



# VStabiliser interface C++ class

---

v2.1.0

---

## Table of contents

---

- [Overview](#)
- [Versions](#)
- [VStabiliser class description](#)
  - [VStabiliser class declaration](#)
  - [getVersion method](#)
  - [initVStabiliser method](#)
  - [setParam method](#)
  - [getParam method](#)
  - [getParams method](#)
  - [executeCommand method](#)
  - [stabilise method](#)
  - [getOffsets methods](#)
  - [encodeSetParamCommand method](#)
  - [encodeCommand method](#)
  - [decodeCommand method](#)
- [Data structures](#)
  - [VStabiliserParam enum](#)
  - [VStabiliserCommand enum](#)
- [VStabiliserParams class description](#)
  - [VStabiliserParams class declaration](#)
  - [Serialize video stabiliser params](#)
  - [Deserialize video stabiliser params](#)
  - [Read params from JSON file and write to JSON file](#)
- [Build and connect to your project](#)

# Overview

**VStabiliser** C++ library provides standard interface as well defines data structures and rules for different implementation of video stabilisation algorithms. **VStabiliser** interface class doesn't do anything, just provides interface, data structures and service functions to encode/decode parameter and commands. Different video stabiliser classes inherit interface from **VStabiliser** C++ class. **VStabiliser.h** file contains list of data structures (**VStabiliserCommand** enum, **VStabiliserParam** enum and **VStabiliserParams** class) and **VStabiliser** class declaration. **VStabiliserCommand** enum contains IDs of commands supported by **VStabiliser** class. **VStabiliserParam** enum contains IDs of params supported by **VStabiliser** class. **VStabiliserParams** class contains fields for video stabiliser params values and provides methods to encode/decode and read/write params from JSON file. All video stabilisers should include params and commands listed in **VStabiliser.h** file. **VStabiliser** class depends on two external libraries (included as submodule): [Frame](#) (video frame data structure) and [ConfigReader](#) (provides methods to read/write JSON config files).

## Versions

**Table 1** - Library versions.

Version	Release date	What's new
1.0.0	18.05.2023	First version.
2.0.0	11.07.2023	<ul style="list-style-type: none"><li>- Added VStabiliserParams class to keep library parameters.</li><li>- Added method to encode/decode commands.</li><li>- Added test application.</li><li>- Added license.</li><li>- Repository made public.</li></ul>
2.0.1	12.07.2023	<ul style="list-style-type: none"><li>- Fixed typo in VStabiliserParams</li><li>- Updated test application.</li></ul>
2.1.0	30.07.2023	<ul style="list-style-type: none"><li>- Switched from radians to radians in params.</li></ul>

## VStabiliser class description

### VStabiliser class declaration

**VStabiliser.h** file contains **VStabiliser** class declaration. Class declaration:

```
class VStabiliser
{
public:
    /**
     * @brief Get string of current class version.
     * @return String of current class version.
```

```

*/
static std::string getVersion();
/**
 * @brief Init all video stabiliser params by params structure.
 * @param params Parameters class.
 * @return TRUE if params was accepted or FALSE if not.
 */
virtual bool initVStabiliser(VStabiliserParams& params) = 0;
/**
 * @brief Set param.
 * @param id Param ID.
 * @param value Param value.
 * @return TRUE if param was accepted or FALSE if not.
 */
virtual bool setParam(VStabiliserParam id, float value) = 0;
/**
 * @brief Get param.
 * @param id Param ID.
 * @return Param value or -1 if this param not supported by implementation.
 */
virtual float getParam(VStabiliserParam id) = 0;
/**
 * @brief Get params.
 * @return Params class.
 */
virtual VStabiliserParams getParams() = 0;
/**
 * @brief Execute command.
 * @param id Command ID.
 * @return TRUE if the command executed or FALSE if not.
 */
virtual bool executeCommand(VStabiliserCommand id) = 0;
/**
 * @brief Stabilise video frame.
 * @param src Source videoe frame.
 * @param dst Result video frame.
 * @return TRUE if video frame processed or FALSE in case any errors.
 */
virtual bool stabilise(cr::video::Frame& src, cr::video::Frame& dst) = 0;
/**
 * @brief Get offsets: horithontal, vertical and rotation. The method must
 * return current offsets which implemented to last processed video frame.
 * @param dx Horizontal offset.
 * @param dy Vertical offset.
 * @param dA Rotational angle, radians.
 */
virtual void getOffsets(float& dx, float& dy, float& dA) = 0;
/**
 * @brief Encode set param command.
 * @param data Pointer to data buffer. Must have size >= 11.
 * @param size Size of encoded data.
 * @param id Parameter id.
 * @param value Parameter value.
 */
static void encodeSetParamCommand(

