ObjectDetectionParameters

Structure containing a set of parameters for the object detection module. More...

Functions

ObjectDetectionParameters (bool enable_tracking_=true, bool enable_segmentation_=false, OBJECT_DETECTION_MODEL **detection_model**=OBJECT_DETECTION_MODEL::MULTI_CLASS_BOX_FAST, float max_range_=-1.f, **BatchParameters** batch_trajectories_parameters=**BatchParameters**(), **OBJECT_FILTERING_MODE** filtering_mode_=**OBJECT_FILTERING_MODE::NMS3D**, float **prediction_timeout_s**=0.2f, bool **allow_reduced_precision_inference**=false, unsigned int instance_id=0, const **sl::String** &**fused_objects_group_name**="", const **sl::String** &**custom_onnx_file**="", const **sl::Resolution** &**custom_onnx_dynamic_input_shape**=**sl::Resolution**(512, 512))

Default constructor. More...

---

bool **save** (**String** filename, **SERIALIZATION_FORMAT** format=**SERIALIZATION_FORMAT::JSON**) const

Saves the current set of parameters into a file to be reloaded with the **load()** method. More...

---

bool **load** (**String** filename, **SERIALIZATION_FORMAT** format=**SERIALIZATION_FORMAT::JSON**)

Loads a set of parameters from the values contained in a previously **saved** file. More...

---

bool **encode** (**String** &serialized_content, **SERIALIZATION_FORMAT** format=**SERIALIZATION_FORMAT::JSON**) const

Generate a JSON Object (with the struct type as a key) containing the serialized struct, converted into a string. More...

---

bool	**decode** (const **String** &serialized_content, **SERIALIZATION_FORMAT** format=**SERIALIZATION_FORMAT::JSON**) Fill the structure from the serialized json object contained in the input string. More...
bool	**operator==** (const **ObjectDetectionParameters** &param1) const
bool	**operator!=** (const **ObjectDetectionParameters** &param1) const

Attributes

unsigned int	**instance_module_id** = 0 Id of the module instance. More...
bool	**enable_tracking** = true Whether the object detection system includes object tracking capabilities across a sequence of images. More...
bool	**enable_segmentation** = false Whether the object masks will be computed. More...
OBJECT_DETECTION_MODEL	**detection_model** = OBJECT_DETECTION_MODEL::MULTI_CLASS_BOX_FAST sl::OBJECT_DETECTION_MODEL to use. More...
sl::String	**fused_objects_group_name** In a multi camera setup, specify which group this model belongs to. More...
sl::String	**custom_onnx_file** Path to the YOLO-like onnx file for custom object detection ran in the ZED SDK. More...
sl::Resolution	**custom_onnx_dynamic_input_shape** **Resolution** to the YOLO-like onnx file for custom object detection ran in the ZED SDK. This resolution defines the

	input tensor size for dynamic shape ONNX model only. The batch and channel dimensions are automatically handled, it assumes it's color images like default YOLO models. More...
float	**max_range** = -1.f Upper depth range for detections. More...
BatchParameters	**batch_parameters** Batching system parameters. More...
OBJECT_FILTERING_MODE	**filtering_mode** = OBJECT_FILTERING_MODE::NMS3D Filtering mode that should be applied to raw detections. More...
float	**prediction_timeout_s** Prediction duration of the ZED SDK when an object is not detected anymore before switching its state to sl::OBJECT_TRACKING_STATE::SEARCHING. More...
bool	**allow_reduced_precision_inference** Whether to allow inference to run at a lower precision to improve runtime and memory usage. More...

Detailed Description

Structure containing a set of parameters for the object detection module.

The default constructor sets all parameters to their default settings.

> **Note**
>
> Parameters can be adjusted by the user.

