**### Multimedia**

**#### [Overview](/application/native/guides/multimedia/overview.md)**

**#### [Media Content](/application/native/guides/multimedia/media-content.md)**

**#### [Metadata](/application/native/guides/multimedia/metadata.md)**

**#### [Image Editing](/application/native/guides/multimedia/image-edit.md)**

**#### [Thumbnail Images](/application/native/guides/multimedia/thumbnail-images.md)**

**#### Visual Detection and Recognition**

**##### [Overview](/application/native/guides/multimedia/media-vision.md)**

|  |
| --- |
| # Pose Detection |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Pose Detection is new feature of Media Vision Inference API since Tizen 6.0. It can provides landmarks detection. In addition, it defines landmarks and parts of a human body and provides to detect a human pose with an motion capture (MoCap) file which a user can create or edit with tools.  Pose Detection is a new feature of Media Vision Inference API since Tizen 6.0. It provides landmark detection. Besides, it defines landmarks and parts of a human body to help detect a human pose with motion capture (MoCap) file, which you can create or edit with tools. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## [Background](#background) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Tizen defines a human body pose landmarks and body parts as shown in below figure. |
|  |

|  |  |
| --- | --- |
|  | Tizen defines the human body pose landmarks and body parts, as follows: |
|  |

|  |
| --- |
| \*\*Figure: Definition of human body pose landmarks and body parts\*\* |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ![Body pose](./media/mediavision\_pose\_tizen\_def.png) |
|  |

|  |
| --- |
| Pose landmark detection models are available in open model zoo such as [hosted model zoo](https://www.tensorflow.org/lite/guide/hosted\_models#floating\_point\_models) or public github site such as [public pose model](https://github.com/tyoungroy/PoseEstimationForMobile). Those models provides different landmarks information such as the number of landmarks and locations. To use them correctly, you have to map those information to landmarks based on the definition. For example, to use the model [public pose model](https://github.com/tyoungroy/PoseEstimationForMobile) which provides 14 landmarks as,  Pose landmark detection models are available in open model zoo such as [hosted model zoo](https://www.tensorflow.org/lite/guide/hosted\_models#floating\_point\_models) or public GitHub site such as [public pose model](https://github.com/tyoungroy/PoseEstimationForMobile). The public pose models provide landmark information, such as the number of landmarks and locations. To use them correctly, you must map the information to landmarks based on the definition.  For example, you can use the [public pose model](https://github.com/tyoungroy/PoseEstimationForMobile), which provides 14 landmarks as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ![Body pose](./media/mediavision\_pose\_public\_model\_def.png), |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| while `-1` means that there is no landmarks, you can create a mapping file. Suppose that the file is `pose\_mapping.txt`, then, you can create `pose\_mapping.txt` as   * `-1` denotes that there are no landmarks. You can create a mapping file. Suppose you create a file with the name `pose\_mapping.txt` then, you can populate the `pose\_mapping.txt` file as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
| 1 |
|  |

|  |
| --- |
| 2 |
|  |

|  |
| --- |
| -1 |
|  |

|  |
| --- |
| 3 |
|  |

|  |
| --- |
| 4 |
|  |

|  |
| --- |
| 5 |
|  |

|  |
| --- |
| 6 |
|  |

|  |
| --- |
| 7 |
|  |

|  |
| --- |
| 8 |
|  |

|  |
| --- |
| -1 |
|  |

|  |
| --- |
| 9 |
|  |

|  |
| --- |
| 10 |
|  |

|  |
| --- |
| 11 |
|  |

|  |
| --- |
| 12 |
|  |

|  |
| --- |
| 13 |
|  |

|  |
| --- |
| 14 |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| `1` denotes that the first landmark of the model corresponds to `MV\_INFERENCE\_HUMAN\_POSE\_HEAD`. `-1` at the 3rd denotes that there is no corresponding landmark `MV\_INFERENCE\_HUMAN\_POSE\_THORAX`. `3` at the 4th denotes that the 3rd landmark of the model corresponds to `MV\_INFERENCE\_HUMAN\_POSE\_RIGHT\_SHOULDER`. You can know how it works with following table.   * `1` denotes that the first landmark of the model that corresponds to `MV\_INFERENCE\_HUMAN\_POSE\_HEAD.` * `-1` at the 3rd noded denotes that there is no corresponding landmark `MV\_INFERENCE\_HUMAN\_POSE\_THORAX.` * `3` at the 4th nodes denote that the 3rd landmark of the model corresponds to `MV\_INFERENCE\_HUMAN\_POSE\_RIGHT\_SHOULDER.`   The following table shows how the public pose model works with different values, definitions, and mappings: |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
| \*\*Table: Example of how [public pose model](https://github.com/tyoungroy/PoseEstimationForMobile) map the definition\*\*  \*\*Table: Example of how [public pose model](<https://github.com/tyoungroy/PoseEstimationForMobile>) maps to the definition\*\* |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| | Value | Defintion | pose\_mapping.txt | |
|  |

