Camera Control by BO-IC400 at Android Application Development

The BT-35E has the camera for shooting the front of the wearer. BO-IC400 can get BT-35E camera data (picture, video) by Android Camera API or Moverio SDK. In addition, the camera data can be applied to various uses (e.g. marker recognition, etc).
This document is described how to control BT-35E camera by BO-IC400.
Note:This document is for OS version 1.1.0 of BO-IC400.

Please refer to following information for BT-35E camera control.

- The method of BT-35E camera control by BO-IC400
- The control possible items by Android API / Moverio SDK
- The specification of BT-35E camera control by Android CameraAPI of BO-IC400

The method of BT-35E camera control by BO-IC400

BO-IC400 can control BT-35E camera by using Android CameraAPI or Moverio SDK. Please refer to "The Camera control page on MoverioBasicFunctionSDK_DevelopersGuide_EN_1.1.0.pdf" for the camera control by Moverio SDK. However, the single application can't use Android CameraAPI and Movrerio SDK at the same time.

There are CameraAPI1 and CameraAPI2 on Android CameraAPI. Please refer to following information for the usages of each API.

- Android CameraAPI1(android.hardware.Camera)

  https://developer.android.com/training/camera

  *Note:CameraAPI1 is deprecated

- Android CameraAPI2(android.hardware.camera2)

  https://developer.android.com/training/camera2

  https://medium.com/google-developers/detecting-camera-features-with-camera2-61675bb7d1bf#.2x3icoqnc

- The sample source code of Android Camera API2

The following information is sample code of camera application by CameraAPI2. Install to BO-IC400 and launch the application with BT-35E connected, BT-35E camera can be used. Please refer to this for implementing application.

https://github.com/googlearchive/android-Camera2Basic?utm_campaign=adp_series_cameragithub_030916&utm_source=gdev&utm_medium=yt-desc

The control possible items by Android API / Moverio SDK

It is possible to control the BT-35E camera by Android CameraAPI2 of BO-IC400. However, it is impossible to change some settings from Android CameraAPI2. Please use Moverio SDK if want to change these settings. The spec of BT-35E camera and controllability of the settings by each API / SDK are described in "The correspondence table of setting controllability by Android CameraAPI2 / Moverio SDK".

The correspondence table of setting controllability by Android CameraAPI2 / Moverio SDK

Settings	BT-35E	Android CameraAPI	Moverio SDK
Resolution	640x480	●	●
	1280x720	●	●
	1920x1080	●	●
	2592x1944	●	●
Exposure compensation	auto/manual	auto only	●
The step of manual exposure compensation	-5 ~ +5		●
Focus control	pan		
Sharpness	0 ~ +128		●
White balance	auto/cloudy_daylight(6000K)/daylight(5500K)/fluorescent(4200K)/incandescent(3200K)/twilight(3500K)	auto only	●
Power frequency	50Hz/60Hz		●

The specification of BT-35E camera control by Android CameraAPI of BO-IC400

The specification of some Android CameraAPI of BO-IC400 are different with standard Android CameraAPI. This topic explains about this specification on CameraAPI2.

■About using BT-35E camera

The code snippet for getting camera characteristics

```
CameraManager manager =
     (CameraManager)getSystemService(CAMERA_SERVICE);

try {
  for (String cameraId : manager.getCameraIdList()) {
    CameraCharacteristics chars
       = manager.getCameraCharacteristics(cameraId);
    // Do something with the characteristics

} catch (CameraAccessException e) {
  e.printStackTrace();
}
```

“The code snippet for getting camera characteristics” is that specify the cameraId and get camera characteristics. BO-IC400 has two USB port (Bottom port, Side port) and it is possible to control the BT-35E camera and USB camera by connecting these. However, BT-35E is supported on Bottom port only. When BT-35E is connected to Bottom port, the cameraId is assigned such as ”The cameraId at connecting BT-35E”.

The cameraId at connecting BT-35E

Camera API	BO-IC400 Back Camera	Bottom port	Side port *
CameraAPI1	1	0	2
CameraAPI2	/dev/video3	0	/dev/video4

* In case of connecting USB camera to Side port

BO-IC400 gets camera characteristics from CameraCharacteristics and can select intended camera.

The code snippet for getting camera lends direction

```
CameraCharacteristics chars
= manager.getCameraCharacteristics(cameraId);
Integer facing = chars.get(CameraCharacteristics.LENS_FACING);
```

“The code snippet for getting camera lens facing” is that get the lens facing from camera characteristics. When BT-35E is connected to Bottom port, the lens facing is assigned such as ”The lens facing at connecting BT-35E”.

The lens facing at connecting BT-35E

Camera API	BO-IC400 Back Camera	Bottom port	Side port *
CameraAPI1	FRONT	BACK	FRONT
CameraAPI2	EXTERNAL	BACK	EXTERNAL

* In case of connecting USB camera to Side port

“The code snippet for using BT-35E camera” is that code snippet in case of using BT-35E camera.