```

```

        uint8_t* data, int& size, VStabiliserParam id, float value);

/**
 * @brief Encode command.
 * @param data Pointer to data buffer. Must have size >= 11.
 * @param size Size of encoded data.
 * @param id Command ID.
 */
static void encodeCommand(
    uint8_t* data, int& size, VStabiliserCommand id);

/**
 * @brief Decode command.
 * @param data Pointer to command data.
 * @param size Size of data.
 * @param paramId Output command ID.
 * @param commandId Output command ID.
 * @param value Param or command value.
 * @return 0 - command decoded, 1 - set param command decoded, -1 - error.
 */
static int decodeCommand(uint8_t* data,
                        int size,
                        VStabiliserParam& paramId,
                        VStabiliserCommand& commandId,
                        float& value);

};

```

## getVersion method

**getVersion()** method return string of current version of **VStabiliser** class. Method declaration:

```
static std::string getVersion();
```

Method can be used without **VStabiliser** class instance. Example:

```
cout << "VStabiliser class version: " << VStabiliser::getVersion() << endl;
```

Console output:

```
VStabiliser class version: 2.0.0
```

## initVStabiliser method

**initVStabiliser(...)** method intended to initialize video stabiliser parameters by parameters structure. Method copy all video stabiliser parameter to internal variables. Method declaration:

```
virtual bool initVStabiliser(VStabiliserParams& params) = 0;
```

Parameter	Description
params	Parameters class (see <b>VStabiliserParams</b> class description).

**Returns:** TRUE if params was accepted or FALSE if not.

## setParam method

**setParam(...)** method intended to change video stabiliser parameter. Method declaration:

```
virtual bool setParam(cr::vstab::VStabiliserParam id, float value) = 0;
```

Parameter	Description
id	Parameter ID according to <b>VStabiliserParam</b> enum.
value	Parameter value. Depends on parameter ID.

**Returns:** TRUE if param was accepted or FALSE if not.

## getParam method

**getParam(...)** method intended to get video stabiliser parameter value. Method declaration:

```
virtual float getParam(cr::vstab::VStabiliserParam id) = 0;
```

Parameter	Description
id	Parameter ID according to <b>VStabiliserParam</b> enum.

**Returns:** Parameter value or -1 if this param not supported by implementation.

## getParams method

**getParams(...)** method designed to obtain all video stabiliser parameters. Method declaration:

```
virtual VStabiliserParams getParams() = 0;
```

**Returns:** **VStabiliserParams** class which contains all current parameters value.

## executeCommand method

**executeCommand(...)** method intended to execute video stabiliser command. Method declaration:

```
virtual bool executeCommand(cr::vstab::VStabiliserCommand id) = 0;
```

Parameter	Description
id	Command ID according to <b>VStabiliserCommand</b> enum.

**Returns:** TRUE if the command executed or FALSE if not.

## stabilise method

**stabilise(...)** method performs video stabilisation. Method declaration:

```
virtual bool stabilise(cr::video::Frame& src, cr::video::Frame& dst) = 0;
```

Parameter	Description
src	Source frame. The methods accepts only RAW frame data (not compressed pixel formats, see description of <b>Frame</b> class). Particular implementation can support all, only one or few pixel formats listed in <b>Frame</b> class description.
dst	Result frame. The pixel format of the result frame must be the same as source frame. If stabilisation disabled (param <b>MODE</b> set to 0) the library must copy data from source frame to result frame.

