



VisionLabs
MACHINES CAN SEE

Configuration Guide

written for LUNA SDK Mobile iOS version 5.31.0

Contents

Configuration Guide	3
Configuration file location	3
Settings	4
System settings	4
Descriptor factory settings	5
FaceDetV2 detector settings	6
LNet	7
HeadPoseEstimator settings	8
EyeEstimator settings	9
Medical mask estimator settings	10
Mouth Estimator settings	12
Face Occlusion Estimator Settings	13
Quality estimator settings	14
GlassesEstimator settings	16
LivenessOneShotRGBEstimator settings	17
NIRLivenessEstimator settings	20
Runtime settings	21

Configuration Guide

Configuration options are specified via `faceengine.conf` file which is basically an XML document with special tag formatting. The document itself is not required to exist, in this case FSDK will fall back to some default settings, which, however, may not be suitable for several tasks.

WARNING! By changing any configuration settings from default ones it is assumed that user understands what these settings do and how they will affect performance and output results of their application. The rule of thumb is this: DO NOT change anything in configuration file unless you really have to.

Always remember that incorrect config may huck the things up very badly. Pay attention to what you configure and how. Always double-check what you deploy.

Some configuration settings may be omitted due to their obscurity and research use case only.

The location where the config file is found varies across different systems but tries to be as consistent as possible.

The config file format is optimized for deserialization of several FSDK types:

- Int1 - scalar 32 bit integral numeric type
- Int2 - 2-d 32 bit integral numeric type (aka Vector2i, Size)
- Int3 - 3-d 32 bit integral numeric type
- Int4 - 4-d 32 bit integral numeric type (aka Rect)
- Float1 - scalar 32 bit floating point numeric type
- Float2 - 2-d 32 bit floating point numeric type (aka Vector2f)
- Float3 - 3-d 32 bit floating point numeric type
- Float4 - 4-d 32 bit floating point numeric type
- String - short null-terminated string (max. 16 characters including the null-terminator)

Configuration file location

The location where the config file is found varies across different systems but tries to be as consistent as possible. Path resolution is the following:

Mobile platforms

- Look for “`data/faceengine.conf`” in current working directory.

Settings

System settings

Parameter	Description	Type	Default value
verboseLogging	Level of log verbosity. 1 - Errors, 2 - Warnings, 3 - Info, 4 - Debug.	"Value::Int1"	2

Verbosity level sets the upper limit of what type of messages may be printed out by the Luna SDK. For example, if user set verboseLogging to 3, it means that Errors, Warnings and Info messages will be printed out to the console. Verbose level of 0 indicates that there are no logging messages printed out at all.

Example:

```
<section name="system">  
  <param name="verboseLogging" type="Value::Int1" x="0" />  
</section>
```

Descriptor factory settings

Descriptor factory is a facility that creates descriptor extractors and matchers. Both of them utilize algorithms that require a number of coefficients (“weights”) to operate properly.

Parameter	Description	Type	Default value
model	CNN face descriptor version. Possible values: 60	Value::Int1	60
useMobileNet	MobileNet is faster but less accurate. Possible values: 0 - don't use mobile net version, 1 - use mobile net version.	Value::Int1	1
distance	Distance between descriptors on matching. L1 faster, L2 make better precision. Possible values: L1, L2. Model 60 supports just L2 distance.	Value::String	L2
descriptorCountWarningLevel	Threshold, that limits the ratio of created descriptors to the amount, defined by your license. When the threshold is exceeded, FSDK prints the warning.	Value::Float1	0.9

Example:

```
<section name="DescriptorFactory::Settings">
  <param name="model" type="Value::Int1" x="60" />
  <param name="useMobileNet" type="Value::Int1" x="1" />
  <param name="distance" type="Value::String" text="L2" />
  <param name="descriptorCountWarningLevel" type="Value::Float1" x="0.9" />
</section>
```

FaceDetV2 detector settings

Parameter	Description	Type	Default value
FirstThreshold	1-st threshold in [0..1] range.	"Value::Float1"	0.6
SecondThreshold	2-nd threshold in [0..1] range.	"Value::Float1"	0.7
ThirdThreshold	3-d threshold in [0..1] range.	"Value::Float1"	0.6
minFaceSize	Minimum face size in pixels.	"Value::Int1"	50
strictlyMinSize	Enforced minimum face size.	"Value::Int1"	0
scaleFactor	Image scale factor.	"Value::Float1"	0.7
padding	Extension of rectangle. Do not change.	"Value::Float4"	see below
redetectTolerance	Redetection threshold.	"Value::Int1"	0
useLNet	Whether to use LNet or not.	"Value::Int"	1

minFaceSize and scaleFactor accelerate face detection at the cost of lower recall for smaller faces.