|  |
| --- |
| | - | - | - | |
|  |

|  |
| --- |
| | 1 | MV\_INFERENCE\_HUMAN\_POSE\_HEAD | 1 | |
|  |

|  |
| --- |
| | 2 | MV\_INFERENCE\_HUMAN\_POSE\_NECK | 2 | |
|  |

|  |
| --- |
| | 3 | MV\_INFERENCE\_HUMAN\_POSE\_THORAX | -1 | |
|  |

|  |
| --- |
| | 4 | MV\_INFERENCE\_HUMAN\_POSE\_RIGHT\_SHOULDER | 3 | |
|  |

|  |
| --- |
| | 5 | MV\_INFERENCE\_HUMAN\_POSE\_RIGHT\_ELBOW | 4 | |
|  |

|  |
| --- |
| | 6 | MV\_INFERENCE\_HUMAN\_POSE\_RIGHT\_WRIST | 5 | |
|  |

|  |
| --- |
| | 7 | MV\_INFERENCE\_HUMAN\_POSE\_LEFT\_SHOULDER | 6 | |
|  |

|  |
| --- |
| | 8 | MV\_INFERENCE\_HUMAN\_POSE\_LEFT\_ELBOW | 7 | |
|  |

|  |
| --- |
| | 9 | MV\_INFERENCE\_HUMAN\_POSE\_LEFT\_WRIST | 8 | |
|  |

|  |
| --- |
| | 10 | MV\_INFERENCE\_HUMAN\_POSE\_PELVIS | -1 | |
|  |

|  |
| --- |
| | 11 | MV\_INFERENCE\_HUMAN\_POSE\_RIGHT\_HIP | 9 | |
|  |

|  |
| --- |
| | 12 | MV\_INFERENCE\_HUMAN\_POSE\_RIGHT\_KNEE | 10 | |
|  |

|  |
| --- |
| | 13 | MV\_INFERENCE\_HUMAN\_POSE\_RIGHT\_ANKLE | 11 | |
|  |

|  |
| --- |
| | 14 | MV\_INFERENCE\_HUMAN\_POSE\_LEFT\_HIP | 12 | |
|  |

|  |
| --- |
| | 15 | MV\_INFERENCE\_HUMAN\_POSE\_LEFT\_KNEE | 13 | |
|  |

|  |
| --- |
| | 16 | MV\_INFERENCE\_HUMAN\_POSE\_LEFT\_ANKLE | 14 | |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| --- |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Motion capture file (MoCap) includes the movements of objects or people. There are various MoCap formats, but well-known BVH (BioVision Hierarchy) file is supported in Media Vision. BVH file has Hierarchy structure to provide landmark information and the structure can be changed. It means that landmarks information is differenct from the definition. To use them correctly, you have to map those information to landmarks based on the definition. For example, an [example](file://./media/mediavision\_pose\_bvh\_sample.bvh) BVH file describes a squat pose as,  The motion capture file (MoCap) includes the movements of objects or people. There are various MoCap formats, but a well-known BVH (BioVision Hierarchy) file is supported in Media Vision. BVH file has a hierarchy structure to provide landmark information, and the structure can be changed. It means that landmarks information is different from the definition. To use them correctly, you have to map the information to the landmarks based on the definition. For example, a [BVH file](file://./media/mediavision\_pose\_bvh\_sample.bvh) describes a squat pose as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ![Body pose](./media/mediavision\_pose\_bvh\_sample.png) |
|  |

|  |
| --- |
| , with 15 landmarks. It starts with Hips and ends with RightRoot. Thus, you can create a mapping file, named of `mocap\_mapping.txt` as  With 15 landmarks, the example starts with Hips and ends with RightRoot. You can create a mapping file named `mocap\_mapping.txt` as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
| Hips,10 |
|  |