**Returns:** TRUE if video frame processed or FALSE in case any errors.

## getOffsets methods

**getOffsets(...)** method returns horizontal offset (pixels), vertical offset (pixels) and rotation angle (radians) implemented to last processed video frame (**stabilise** method). Method declaration:

```
virtual void getOffsets(float& dx, float& dy, float& dA) = 0;
```

Parameter	Description
dX	Reference to output value of horizontal offset (pixels) implemented to last processed video frame.
dY	Reference to output value of vertical offset (pixels) implemented to last processed video frame.
dA	Reference to output value of rotational angle (radiansee) implemented to last processed video frame.

## encodeSetParamCommand method

**encodeSetParamCommand(...)** static method designed to encode command to change any parameters of remote video stabiliser. To control a stabiliser remotely, the developer has to develop his own protocol and according to it encode the command and deliver it over the communication channel. To simplify this, the **VStabiliser** class contains static methods for encoding the control commands. The **VStabiliser** class provides two types of commands: a parameter change command (SET\_PARAM) and an action command (COMMAND). **encodeSetParamCommand(...)** designed to encode SET\_PARAM command. Method declaration:

```
static void encodeSetParamCommand(uint8_t* data, int& size, VStabiliserParam id, float value);
```

Parameter	Description
data	Pointer to data buffer for encoded command. Must have size >= 11.
size	Size of encoded data. Will be 11 bytes.
id	Parameter ID according to <b>VStabiliserParam</b> enum.
value	Parameter value.

**SET\_PARAM** command format:

Byte	Value	Description
0	0x01	SET_PARAM command header value.
1	0x02	Major version of VStabiliser class.
2	0x00	Minor version of VStabiliser class.
3	id	Parameter ID <b>int32_t</b> in Little-endian format.
4	id	Parameter ID <b>int32_t</b> in Little-endian format.
5	id	Parameter ID <b>int32_t</b> in Little-endian format.
6	id	Parameter ID <b>int32_t</b> in Little-endian format.
7	value	Parameter value <b>float</b> in Little-endian format.
8	value	Parameter value <b>float</b> in Little-endian format.
9	value	Parameter value <b>float</b> in Little-endian format.
10	value	Parameter value <b>float</b> in Little-endian format.

**encodeSetParamCommand(...)** is static and used without **VStabiliser** class instance. This method used on client side (control system). Example:

```

// Buffer for encoded data.
uint8_t data[11];
// Size of encoded data.
int size = 0;
// Random parameter value.
float outValue = (float)(rand() % 20);
// Encode command.
VStabiliser::encodeSetParamCommand(data, size, VStabiliserParam::INSTANT_X_OFFSET,
outValue);

```

## encodeCommand method

**encodeCommand(...)** static method designed to encode video stabiliser action command. To control a stabiliser remotely, the developer has to develop his own protocol and according to it encode the command and deliver it over the communication channel. To simplify this, the **VStabiliser** class contains static methods for encoding the control commands. The **VStabiliser** class provides two types of commands: a parameter change command (SET\_PARAM) and an action command (COMMAND). **encodeCommand(...)** designed to encode COMMAND command (action command). Method declaration:

```

static void encodeCommand(uint8_t* data, int& size, VStabiliserCommand id);

```

Parameter	Description
data	Pointer to data buffer for encoded command. Must have size >= 11.
size	Size of encoded data. Will be 7 bytes.
id	Command ID according to <b>VStabiliserCommand</b> enum.

**COMMAND** format:

Byte	Value	Description
0	0x00	SET_PARAM command header value.
1	0x04	Major version of VStabiliser class.
2	0x00	Minor version of VStabiliser class.
3	id	Command ID <b>int32_t</b> in Little-endian format.
4	id	Command ID <b>int32_t</b> in Little-endian format.
5	id	Command ID <b>int32_t</b> in Little-endian format.
6	id	Command ID <b>int32_t</b> in Little-endian format.