All detections with a size (max of width/height) smaller than strictlyMinSize will be excluded from the final result.

Example:

```
<section name="FaceDetV2::Settings">
  <param name="FirstThreshold" type="Value::Float1" x="0.51385"/>
  <param name="SecondThreshold" type="Value::Float1" x="0.248"/>
  <param name="ThirdThreshold" type="Value::Float1" x="0.76"/>
  <param name="minFaceSize" type="Value::Int1" x="50" />
  <param name="strictlyMinSize" type="Value::Int1" x="0" />
  <param name="scaleFactor" type="Value::Float1" x="0.7" />
  <param name="padding" type="Value::Float4" x="-0.20099958" y="
    0.10210337" z="0.20363552" w="0.08490226" />
  <param name="redetectTolerance" type="Value::Int1" x="0" />
  <param name="useLNet" type="Value::Int1" x="0" />
</section>
```

LNet

This group of parameters is non-public. Do not change any of the parameters.

HeadPoseEstimator settings

In mobile mode, HeadPose estimator is able to compute head pose angles using raw input image data.

EyeEstimator settings

This estimator aims to determine:

- Eye state: Open, Closed, Occluded;
- Precise eye iris location as an array of landmarks;
- Precise eyelid location as an array of landmarks.

To determine more exact eye state additional auxiliary model `eye_status_estimation_*.plan` is used. You can enable this auxiliary model through config (`faceengine.conf`).

Parameter	Description	Type	Default value
<code>useStatusPlan</code>	0 - Off, 1 - On	<code>"Value::Int1"</code>	1

Example:

```
<section name="EyeEstimator::Settings">  
  <param name="useStatusPlan" type="Value::Int1" x="1"/>  
</section>
```

Medical mask estimator settings

Medical mask estimator predicts predominant mask features.

Estimator accuracy depends on thresholds listed below.

If accuracy (low FPR) is more important, TPR could be sacrificed by heightening the threshold.

Corresponding FPR and TPR values are also listed below.

Thresholds for MedicalMaskEstimation

The below parameters specify parameter thresholds in the [0..1] range and are of the "Value : Float1" type.

maskThreshold

Threshold range	Default threshold	FPR range	TPR range
0.65 - 0.9	0.65	0.014 - 0.01	0.976 - 0.886

noMaskThreshold

Threshold range	Default threshold	FPR range	TPR range
0.65 - 0.79	0.65	0.01 - 0.005	0.94 - 0.903

occludedFaceThreshold

Threshold range	Default threshold	FPR range	TPR range
0.5 - 0.602	0.5	0.016 - 0.01	0.924 - 0.881

Thresholds for MedicalMaskEstimationExtended

The below parameters specify parameter thresholds in the [0..1] range and are of the "Value : Float1" type.

maskExtendedThreshold

Threshold range	Default threshold	FPR range	TPR range
0.65 - 0.784	0.65	0.013 - 0.01	0.923 - 0.894

noMaskExtendedThreshold

Threshold range	Default threshold	FPR range	TPR range
0.65 - 0.79	0.65	0.01 - 0.005	0.94 - 0.903

maskNotInPlaceExtendedThreshold

Threshold range	Default threshold	FPR range	TPR range
0.65 - 0.85	0.65	0.009 - 0.005	0.918 - 0.833

occludedFaceExtendedThreshold

Threshold range	Default threshold	FPR range	TPR range
0.5 - 0.602	0.5	0.016 - 0.01	0.924 - 0.881

Example:

```
<section name="MedicalMaskEstimatorV3::Settings">
  <param name="maskExtendedThreshold" type="Value::Float1" x="0.65"/>
  <param name="noMaskExtendedThreshold" type="Value::Float1" x="0.65"
    />
  <param name="maskNotInPlaceExtendedThreshold" type="Value::Float1" x=
    "0.65"/>
  <param name="occludedFaceExtendedThreshold" type="Value::Float1" x="
    0.5"/>
  <param name="maskThreshold" type="Value::Float1" x="0.65"/>
  <param name="noMaskThreshold" type="Value::Float1" x="0.65"/>
  <param name="occludedFaceThreshold" type="Value::Float1" x="0.65"/>
</section>
```

Mouth Estimator settings

Mouth estimator predicts predominant mouth state.

