Camera Motion Metadata Track

Status: (Draft)

Overview

This doc describes of a simple timed metadata track by which MP4 containers may embed detailed metadata about the camera motion during the video capture.

Typical video capture devices have additional sensors that can provide information about how the camera devices are moving. For example,

- Mobile phones typically contain gyroscope, accelerometer, magnetometer and GPS.
- Sensor fusion can be used to track the 3DoF pose of the camera devices.
- SLAM can be used to track the 6DoF pose of the camera devices (e.g. Tango).
- Exposure information can be used to interpolate per-scanline motion.

These rich camera motion metadata can be saved as video metadata for advanced post-processing and various applications. For example:

- Frame-level rotation information can be used to stabilize videos, and scanline-level motion data allows to reduce rolling shutter effects.
- IMU readings and derived 3Dof poses can be used to evaluate time alignment and geometric alignment between IMU and camera

Below is the specification for a CAmera Motion Metadata (CAMM) track, which includes a new sample entry for indicating the existence of the track and the definition of the sample data.

SampleEntry

The video file should contain the following sample entry box to indicate the custom metadata track, and the subComponentType of the track should be set to 'meta'.

Camera Motion Metadata Sample Entry (camm)

Definition
Box Type: `camm`
Container: `stsd`
A sample entry indicating the data track that saves the camera motion.

```
Syntax
aligned(8) class CameraMotionMetadataSampleEntry extends SampleEntry('camm') {
}
```

Data Format

The metadata track contains a stream of metadata samples. The presentation time (pts) of the data samples should reflect the capture timestamp of the data in the common clock as video. The data samples are represented as bitstream in the following format:

Fields:	Units	Description:
uint16 reserved;		Reserved. Should be 0.
uint16 type;		The type of the data packet.
switch (type) {		

case 0:
  float angle_axis[3];
break;
	radians	Angle axis orientation in radians representing the rotation from local camera coordinate to world coordinate system. The world coordinate system is defined by applications..
		
		Let M be the 3x3 rotation matrix corresponding the the angle axis vector. For any ray X in local coordinate system, ray direction in world coordinate is M * X.
		
		This information can be obtained by running 3DoF sensor fusion on the camera device. After integrating the IMU readings, only the integrated global orientation needs to be recorded.

case 1:
  int32 pixel_exposure_time;
  int32 rolling_shutter_skew_time;
break;
	nanoseconds	This metadata is per video frame. The pts of this metadata should be the exposure start of the first used scanline in a video frame.
		
		pixel_exposure_time_ns is the exposure time for a single pixel in nanoseconds and rolling_shutter_skew_time_ns is the delay between the exposure of the first used scanline and the last used scanline. They can be used used to interpolate per-scanline metadata.
		
		The pts of the corresponding frame should be within pts_of_this_metadata and pts_of_this_metadata + pixel_exposure_time_ns + rolling_shutter_skew_time_ns

		When this information is not saved, video recorder should make the best effort to adjust the pts of the video frame to be at the center of the frame exposure.

case 2:
  float gyro[3];
break;
	radians/s	Gyroscope signal in rad/s around XYZ axes of the camera. Rotation is positive in the counterclockwise direction.
		
		Note that initial gyro reading are in the IMU coordinate system defined by its driver, and proper transform is required to convert it to camera coordinate system.
		
		Refer to Android Sensor.TYPE_GYROSCOPE

case 3:
  float acc[3];
break;
	m/s^2	Accelerometer reading in m/s^2 along XYZ axes of the camera.
		
		Note that initial accelerometer reading are in the IMU coordinate system defined by its driver, and proper transform is required to convert it to camera coordinate system.
		
		Refer to Android Sensor.TYPE_ACCELEROMETER

case 4:
  float position[3];
break;
		3D position of the camera. 3D position and angle axis rotation together defines the 6Dof pose of the camera, and they are in a common application-defined coordinate system.
		
		This information can be obtained by running 6Dof tracking on the camera device.

case 5
  float latitude;
  float longitude;
  float altitude;
break;
	degrees	Minimal GPS coordinate.

case 6:		// extended gps
double time_gps_epoch;	s	Time since GPS epoch when measurement was taken
int gps_fix_type;		0 (no fix), 2 (2D fix), 3 (3D fix)
double latitude;	degrees	Latitude in degrees
double longitude;	degrees	Longitude in degrees
float altitude;	m	Height above the WGS-84 ellipsoid

float horizontal_accuracy;	m	Horizontal (lat/long) accuracy
float vertical_accuracy;	m	Vertical (altitude) accuracy
float velocity_east;	m/s	Velocity in the east direction
float velocity_north;	m/s	Velocity in the north direction
float velocity_up;	m/s	Velocity in the up direction
float speed_accuracy;	m/s	Speed accuracy
break;		

case 7:
float field[3];
break;
	micro tesla	Ambient magnetic field .
		
		Refer to Android Sensor.TYPE_MAGNETIC_FIELD

}		

Notes:

- Presentation time (pts) of the data samples and video frames are the capture timestamps from a common clock.
- The coordinate systems are right-hand sided. The camera coordinate system is defined as X pointing right, Y pointing downward, and Z pointing forward. The Y-axis of the global coordinate system should point down along the gravity vector.
- IMU readings are typically in its own IMU coordinate system, and necessary rotation is needed to map them to the camera coordinate system if the two coordinate systems are different.
- All fields are little-endian and least significant bit first, and the 32-bit floating points are of IEEE 754-1985 format. The video recorder just need to maintain a struct of these fields in memory and copy the raw data to video packets.
- To accurately synchronize the video frame and the metadata, the pts of the video frame should be at the center of its exposure (can also be inferred from exposure metadata).
- Encoder need to choose a large time-scale in order to get accurate pts. Alternatively, a diff time can be added for finer timing.

Issues

- Only allows one-packet-per-data-sample. Embedded device may have issues writing very high frequency packets.
    - More IO pressure.
    - Increases file header size e.g. stsc and stco, and stsz (if varying packet size)
- It might be problematic to mix different kind of data with different delay, which causes the pts going back and forth (to be investigated). This however can be worked around by buffering the metadata, and write them in monotonic order.