），它们的默认值分别是 0 和 1，即是将 Event 段的所有数据建立成一个图像帧，如下图所示，则将会用 Event 段数据的 1/4 到 3/4 之间的数据建立成一个图像帧。



2.2.42 setTimeScale

void CeleX4::setTimeScale(float scale)

Parameters

[in] **scale** 时间尺度

该方法用于根据所设置的时间尺度将每个 event 数据的时间信息进行重映射。时间尺度的单位是毫秒。

2.2.43 setEventCountStepSize

void CeleX4::setEventCountStepSize(uint32_t size)

Parameters

[in] **size**像素每变化一次，event计数的步长

该方法用于设置 event 计数的步长。像素亮度每变化一次，该像素的灰度则变化一个步长。对于不同的场景，通过设置步长，EventCountPic 可以获得更好的可视化效果。默认步长为 9。

2.2.44 enableOpticalFlow

void CeleX4::enableOpticalFlow(bool enable)

Parameters

[in] **enable** 开启或禁用光流功能的状态

该方法用于开启或关闭 Event 或 FullPic-Event 模式下光流功能，关于光流的描述请参考 1.2.4 章节。

See also

isOpticalFlowEnabled

2.2.45 isOpticalFlowEnabled

bool CeleX4::isOpticalFlowEnabled()

Returns

光流功能是否开启的状态

该方法用于判断光流功能是否开启，开启返回 true，否则返回 false，关于光流的详细信息，请参见 1.2.4 章节。

See also

enableOpticalFlow

2.2.46 setOpticalFlowLatencyTime

void CeleX4::setOpticalFlowLatencyTime(uint32_t msec)

Parameters

[in] **msec** 光流数据的帧长，单位为 ms

该方法用于设置一个 Optical Flow Frame 的 Latency 时间，默认值为 100ms，关于 Optical Flow 的描述请参考 1.2.4 章节。

See also

setOpticalFlowSliceCount

2.2.47 setOpticalFlowSliceCount

void CeleX4::setOpticalFlowSliceCount(uint32_t count)

Parameters

[in] **count** 光流帧的切片个数

该方法用于设置一个光流帧的切片个数，默认值为 255，关于光流的描述请参考 1.2.4 章节。

See also

setOpticalFlowLatencyTime

2.2.48 getOpticalFlowPicBuffer

unsigned char* CeleX4::getOpticalFlowPicBuffer()

Returns

Event 或 FullPic-Event 模式下的光流图像帧

该方法用于获取 Event 或 FullPic-Event 模式下的光流图像帧，关于光流的详细信息，请参见 1.2.4 章节。需要注意的是，要获取光流数据需要先开启光流功能，具体的示例代码如下：

```
//now, open the sensor device
if (pCelex != NULL)
    pCelex->openSensor("");
pCelex->enableOpticalFlow(true); //enable optical-flow option
while (true)
{
    pCelex->pipeOutFPGAData();
    //get the optical-flow buffer
    unsigned char* pFullPicBuffer = pCelex->getOpticalFlowPicBuffer();
}
```

See also

getOpticalFlowPicMat

getOpticalFlowDirectionPicBuffer

getOpticalFlowSpeedPicBuffer

2.2.49 getOpticalFlowPicMat

cv::Mat **CeleX4::** getOpticalFlowPicMat ()

Returns

Event 或 FullPic-Event 模式下 cv::Mat 格式的光流图像帧

该方法用于获取 Event 模式下的一帧 cv::Mat 格式的光流图像。更多关于光流的概念可见章节 1.2.4。

See also

getOpticalFlowPicBuffer

getOpticalFlowDirectionPicMat

getOpticalFlowSpeedPicMat

2.2.50 getOpticalFlowDirectionPicBuffer

unsigned char* CeleX4::getOpticalFlowDirectionPicBuffer()

Returns

Event 或 FullPic-Event 模式下光流方向的图像帧。

该方法用于获取在 Optical Flow 图像帧上计算出来的每个像素的方向 Buffer，为了在图像上可以直接显示，我们把角度值[0, 359]映射到[0, 255]区间了。更多关于光流图像的方向图可见章节 1.2.4。

See also

getOpticalFlowDirectionPicMat

getOpticalFlowPicBuffer

getOpticalFlowSpeedPicBuffer

2.2.51 getOpticalFlowDirectionPicMat

cv::Mat CeleX4::getOpticalFlowDirectionPicMat()