Estimator accuracy depends on thresholds listed below.

FPR and TPR values are specified for 0.5 threshold

Table 12: “Thresholds for MouthEstimation”

Parameter	Description	Type	Default value	Threshold range	TPR	FPR
occlusionThreshold	Threshold in the [0..1] range	Value::Float1	0.5	0.4 – 0.6	0.96	0.009
smileThreshold	Threshold in the [0..1] range.	Value::Float1	0.5	0.4 – 0.6	0.97	0.04
openThreshold	Threshold in the [0..1] range.	Value::Float1	0.5	0.4 – 0.6	0.986	0.01

Example:

```
<section name="MouthEstimator::Settings">
  <param name="occlusionThreshold" type="Value::Float1" x="0.5"/>
  <param name="smileThreshold" type="Value::Float1" x="0.5"/>
  <param name="openThreshold" type="Value::Float1" x="0.5"/>
</section>
```

Face Occlusion Estimator Settings

The Face Occlusion estimator predicts occlusion of various face areas such as the forehead, eyes, nose, mouth, and lower face.

Estimator accuracy depends on thresholds listed below.

Parameter	Description	Type	Default	Threshold range
normalHairCoeff	Threshold for hair*.	Value::Float1	0.15	0.0 - 1.0
overallOcclusionThreshold	Overall threshold.	Value::Float1	0.14	0.0 - 1.0
foreheadThreshold	Threshold for forehead.	Value::Float1	0.1	0.0 - 1.0
eyeThreshold	Threshold for eye.	Value::Float1	0.4	0.0 - 1.0
noseThreshold	Threshold for nose.	Value::Float1	0.4	0.0 - 1.0
mouthThreshold	Threshold for mouth.	Value::Float1	0.15	0.0 - 1.0
lowerFaceThreshold	Threshold for lower face.	Value::Float1	0.2	0.0 - 1.0

Note:

The `normalHairCoeff` parameter determines whether the presence of hair is considered when analyzing face occlusion.

- If the percentage of hair occlusion is lower than `normalHairCoeff`, hair is not taken into account in the analysis.
- If the hair occlusion percentage exceeds `normalHairCoeff`, the excess hair occlusion is added to the overall face occlusion percentage.

Example calculation:

- Overall face occlusion score without hair: 0.05
- Overall face occlusion score with hair: 0.2
- `normalHairCoeff`: 0.15

In this case, the resulting overall face occlusion score would be calculated as follows:

- Resulting score = $0.05 + (0.2 - 0.15) = 0.1$.

Example:

```
<section name="FaceOcclusionEstimator::Settings">
  <param name="normalHairCoeff" type="Value::Float1" x="0.15"/>
  <param name="overallOcclusionThreshold" type="Value::Float1" x="0.07"/>
  <param name="foreheadThreshold" type="Value::Float1" x="0.1"/>
  <param name="eyeThreshold" type="Value::Float1" x="0.15"/>
  <param name="noseThreshold" type="Value::Float1" x="0.2"/>
  <param name="mouthThreshold" type="Value::Float1" x="0.15"/>
  <param name="lowerFaceThreshold" type="Value::Float1" x="0.2"/>
</section>
```

Quality estimator settings

Quality estimator looks at several image parameters, like lightness (think overexposure), darkness (think underexposure), blurriness, illumination uniformity value, specularity value. Every float value is comparing with according threshold.

Parameter	Type	Default value
blurThreshold	"Value::Float1"	x="0.58"
lightThreshold	"Value::Float1"	x="0.58"
darknessThreshold	"Value::Float1"	x="0.52"
illuminationThreshold	"Value::Float1"	x="0.3"
specularityThreshold	"Value::Float1"	x="0.3"
usePlanV1	"Value::Int1"	x="1"
usePlanV2	"Value::Int1"	x="1"

Example:

```
<section name="QualityEstimator::Settings">
  <param name="blurThreshold" type="Value::Float1" x="0.58"/>
  <param name="lightThreshold" type="Value::Float1" x="0.58"/>
  <param name="darknessThreshold" type="Value::Float1" x="0.52"/>
  <param name="illuminationThreshold" type="Value::Float1" x="0.3"/>
  <param name="specularityThreshold" type="Value::Float1" x="0.3"/>
  <param name="usePlanV1" type="Value::Int1" x="1" />
  <param name="usePlanV2" type="Value::Int1" x="1" />
</section>
```

Note: usePlanV1 toggles the Quality estimation, usePlanV2 toggles the SubjectiveQuality estimation. Note that you cannot disable both the parameters at the same time. In case you do this, you will receive the `fsdk::FSDKError::InvalidConfig` error code and the following logs:

```
[27.06.2024 12:38:59] [Error] QualityEstimator::Settings Failed to create
QualityEstimator! The both parameters: "usePlanV1" and "usePlanV2" in
section "QualityEstimator::Settings" are disabled at the same time.
```

GlassesEstimator settings

The glasses estimator estimates what types of glasses, if any, a person is currently wearing. Estimation quality depends on threshold values listed below. These threshold values are set to optimal by default.