|  |
| --- |
| Neck,2 |
|  |

|  |
| --- |
| Head,1 |
|  |

|  |
| --- |
| LeftUpArm,7 |
|  |

|  |
| --- |
| LeftLowArm,8 |
|  |

|  |
| --- |
| LeftHand,9 |
|  |

|  |
| --- |
| RightUpArm,4 |
|  |

|  |
| --- |
| RightLowArm,5 |
|  |

|  |
| --- |
| RightHand,6 |
|  |

|  |
| --- |
| LeftUpLeg,14 |
|  |

|  |
| --- |
| LeftLowLeg,15 |
|  |

|  |
| --- |
| LeftFoot,16 |
|  |

|  |
| --- |
| RightUpLeg,11 |
|  |

|  |
| --- |
| RightLowLeg,12 |
|  |

|  |
| --- |
| RightFoot,13 |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Prerequisites |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| To enable your application to use the media vision inference functionality: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 1. To use the functions and data types of the Media Vision Inference API (in [mobile](../../api/mobile/latest/group\_\_CAPI\_\_MEDIA\_\_VISION\_\_INFERENCE\_\_MODULE.html) and [wearable](../../api/wearable/latest/group\_\_CAPI\_\_MEDIA\_\_VISION\_\_INFERENCE\_\_MODULE.html) applications), include the `<mv\_inference.h>` header file in your application. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| In addition, you must include the `<image\_util.h>` header file to handle the image decoding tasks, or the `<camera.h>` header file to provide the preview images: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| #include <mv\_inference.h> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* Image decoding for image recognition \*/ |
|  |

|  |
| --- |
| #include <image\_util.h> |
|  |

|  |
| --- |
| /\* Preview images for image tracking \*/ |
|  |

|  |
| --- |
| #include <camera.h> |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
| 2. Create a structure to store the global data. |
| 2. Create a structure to store global data. |

|  |
| --- |
|  |
|  |

|  |
| --- |
| For pose detection , use the following `imagedata\_s` structure: |
| For pose detection, use the following `imagedata\_s` structure as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| struct \_imagedata\_s { |
|  |

|  |
| --- |
| mv\_source\_h g\_source; |
|  |

|  |
| --- |
| mv\_engine\_config\_h g\_engine\_config; |
|  |

|  |
| --- |
| mv\_inference\_h g\_inferece; |
|  |

|  |
| --- |
| mv\_pose\_h g\_pose; |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
| typedef struct \_imagedata\_s imagedata\_s; |
|  |

|  |
| --- |
| static imagedata\_s imagedata; |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Detect Human Pose |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| To detect human pose from an image: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 1. Create the source and engine configuration handles: |
| 1. Create the source and engine configuration handles as follows: |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| int error\_code = 0; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_create\_source(&imagedata.g\_source); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_create\_engine\_config(&imagedata.g\_engine\_config); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code= %d", error\_code); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
| 2. Decode the image file and fill the `g\_source` handle with the decoded raw data. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| In the following example, `sample.jpg` is the image with a person gets a squat pose and is in the `<OwnDataPath>` folder. The `<OwnDataPath>` refers to your own data path:  In the following example, a person is shown in a squat pose, and the image is in the `<OwnDataPath>` folder. The `<OwnDataPath>` refers to your data path: |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| /\* For details, see the Image Util API Reference \*/ |
|  |

|  |
| --- |
| unsigned char \*dataBuffer = NULL; |
|  |

|  |
| --- |
| unsigned long long bufferSize = 0; |
|  |

|  |
| --- |
| unsigned long width = 0; |
|  |

|  |
| --- |
| unsigned long height = 0; |
|  |

|  |
| --- |
| image\_util\_decode\_h imageDecoder = NULL; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = image\_util\_decode\_create(&imageDecoder); |
|  |

|  |
| --- |
| if (error\_code != IMAGE\_UTIL\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = image\_util\_decode\_set\_input\_path(imageDecoder, "/<OwnDataPath>/sample.jpg"); |
|  |

|  |
| --- |
| if (error\_code != IMAGE\_UTIL\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = image\_util\_decode\_set\_colorspace(imageDecoder, IMAGE\_UTIL\_COLORSPACE\_RGB888); |
|  |

|  |
| --- |
| if (error\_code != IMAGE\_UTIL\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = image\_util\_decode\_set\_output\_buffer(imageDecoder, &dataBuffer); |
|  |

|  |
| --- |
| if (error\_code != IMAGE\_UTIL\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = image\_util\_decode\_run(imageDecoder, &width, &height, &bufferSize); |
|  |

|  |
| --- |
| if (error\_code != IMAGE\_UTIL\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = image\_util\_decode\_destroy(imageDecoder); |
|  |

|  |
| --- |
| if (error\_code != IMAGE\_UTIL\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* Fill the dataBuffer to g\_source \*/ |
|  |