The code snippet for using BT-35E camera

```
CameraManager manager =
     (CameraManager)getSystemService(CAMERA_SERVICE);

try {
  for (String cameraId : manager.getCameraIdList()) {
    CameraCharacteristics chars
      = manager.getCameraCharacteristics(cameraId);
    Integer facing = chars.get(CameraCharacteristics.LENS_FACING);
  if (facing != null && facing ==
      CameraCharacteristics.LENS_FACING_BACK) {
    // Open BT-35E camera (Please refer to upper link)
  }
} catch (CameraAccessException e) {
  e.printStackTrace();
}
```

■About using USB camera

When USB camera is connected to USB port (Bottom or Side), the cameraId is assigned such as ”The cameraId at connecting USB camera”.

The cameraId at connecting USB camera

Camera API	BO-IC400 Back Camera	USB Bottom Port *	USB Side Port *
CameraAPI1	0	1 or 2	1 or 2
CameraAPI2	0	/dev/video3 or 4	/dev/video3 or 4

* cameraId can be assigned a lower number Id from the first recognized device

The lens facing is assigned such as ”The lens facing at connecting USB camera”.

The lens facing at connecting USB camera

Camera API	BO-IC400 Back Camera	USB Bottom Port	USB Side Port
CameraAPI1	BACK	FRONT	FRONT
CameraAPI2	BACK	EXTERNAL	EXTERNAL

If BO-IC400 use USB camera, please specify the parameters of ”The cameraId at connecting USB camera” and ”The lens facing at connecting USB camera” to the source code of ”The code snippet for using BT-35E camera”.

■About the screen rotation process

At Android CameraAPI2 sample source code, the screen rotation process depending on the BO-IC400 rotate state is executed in order to do appropriate camera preview.

However, when the same process is executed using BT-35E camera, the BO-IC400 rotate state and BT-35E rotate state don’t synchronize. As a result, the mismatch of camera preview is occurred. Therefore, when BO-IC400 use BT-35E camera, recommend to not include the screen rotation process such as ”The souce code of not including screen rotation process” (Comment out “matrix.postRoatate()”).

The souce code of not including screen roatation process

```
if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 ==
rotation) {
    bufferRect.offset(centerX - bufferRect.centerX(), centerY -
bufferRect.centerY());
    matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
    float scale = Math.max(
        (float) viewHeight / mPreviewSize.getHeight(),
        (float) viewWidth / mPreviewSize.getWidth());
    matrix.postScale(scale, scale, centerX, centerY);
    // matrix.postRotate(90 * (rotation - 2), centerX, centerY);
}
```

In addition, the return values of AndroidAPI regarding screen rotation are fixed value on Trackpad mode of BO-IC400 such as ”The AndroidAPI regarding screen rotation”.

The AndroidAPI regarding screen rotation

Android API	Retuen value
Display#getRotation()	ROTATION_0
Configuration#orientation	ORIENTATION_LANDSCAPE

■About the aspect ratio of preview

The mounting angle differs between the camera mounted on BO-IC400 and the camera mounted on BT-35E. Please refer to the table below for details.

The return value of CameraCharacteristics.get(CameraCharacteristics.SENSOR_ORIENTATION)

Type of Camera	Return value
BO-IC400 Back Camera	90
BT-35E Camera	0

Therefore, if use BT-35E camera and preview the image assuming 90-degree mounting angle, the preview screen may be distorted. For example, a square object will appear as a rectangle. It is recommended that pay attention to the mounting angle and preview the image appropriately for the resolution of the acquired image.

The Android CameraAPI2 sample source code assumes a 90-degree mounting angle for previewing. With the following modification such as “The code of changing appropriate aspect ratio”, the BT-35E camera can preview with the proper aspect ratio.

The code of changing appropriate aspect ratio

```
if(mSensorOrientation != 0){
      // When the mounting angle is not 0-degree, it is BO-IC400 back
camera, so it should be the same as the original source code.
      if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
            mTextureView.setAspectRatio(
                  mPreviewSize.getWidth(),mPreviewSize.getHeight());
      } else {
            mTextureView.setAspectRatio(
                  mPreviewSize.getHeight(), mPreviewSize.getWidth());
      }
} else {
      // When the mounting angle is 0-degree, it is determined to be
BT-35E camera (USB camera).
      //BT-35E camera doesn’t rotate regardless of the device state,
so the aspect ratio is always assumed to be horizontal preview.
      mTextureView.setAspectRatio(
                  mPreviewSize.getWidth(), mPreviewSize.getHeight());
}
```

```
if(mSensorOrientation !=0) {
   // For BT-35E camera, the following process is not necessary
because BT-35E camera doesn’t rotate regardless of the device state.
   if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 ==
rotation) {
      bufferRect.offset(centerX - bufferRect.centerX(), centerY -
bufferRect.centerY());
      matrix.setRectToRect(viewRect, bufferRect,
Matrix.ScaleToFit.FILL);
      float scale = Math.max(
            (float) viewHeight / mPreviewSize.getHeight(),
            (float) viewWidth / mPreviewSize.getWidth());
      matrix.postScale(scale, scale, centerX, centerY);
      matrix.postRotate(90 * (rotation - 2), centerX, centerY);
   } else if (Surface.ROTATION_180 == rotation) {
      matrix.postRotate(180, centerX, centerY);
   }
}
```