Parameter	Description	Type	Default value
noGlassesThreshold	noGlasses threshold in [0..1] range.	"Value::Float1"	1
eyeGlassesThreshold	eyeGlasses threshold in [0..1] range.	"Value::Float1"	1
sunGlassesThreshold	sunGlasses threshold in [0..1] range.	"Value::Float1"	1

Example:

```
<section name="GlassesEstimator::Settings">
  <param name="noGlassesThreshold" type="Value::Float1" x="1"/>
  <param name="eyeGlassesThreshold" type="Value::Float1" x="1"/>
  <param name="sunGlassesThreshold" type="Value::Float1" x="1"/>
</section>
```


LivenessOneShotRGBEstimator settings

This estimator tells whether the person's face is real or fake (photo, printed image). Thresholds are listed below.

Liveness protects from presentation attacks - when user tries to cheat biometric system by demonstrating fake face to the face capturing camera, but not from image substitution attacks - when fake image is sent directly to the system, bypassing the camera.

LivenessOneShotRGBEstimator supports images, which are captured on Mobile devices or Webcam (PC or laptop). Correct working of the estimator with other source images is not guaranteed.

Supported shooting mode: cooperative, which means that user must interact with the camera and look at it.

User scenarios examples: authentication in mobile application, confirmation of transactions with biometric facial verification.

Image resolution minimum requirements:

- Mobile devices - 720 × 960 px
- Webcam (PC or laptop) - 1280 x 720 px

Parameter	Description	Type	Default value
version	Estimator version, 11 or 10	"Value::Int1"	11
netType	NET version.	"Value::Int1"	0
realThreshold	Threshold in [0..1] range.	"Value::Float1"	0.5
qualityThreshold	Threshold in [0..1] range.	"Value::Float1"	0.5
calibrationCoeff_v11	Coefficient in [0..1] range.	"Value::Float1"	0.8868
calibrationCoeff_v12	Coefficient in [0..1] range.	"Value::Float1"	0.9027
mobileRealThreshold	Threshold in [0..1] range.	"Value::Float1"	0.5
mobileQualityThreshold	Threshold in [0..1] range.	"Value::Float1"	0.5
mobileCalibrationCoeff_v11	Coefficient in [0..1] range.	"Value::Float1"	0.9967
mobileCalibrationCoeff_v12	Coefficient in [0..1] range.	"Value::Float1"	0.9967

Parameter	Description	Type	Default value
liteRealThreshold	Threshold in [0..1] range.	"Value::Float1" "	0.5
liteQualityThreshold	Threshold in [0..1] range.	"Value::Float1" "	0.5
liteCalibrationCoeff_v11	Coefficient in [0..1] range.	"Value::Float1" "	0.9923
liteCalibrationCoeff_v12	Coefficient in [0..1] range.	"Value::Float1" "	0.9658

Parameters ending with _v11 apply only to version 11 of the estimator (version = 11). Parameters ending with _v12 apply only to version 12 of the estimator (version = 12). Other parameters apply to both versions.