Returns

Event 或 FullPic-Event 模式下 cv::Mat 格式表示光流方向的图像帧。

该方法用于获取到计算出光流图像上每个像素方向后的图像。为了在图像上可以直接显示方向图像，将角度值[0, 359]映射到颜色[0, 255]区间。更多关于光流图像的方向图可见章节 1.2.4。

See also

getOpticalFlowDirectionBuffer

getOpticalFlowPicMat

getOpticalFlowSpeedPicMat

2.2.52 getOpticalFlowSpeedPicBuffer

unsigned char* CeleX4::getOpticalFlowSpeedPicBuffer()

Returns

Event 或 FullPic-Event 模式下光流速度的图像帧

调用该方法可以获取到计算出光流图像上每个像素速度后的图像。为了在图像上可以直接显示方向图像，将速度值映射到颜色[0，255]区间。更多关于光流图像的速度图可见章节 1.2.4。

See also

getOpticalFlowSpeedPicMat

getOpticalFlowPicBuffer

getOpticalFlowDirectionPicBuffer

2.2.53 getOpticalFlowSpeedPicMat

cv::Mat CeleX4:: getOpticalFlowSpeedPicMat ()

Returns

Event 或 FullPic-Event 模式下 cv::Mat 格式表示光流速度的图像帧。

调用该方法可以获取到计算出光流图像上每个像素速度后的图像。为了在图像上可以直接显示方向图像，将速度值映射到颜色[0，255]区间。更多关于光流图像的速度图可见章节 1.2.4。

See also

getOpticalFlowSpeedBuffer

getOpticalFlowPicMat

getOpticalFlowDirectionPicMat

2.2.54 startRecording

void CeleX4::startRecording(std::string filePath)

Parameters

[in] **filePath** 存放录制的 bin 文件的路径

该方法用于开始录制 Sensor 的原始数据（未经任何处理的数据），并存储在用户指定的文件中。录制时 Sensor 工作在什么模式，那么录制的 bin 文件就是该模式的数据。

See also

stopRecording

2.2.55 stopRecording

void CeleX4::stopRecording()

该方法用于停止录制 Sensor 的数据，并停止向 bin 文件中写数据。

See also

startRecording

2.2.56 openPlaybackFile

bool CeleX4::openPlaybackFile(string filePath)

Parameters

[in] **filePath** 要回播的 bin 文件的路径+名称

Returns

是否成功打开要回播的 bin 文件的状态

该方法用于打开一个用户指定的 bin 文件，成功打开文件返回 true，否则返回 false。

See also

readPlayBackData

2.2.57 readPlayBackData

bool CeleX4::readPlayBackData(long length)

Parameters

[in] **length** 每次从 bin 文件中要读取字节长度，默认值为 1968644

Returns

是否已经读到文件结尾

该方法用于从打开的 bin 文件中读取指定字节长度的数据，并存入 API 内部的数据缓存中，如果已经读到文件末尾处，则返回 true，否则返回 false。需要注意的是，调用该接口之前，要先调用 *openPlaybackFile* 接口打开一个 bin 文件。

See also

openPlaybackFile

2.2.58 getAttributes

BinFileAttributes CeleX4:: getAttributes(std::string& binFile)

Parameters

[in] **binFile** bin文件的路径

Returns

bin 文件属性的结构体

调用此方法可以获取到 bin 文件的相关属性。主要包括 bin 文件的录制时长、bin 文件的类型、文件的长度。更多关于 bin 文件的属性返回值可见 *BinFileAttributes*。

2.2.59 convertBinToAVI

```
void CeleX4:: convertBinToAVI (std::string binFile,
                               emEventPicMode picMode,
                               uint32_t frameTime,
                               uint32_t intervalTime,
                               cv::VideoWriter writer
                               )
void CeleX4:: convertBinToAVI (std::string binFile,
                               cv::VideoWriter writer
                               )
```

Parameters

[in] **binFile** bin文件的路径

[in] **picMode** Event模式下可输出的图像类型

[in] **frameTime** Event模式下的建帧时间

[in] **intervalTime** 两帧之间的建帧间隔

[in] **writer** 用于视频写入的VideoWriter

调用该方法可以将bin文件转换为AVI类型视频文件。转换时需要指定输出的图像的类型、建帧时间与帧间隔时间。建帧时间和帧间隔的只适用于转换Event模式下的数据，其时间单位都为毫秒。目前，支持输出的图像类型主要有7种，详细图像类型可见 *emEventPicMode*。

Note

仅限转换 Event 模式下的数据时使用。如果需要转换 Full-Picture 模式或是 FullPic-Event 模式下的 Full Pic 图像，可以使用只有两个输入参数的接口。