|  |
| --- |
| error\_code = mv\_source\_fill\_by\_buffer(imagedata.g\_source, dataBuffer, (unsigned int)bufferSize, |
|  |

|  |
| --- |
| (unsigned int)width, (unsigned int)height, MEDIA\_VISION\_COLORSPACE\_RGB888); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| free(dataBuffer); |
|  |
|  |

|  |
| --- |
| dataBuffer = NULL; |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
| 3. To detect landmark of pose from the `sample.jpg`, |
| 3. To detect landmark of the pose from the `sample.jpg,` create a `g\_inference` media vision inference handle as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| <img src="./media/mediavision\_pose\_sample\_sumo.png" width=300>, |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| error\_code = mv\_inference\_create(&imagedata.g\_inference); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
| 4. Configure `g\_engine\_config` with a body pose model data and its mapping file. In the following example, TensorFlow Lite model is used and `data.tflite` and its mapping file `data\_mapping.txt` are in `<OwnDataPath>`. Model data is available in open model zoo such as [hosted model zoo](https://www.tensorflow.org/lite/guide/hosted\_models#floating\_point\_models) or [public pose model](https://github.com/tyoungroy/PoseEstimationForMobile). Suppose model data and its mapping file in Section <a name="background">Background</a>:  4. Configure `g\_engine\_config` with a body pose model data and its mapping file. In the following example, TensorFlow Lite model is used and `data.tflite` and its mapping file `data\_mapping.txt` are stored in your `<OwnDataPath>`.  The model data is available in open model zoo such as [hosted model zoo](https://www.tensorflow.org/lite/guide/hosted\_models#floating\_point\_models) or the [public pose model](https://github.com/tyoungroy/PoseEstimationForMobile). Suppose model data and its mapping file in Section <a name="background">Background</a>: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| #define MODEL\_DATA "OwnDataPath/data.tflite" |
|  |

|  |
| --- |
| #define MODEL\_MAPPING\_FILE "OwnDataPath/data\_mapping.txt" |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| char \*inputNodeName = "image"; |
|  |

|  |
| --- |
| char \*outputNodeName[] = { "output" }; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_string\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_MODEL\_WEIGHT\_FILE\_PATH, |
|  |

|  |
| --- |
| MODEL\_DATA); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_string\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_INPUT\_DATA\_TYPE, |
|  |

|  |
| --- |
| MV\_INFERENCE\_DATA\_FLOAT32); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_string\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_MODEL\_USER\_FILE\_PATH, |
|  |

|  |
| --- |
| MODEL\_MAPPING\_FILE); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_double\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_MODEL\_MEAN\_VALUE, |
|  |

|  |
| --- |
| 0.0); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_double\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_MODEL\_STD\_VALUE, |
|  |

|  |
| --- |
| 1.0); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_int\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_BACKEND\_TYPE, |
|  |

|  |
| --- |
| MV\_INFERENCE\_BACKEND\_TFLITE); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_int\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_BACKEND\_TYPE, |
|  |

|  |
| --- |
| MV\_INFERENCE\_TARGET\_CPU); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_int\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_INPUT\_TENSOR\_WIDTH, |
|  |

|  |
| --- |
| 192); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_int\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_INPUT\_TENSOR\_HEIGHT, |
|  |

|  |
| --- |
| 192); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_int\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_INPUT\_TENSOR\_CHANNELS, |
|  |

|  |
| --- |
| 3); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_string\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_INPUT\_NODE\_NAME, |
|  |

|  |
| --- |
| inputNodeName); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_engine\_config\_set\_array\_string\_attribute(handle, |
|  |

|  |
| --- |
| MV\_INFERENCE\_OUTPUT\_NODE\_NAMES, |
|  |

|  |
| --- |
| outputNodeName, 1); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| For more information on the configuration attributes such as `MV\_INFERENCE\_MODEL\_WEIGHT\_FILE\_PATH`, see Media Vision Inference API (in [mobile](../../api/mobile/latest/group\_\_CAPI\_\_MEDIA\_\_VISION\_\_INFERENCE\_\_MODULE.html) and [wearable](../../api/wearable/latest/group\_\_CAPI\_\_MEDIA\_\_VISION\_\_INFERENCE\_\_MODULE.html) applications). |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 5. Use `mv\_inference\_configure()` to configure`g\_inference` inference handle with`g\_engine\_config`:  5.Use `mv\_inference\_configure()` to configure`g\_inference` inference handle with`g\_engine\_config`as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| error\_code = mv\_inference\_configure(imagedata.g\_inference, imagedata.g\_engine\_config); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 6. Use`mv\_inference\_prepare()` to prepare inference: |
| 6. Use`mv\_inference\_prepare()` to prepare inference as follows: |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| error\_code = mv\_inference\_prepare(imagedata.g\_inference); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 7. To detect a pose, create `g\_pose` media vision pose handle: |
| 7. To detect a pose, create `g\_pose` media vision pose handle as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_pose\_create(&imagedata.g\_pose); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 8. Set`mocap.bvh` and its mapping file`mocap\_mapping.txt` to`g\_pose` handle to compare and detect pose. Suppose MoCap file and its mapping file in Section <a name="background">Background</a>: |
|  |