Constructor and Destructor

```
tectionParameters ( bool                        enable_tracking_ = true,
                    bool                        enable_segmentation_ = false
                    OBJECT_DETECTION_MODEL      detection_model = OBJECT_DET
                    float                       max_range_ = -1.f,
                    BatchParameters             batch_trajectories_parameter
                    OBJECT_FILTERING_MODE       filtering_mode_ = OBJECT_FIL
                    float                       prediction_timeout_s = 0.2f,
                    bool                        allow_reduced_precision_infe
                    unsigned int                instance_id = 0,
                    const sl::String &          fused_objects_group_name = "
                    const sl::String &          custom_onnx_file = "",
                    const sl::Resolution &      custom_onnx_dynamic_input_sh
                  )
```

Default constructor.
All the parameters are set to their default values.

Functions

```
bool save ( String                  filename,
            SERIALIZATION_FORMAT    format = SERIALIZATION_FORMAT::JSON
          )                         const
```

Saves the current set of parameters into a file to be reloaded with the **`load()`** method.

Parameters

- **`filename`** : Name of the file which will be created to store the parameters (extension '.yml' will be added if not set).

Returns

- True if the file was successfully saved, otherwise false.

> **Warning**
>
> For security reasons, the file must not already exist.
> In case a file already exists, the method will return false and existing file will not be updated.

```
bool load ( String                  filename,
            SERIALIZATION_FORMAT    format = SERIALIZATION_FORMAT::JSON
          )
```

Loads a set of parameters from the values contained in a previously **saved** file.

Parameters

- **`filename`** : Path to the file from which the parameters will be loaded (extension '.yml' will be added at the end of the filename if not detected).

Returns

- True if the file was successfully loaded, otherwise false.

```
bool
encode    ( String &              serialized_content,
            SERIALIZATION_FORMAT  format = SERIALIZATION_FORMAT::JSON
            )                     const
```

Generate a JSON Object (with the struct type as a key) containing the serialized struct, converted into a string.

Parameters

- **serialized_content** output string containing the JSON Object
- **format** serialization format, default is JSON

Returns

- True if file was successfully saved, otherwise false.

```
bool
decode    ( const String &        serialized_content,
            SERIALIZATION_FORMAT  format = SERIALIZATION_FORMAT::JSON
            )
```

Fill the structure from the serialized json object contained in the input string.

Parameters

- **serialized_content** input string containing the JSON Object
- **format** serialization format, default is JSON

Returns

- True if the decoding was successful, otherwise false.

```
bool operator== ( const ObjectDetectionParameters &  param1 ) const
```

Comparison operator ==

Parameters

- **ObjectDetectionParameters** to compare

Returns

- true if the two struct are identical

```
bool operator!= ( const ObjectDetectionParameters &  param1 ) const
```

Comparison operator !=

Parameters

- **ObjectDetectionParameters** to compare

Returns

- true if the two struct are different

Variables

```
unsigned int instance_module_id = 0
```

Id of the module instance.
This is used to identify which object detection module instance is used.

`bool enable_tracking = true`

Whether the object detection system includes object tracking capabilities across a sequence of images.

`bool enable_segmentation = false`

Whether the object masks will be computed.

`OBJECT_DETECTION_MODEL detection_model = OBJECT_DETECTION_MODEL::MULTI_CLASS_BOX_FAST`

---

sl::OBJECT_DETECTION_MODEL to use.

`sl::String fused_objects_group_name`

In a multi camera setup, specify which group this model belongs to.
In a multi camera setup, multiple cameras can be used to detect objects and multiple detector having similar output layout can see the same object. Therefore, `Fusion` will fuse together the outputs received by multiple detectors only if they are part of the same `fused_objects_group_name`.

> **Note**
>
> This parameter is not used when not using a multi-camera setup and must be set in a multi camera setup.

`sl::String custom_onnx_file`

Path to the YOLO-like onnx file for custom object detection ran in the ZED SDK.
When `detection_model` is OBJECT_DETECTION_MODEL::CUSTOM_YOLOLIKE_BOX_OBJECTS, a onnx model must be passed so that the ZED SDK can optimize it for your GPU and run inference on it.
The resulting optimized model will be saved for re-use in the future.