2.2.60 startRecordingVideo

void CeleX4::startRecordingVideo(std::string filePath,
std::string fullPicName,
std::string eventName,
int fourcc,
double fps
)

Parameters

[in] **filePath** 存放录制的视频的路径

[in] **fullPicName** 录制的视频（Full Pic）的名称

[in] **eventName** 录制的视频（Event Binary Pic）的名称

[in] **fourcc** 用于压缩帧的 4 字符编解码器代码，opencv 参数

[in] **fps** 创建的视频流的帧率，opencv 参数

该方法用于开始录制 Sensor 的图像帧数据（处理过的数据），并以视频格式存储在用户指定的目录中。只有当 Sensor 工作 FullPic-Event 模式时，才会同时录制 Full Pic 和 Event 的视频，在 Full-Picture 和 Event 时，只分别录制 Full Pic 视频（eventName 可以为空）和 Event 视频（fullPicName 可以为空）。

See also

stopRecordingVideo

2.2.61 stopRecordingVideo

void CeleX4::stopRecordingVideo()

该方法用于停止录制视频格式的 Sensor 数据。

See also

startRecordingVideo

2.2.62 enableAutoAdjustBrightness

void CeleX4::enableAutoAdjustBrightness(bool enable)
Parameters

[in] **enable** 开启或禁用自动调节亮度的状态

该方法用于开启或禁用自动调节图像亮度功能，默认该功能是未开启的。开启这个功能后，API 内部就会根据外界的光强自动调用 *setBrightness* 接口调节 Sensor 输出的图像亮度。

See also

setBrightness

2.3 CeleX4DataManager Class Reference

该类用来通知图像帧数据已经处理完成，此时，可以获取 API 已经的处理好的数据来做各种算法或显示。要接收这些通知，需要从此类派生自己的 DataManager 子类，并重载 *onFrameDataUpdated* 方法，具体的实例代码如下：

```
#include "celex.h"
#include "celexdatamanager.h"
#include "celex4processeddata.h"
#include <opencv2/opencv.hpp>
#include <Windows.h>
#define FPN_PATH       "./FPN.txt"
using namespace std;
using namespace cv;
class SensorDataObserver : public CeleX4DataManager
{
public:
	SensorDataObserver(CX4SensorDataServer* pServer)
	{
		m_pServer = pServer;
		m_pServer->registerData(this, CeleX4DataManager::CeleX_Frame_Data);
	}
	~SensorDataObserver()
	{
		m_pServer->registerData(this, CeleX4DataManager::CeleX_Frame_Data);
	}
	virtual void onFrameDataUpdated(CeleX4ProcessedData* pSensorData);//overrides Observer operation

	CX4SensorDataServer* m_pServer;
};

void SensorDataObserver::onFrameDataUpdated(CeleX4ProcessedData* pSensorData)
{
	if (NULL == pSensorData)
```

```
            return;
        //get buffers when sensor works in EventMode
        if (pSensorData->getEventBasePic(EventBinaryPic))
        {
            cv::Mat matFullPic(640, 768, CV_8UC1, pSensorData->getEventBasePic(EventAccumulatedPic));
//event accumulative pic
            cv::Mat matEventPic(640, 768, CV_8UC1, pSensorData->getEventBasePic(EventBinaryPic)); //event
binary pic
            cv::imshow("FullPic", matFullPic);
            cv::imshow("EventPic", matEventPic);
            cvWaitKey(30);
        }
}

int main()
{
        CeleX* pCelexSensor = new CeleX;
        if (NULL == pCelexSensor)
            return 0;
        pCelexSensor->openSensor("");
        pCelexSensor->setFpnFile(FPN_PATH);
        emSensorMode sensorMode = FullPic_Event_Mode; //EventMode, FullPic_Event_Mode or
FullPictureMode
        pCelexSensor->setSensorMode(sensorMode);
        SensorDataObserver* pSensorData = new SensorDataObserver(pCelexSensor->getSensorDataServer());
        while (true)
        {
            pCelexSensor->pipeOutFPGAData();
            Sleep(10);
        }
        return 1;
}
```

2.4 CeleX NameSpace Reference

所有图像处理相关的类和方法都在 namespace **dvs** 下，可以使用 dvs:: specifier 或者 using namespace dvs 两种方法调用 dvs 里面的函数，示例如下：

```
#include "eventproc.h"
…
dvs::segmentationByMultislice(multislicebyte, ratio, segimage);
…
```

or

```
#include "eventproc.h"
```

```
using namespace dvs;
…
segmentationByMultislice(multislicebyte, ratio, segimage);
…
```