|  |  |
| --- | --- |
|  | 8. Set`mocap.bvh` and its mapping file`mocap\_mapping.txt` to`g\_pose` handle to compare and detect the pose. Suppose MoCap file and its mapping file in Section <a name="background">Background</a>: |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| #define MOCAP\_DATA "OwnDataPath/mocap.bvh" |
|  |

|  |
| --- |
| #define MOCAP\_MAPPING\_FILE "OwnDataPath/mocap\_mapping.txt" |
|  |

|  |
| --- |
| error\_code = mv\_pose\_set\_from\_file(imagedata.g\_pose, MOCAP\_DATA, MOCAP\_MAPPING\_FILE); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 9. Use`mv\_inference\_pose\_landmark\_detect()` to detect landmarks from the image: |
| 9. Use`mv\_inference\_pose\_landmark\_detect()` to detect landmarks from the image as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| error\_code = mv\_inference\_pose\_landmark\_detect(imagedata.g\_source, &imagedata.g\_inference, NULL, \_on\_pose\_landmark\_detected\_cb, NULL); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| `mv\_inference\_pose\_landmark\_detect()` invokes `\_on\_pose\_landmark\_detected\_cb()` callback.  `mv\_inference\_pose\_landmark\_detect()` invokes the `\_on\_pose\_landmark\_detected\_cb()` callback. |
|  |

|  |
| --- |
| The following callback example shows how to detect pose from the detected landmark: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| static void |
|  |

|  |
| --- |
| \_on\_pose\_landmark\_detected\_cb(mv\_source\_h source, mv\_inference\_pose\_result\_h pose, void \*user\_data) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| float pose\_score = 0.0f; |
|  |

|  |
| --- |
| error\_code = mv\_pose\_compare(imagedata.g\_pose, pose, |
|  |

|  |
| --- |
| (MV\_INFERENCE\_HUMAN\_BODY\_PART\_LEG\_LEFT | |
|  |

|  |
| --- |
| MV\_INFERENCE\_HUMAN\_BODY\_PART\_LEG\_RIGHT), |
|  |

|  |
| --- |
| &pose\_score); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) { |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
| return; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if (pose\_score > 0.9f) { |
|  |

|  |
| --- |
| dlog\_print(DLOG\_INFO, LOG\_TAG, "success to detect pose with score %.3f\n", pose\_score); |
|  |

|  |
| --- |
| } else { |
|  |

|  |
| --- |
| dlog\_print(DLOG\_INFO, LOG\_TAG, "fail to detect pose with score %.3f\n", pose\_score); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 10. After the detetion is complete, destroy the source, engine configuration, the inference, and the pose handles using `mv\_destroy\_source()`, `mv\_destroy\_engine\_config()`, `mv\_inference\_destroy()`, and `mv\_pose\_destroy()`:  10. After the detection is complete, destroy the source, engine configuration, the inference, and the pose handles using `mv\_destroy\_source()`, `mv\_destroy\_engine\_config()`, `mv\_inference\_destroy()`, and `mv\_pose\_destroy()` as follows: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ```c |
|  |

|  |
| --- |
| error\_code = mv\_destroy\_source(imagedata.g\_source); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_destroy\_engine\_config(imagedata.g\_engine\_config); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = mv\_inference\_destroy(imagedata.g\_inference); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| error\_code = |
|  |

|  |
| --- |
| mv\_pose\_destroy(imagedata.g\_pose); |
|  |

|  |
| --- |
| if (error\_code != MEDIA\_VISION\_ERROR\_NONE) |
|  |

|  |
| --- |
| dlog\_print(DLOG\_ERROR, LOG\_TAG, "error code = %d", error\_code); |
|  |

|  |
| --- |
| ``` |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Related Information |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| - Dependencies |
|  |

|  |
| --- |
| - Tizen 6.0 and Higher for Mobile |
|  |

- Tizen 6.0 and Higher for Wearable