> **Attention**
>
> - The model must be a YOLO-like model.
> - The caching uses the deserialized `custom_onnx_file` along with your GPU specs to decide whether to use the cached optmized model or to optimize the passed onnx model. If you change the weights of the onnx file and pass the same path, the ZED SDK will detect the difference and optimize the new model.

> **Note**
>
> This parameter is useless when detection_model is not OBJECT_DETECTION_MODEL::CUSTOM_YOLOLIKE_BOX_OBJECTS.

`sl::Resolution custom_onnx_dynamic_input_shape`

Resolution to the YOLO-like onnx file for custom object detection ran in the ZED SDK.
This resolution defines the input tensor size for dynamic shape ONNX model only. The batch and channel dimensions are automatically handled, it assumes it's color images like default YOLO models.

> **Note**
>
> This parameter is only used when detection_model is OBJECT_DETECTION_MODEL::CUSTOM_YOLOLIKE_BOX_OBJECTS and the provided ONNX file is using dynamic shapes.

> **Attention**
>
> - Multiple model only support squared images

\default Squared images 512x512 (input tensor will be 1x3x512x512)

`float max_range = -1.f`

Upper depth range for detections.
Default: -1.f (value set in **`sl::InitParameters.depth_maximum_distance`**)

> **Note**
>
> The value cannot be greater than **`sl::InitParameters.depth_maximum_distance`** and its unit is defined in **`sl::InitParameters.coordinate_units`**.

`BatchParameters batch_parameters`

Batching system parameters.
Batching system (introduced in 3.5) performs short-term re-identification with deep-learning and trajectories filtering.
`sl::BatchParameters.enable` must to be true to use this feature (by default disabled).

`OBJECT_FILTERING_MODE filtering_mode = OBJECT_FILTERING_MODE::NMS3D`

Filtering mode that should be applied to raw detections.
Default: **`sl::OBJECT_FILTERING_MODE::NMS_3D`** (same behavior as previous ZED SDK version)

> **Note**
>
> This parameter is only used in detection model sl::OBJECT_DETECTION_MODEL::MULTI_CLASS_BOX_XXX and sl::OBJECT_DETECTION_MODEL::CUSTOM_BOX_OBJECTS.
> For custom object, it is recommended to use **`sl::OBJECT_FILTERING_MODE::NMS_3D_PER_CLASS`** or **`sl::OBJECT_FILTERING_MODE::NONE`**.
> In this case, you might need to add your own NMS filter before ingesting the boxes into the object detection module.

`float prediction_timeout_s`

Prediction duration of the ZED SDK when an object is not detected anymore before switching its state to **`sl::OBJECT_TRACKING_STATE::SEARCHING`**.
It prevents the jittering of the object state when there is a short misdetection.
The user can define their own prediction time duration.

> **Note**
>
> During this time, the object will have **`sl::OBJECT_TRACKING_STATE::OK`** state even if it is not detected.
> The duration is expressed in seconds.

> **Warning**
>
> **`prediction_timeout_s`** will be clamped to 1 second as the prediction is getting worse with time.
> Setting this parameter to 0 disables the ZED SDK predictions.

`bool allow_reduced_precision_inference`

Whether to allow inference to run at a lower precision to improve runtime and memory usage.
It might increase the initial optimization time and could include downloading calibration data or calibration cache and slightly reduce the accuracy.

> **Note**
>
> The fp16 is automatically enabled if the GPU is compatible and provides a speed up of almost x2 and reduce memory usage by almost half, no precision loss.
> This setting allow int8 precision which can speed up by another x2 factor (compared to fp16, or x4 compared to fp32) and half the fp16 memory usage, however some accuracy could be lost.
> The accuracy loss should not exceed 1-2% on the compatible models.
> The current compatible models are all sl::AI_MODELS::HUMAN_BODY_XXXX.