For example, if calibrationCoeff_v<version> for the selected version is found, it is loaded. Otherwise, if calibrationCoeff is found, it is loaded. Otherwise, calibrationCoeff's default value (hardcoded in SDK) is used.

```
<section name="LivenessOneShotRGBEstimator::Settings">
  <param name="version" type="Value::Int1" x="12" />
  <param name="netType" type="Value::Int1" x="0" />
  <!--Parameters for backend version (netType == 0) -->
  <param name="realThreshold" type="Value::Float1" x="0.5"/>
  <param name="qualityThreshold" type="Value::Float1" x="0.5" />

  <!-- If calibrationCoeff_v<version> for the selected version is found,
  it is loaded. Otherwise, if calibrationCoeff is found, it is loaded.
  Otherwise, calibrationCoeff's default value (hardcoded in SDK) is used.
  -->
  <param name="calibrationCoeff_v11" type="Value::Float1" x="0.8868"/>
  <param name="calibrationCoeff_2xl_v11" type="Value::Float1" x="0.8705"/>
  <param name="calibrationCoeff_v12" type="Value::Float1" x="0.9027"/>
  <param name="calibrationCoeff_2xl_v12" type="Value::Float1" x="0.88"/>
  <!--Parameters for mobile version (netType == 1) -->
  <param name="mobileRealThreshold" type="Value::Float1" x="0.5"/>
  <param name="mobileQualityThreshold" type="Value::Float1" x="0.5" />
  <param name="mobileCalibrationCoeff_v11" type="Value::Float1" x="0.9967"
    />
  <param name="mobileCalibrationCoeff_v12" type="Value::Float1" x="0.9967"
    />
</section>
```

```
<!--Parameters for lite version (netType == 2) -->  
<param name="liteRealThreshold" type="Value::Float1" x="0.5"/>  
<param name="liteQualityThreshold" type="Value::Float1" x="0.5" />  
<param name="liteCalibrationCoeff_v11" type="Value::Float1" x="0.9923"/>  
<param name="liteCalibrationCoeff_v12" type="Value::Float1" x="0.9658"/>  
</section>
```

NIRLivenessEstimator settings

NIRLivenessEstimator determines whether a person's face is real or fake, such as a photo or printed image. The input image must be captured by an infrared camera. The estimator returns a boolean indicating whether the face is real (`true`) or fake (`false`).

Configuration parameters:

Parameter	Description	Type	Default value
<code>realThreshold</code>	Threshold in [0..1] range.	<code>Value::Float1</code>	<code>0.5</code>
<code>defaultEstimatorMode</code>	Configuration of plan files usage.	<code>Value::Int1</code>	<code>2</code>

```
<section name="NIRLivenessEstimator::Settings">
  <param name="realThreshold" type="Value::Float1" x="0.5"/>
  <!-- Currently, available values to select the estimation mode are:
        1 and 2. -->
  <param name="defaultEstimatorMode" type="Value::Int1" x="2"/>
</section>
```

Estimation modes:

The estimator can operate in two modes: a fast mode and a slower but more accurate mode. By default, it is recommended to use the more accurate mode, which is designated as mode 2.

Currently, the available values for selecting the estimation mode are: `Default`, `M1`, and `M2`. The `Default` scenario means that the mode is specified in the configuration file (see `ISettingsProvider`).

Implementation description:

NIRLiveness Estimation Mode Enum

```
enum class NIRLivenessMode {
    Default, // Specified in config file.
    M1,      // M1.
    M2       // M2.
};
```

Runtime settings

Runtime configuration file provides parameters that user can tweak to achieve optimal performance of their app.

Note: The setting `<param name="numThreads" type="Value::Int1" x="-1"/>` means that will be taken the maximum number of available threads. This number of threads is equal to according number of available processor cores.

The name of runtime configuration file is `runtime.conf` and its placed in data directory. Its settings are described below:

Parameter	Description	Type	Default value
<code>cpuClass</code>	Class of cpu by supported instructions - cpu, arm, auto.	"Value::String"	"auto"
<code>deviceClass</code>	Execution device type - cpu, gpu.	"Value::String"	"cpu"
<code>numThreads</code>	Number of worker threads. Default: number of CPU logical cores.	"Value::Int1"	-1
<code>verboseLogging</code>	Level of log verbosity. 1 - Errors, 2 - Warnings, 3 - Info, 4 - Debug.	"Value::Int1"	0
<code>programCacheSize</code>	Maximum number of Program objects in cache. Should be less than 10000.	"Value::Int1"	128

Verbosity level sets the upper limit of what type of messages may be printed out. For example, if user set `verboseLogging` to 3, it means that Errors, Warnings and Info messages will be printed out to the console. Verbose level of 0 indicates that there are no logging messages printed out at all.

Increasing the `programCacheSize` increases memory usage and potentially improves performance. Be careful, too large a value of this parameter can lead to a crash due to insufficient memory.

Example:

```
<section name="Runtime">
  <param name="cpuClass" type="Value::String" text="auto" />
  <param name="deviceClass" type="Value::String" text="cpu" />
  <param name="numThreads" type="Value::Int1" x="-1" />
  <param name="verboseLogging" type="Value::Int1" x="0" />
  <param name="programCacheSize" type="Value::Int1" x="128" />
</section>
```