2.4.1 denoisingByNeighborhood

void dvs::denoisingByNeighborhood(const cv::Mat& countEventImg,

cv::Mat& denoisedImg

)

Parameters

[in] **countEventImg** Event模式下的计数图像

[out] **denoisedImg** 去噪后的二值图像

调用该方法可以对Event模式下的Event count图像进行领域去噪。包括时域和空间域上的去噪。

2.4.2. denoisingMaskByEventTime

int dvs::denoisingMaskByEventTime(const cv::Mat& countEventImg,

double timelength,

cv::Mat& denoiseMaskImg

)

Parameters

[in] **countEventImg** Event模式下的计数图像

[in] **timelength** 时间长度

[out] **denoisedImg** 去噪后的二值图像

调用该方法可以对Event模式下的Event count图像进行Mask去噪。该去噪方法将消除灰度值小于阈值的噪声。去噪时的阈值与时间长度的设置有关。

3. 附录

3.1. Opal Kelly FPGA 板上的控制寄存器

应用程序可以配置 Opal Kelly FPGA 板上的控制寄存器，而这些控制寄存器将配置传感器的行为，以下表格 4-1 是使用的可配置寄存器列表。

表 4-1 配置寄存器列表

Registor	Bit	Command Name	Address	Value	Mask	Description
0x00	[0]	Reset All	0x00	0x01	0x01	Reset Sensor & FPGA
		Dereset All	0x00	0x00	0x01	
	[1]	Reset FPGA	0x00	0x02	0x02	Reset FPGA
		Dereset FPGA	0x00	0x00	0x02	
	[4]	Forcefire ON	0x00	0x10	0x10	Trigger a full picture
		Forcefire OFF	0x00	0x00	0x10	
	[5]	set GAIN to #value#	0x00	value	0x20	
	[6]	SetADC Enalbe	0x00	0x40	0x40	ADC enable
		SetADC Disable	0x00	0x00	0x40	ADC disable
0x02	[7:0]	set Clock_M to #value#	0x02	value	0x00FF	Clock_M
	[15:8]	set Clock_D to #value#	0x02	value	0xFF00	Clock_D
	[16]	Clock_APPLY OFF	0x02	0x00	0x10000	Clock_APPLY
		Clock_APPLY ON	0x02	0x10000	0x10000	
0x03	[0]	VDD_IO_EN OFF	0x03	0x00	0x01	VDD_IO_EN
		VDD_IO_EN ON	0x03	0x01	0x01	
	[1]	VDD_RB_EN OFF	0x03	0x00	0x02	VDD_RB_EN
		VDD_RB_EN ON	0x03	0x02	0x02	
	[2]	VDD_CORE_EN OFF	0x03	0x00	0x04	VDD_CORE_EN
		VDD_CORE_EN ON	0x03	0x04	0x04	
0x04	[17:8]	set resolution to #value#	0x04	value	0x3FF00	SPI value
		set EVT_VL to #value#	0x04	value	0x3FF00	0x3FF00
		set EVT_VH to #value#	0x04	value	0x3FF00	0x3FF00
		set RAMP- to #value#	0x04	value	0x3FF00	0x3FF00
		set RAMP+ to #value#	0x04	value	0x3FF00	0x3FF00
		set REF- to #value#	0x04	value	0x3FF00	0x3FF00
		set REF+ to #value#	0x04	value	0x3FF00	0x3FF00
		set REF-H to #value#	0x04	value	0x3FF00	0x3FF00
		set REF+H to #value#	0x04	value	0x3FF00	0x3FF00

		set EVT_DC to #value#	0x04	value	0x3FF00	0x3FF00
		set LEAK to #value#	0x04	value	0x3FF00	0x3FF00
		set CDS_DC to #value#	0x04	value	0x3FF00	0x3FF00
		set CDS_V1 to #value#	0x04	value	0x3FF00	0x3FF00
		set PixBias to #value#	0x04	value	0x3FF00	0x3FF00
	[23:20]	set DAC_CHANNEL to #fixed value#	0x03	value	0xF00000	Sensor SPI address
0x05	[0]	DAC_APPLY OFF	0x05	0x00	0x01	DAC_APPLY
		DAC_APPLY ON	0x05	0x01	0x01	
	[1]	SPI_DEFAULT OFF	0x05	0x00	0x02	SPI_DEFAULT
		SPI_DEFAULT ON	0x05	0x02	0x02	