**encodeCommand(...)** is static and used without **VStabiliser** class instance. This method used on client side (control system). Encoding example:



```
// Buffer for encoded data.
uint8_t data[11];
// Size of encoded data.
int size = 0;
// Encode command.
VStabiliser::encodeCommand(data, size, VStabiliserCommand::ON);
```

## decodeCommand method

**decodeCommand(...)** static method designed to decode command on video stabiliser side. To control a stabiliser remotely, the developer has to develop his own protocol and according to it encode the command and deliver it over the communication channel. To simplify this, the **VStabiliser** interface class contains static method to decode input command (commands should be encoded by methods **encodeSetParamsCommand(...)** or **encodeCommand(...)**). The **VStabiliser** class provides two types of commands: a parameter change command (SET\_PARAM) and an action command (COMMAND). Method declaration:

```
static int decodeCommand(uint8_t* data, int size, VStabiliserParam& paramId,
VStabiliserCommand& commandId, float& value);
```

Parameter	Description
data	Pointer to input command.
size	Size of command. Should be 11 bytes (for SET_PARAMS) or 7 bytes for (COMMAND).
paramId	Video stabiliser parameter ID according to <b>VStabiliserParam</b> enum. After decoding SET_PARAM command the method will return parameter ID.
commandId	VStabiliser command ID according to <b>VStabiliserCommand</b> enum. After decoding COMMAND the method will return command ID.
value	Video stabiliser parameter value after decoding SET_PARAM command.

**Returns:** **0** - in case decoding COMMAND, **1** - in case decoding SET\_PARAM command or **-1** in case errors.

## Data structures

### VStabiliserParam enum

**VStabiliserParam** enum lists video stabiliser parameters to set or to obtain. Enum declaration.

```
enum class VStabiliserParam
{
    /// Scale factor. value depends on implementation. Default:
    /// If 1 the library will process original frame size, if 2
    /// the library will scale original frame size by 2, if 3 - by 3.
```

```

SCALE_FACTOR = 1,
/// Maximum horizontal image shift in pixels per video frame. If image shift
/// bigger than this limit the library should compensate only X_OFFSET_LIMIT
/// shift.
X_OFFSET_LIMIT,
/// Maximum vertical image shift in pixels per video frame. If image shift
/// bigger than this limit the library should compensate only Y_OFFSET_LIMIT
/// shift.
Y_OFFSET_LIMIT,
/// Maximum rotational image angle in radians per video frame. If image
/// absolute rotational angle bigger than this limit the library should
/// compensate only A_OFFSET_LIMIT angle.
A_OFFSET_LIMIT,
/// Horizontal smoothing coefficient of constant camera movement. The range
/// of values depends on the specific implementation of the stibilisation
/// algorithm. Default values [0-1]: 0 - the library will not compensate for
/// constant camera motion, video will not be stabilized, 1 - no smoothing
/// of constant camera motion (the library will compensate for the current
/// picture drift completely without considering constant motion).
X_FILTER_COEFF,
/// Vertical smoothing coefficient of constant camera movement. The range
/// of values depends on the specific implementation of the stibilisation
/// algorithm. Default values [0-1]: 0 - the library will not compensate for
/// constant camera motion, video will not be stabilized, 1 - no smoothing
/// of constant camera motion (the library will compensate for the current
/// picture drift completely without considering constant motion).
Y_FILTER_COEFF,
/// Rotational smoothing coefficient of constant camera movement. The range
/// of values depends on the specific implementation of the stibilisation
/// algorithm. Default values [0-1]: 0 - the library will not compensate for
/// constant camera motion, video will not be stabilized, 1 - no smoothing
/// of constant camera motion (the library will compensate for the current
/// picture drift completely without considering constant motion).
A_FILTER_COEFF,
/// Stabilisation mode:
/// 0 - Stabilisation off. The library should just copy input image.
/// 1 - Stabilisation on.
MODE,
/// Transparent border mode:
/// 0 - Not transparent borders (black borders).
/// 1 - Transparent borders (parts of previous images).
/// Particular implementation can have additional modes.
TRANSPARENT_BORDER,
/// Constant horizontal image offset in pixels. The library should add this
/// offset to each processed video frame.
CONST_X_OFFSET,
/// Constant vertical image offset in pixels. The library should add this
/// offset to each processed video frame.
CONST_Y_OFFSET,
/// Constant rotational angle in radians. The library should add this
/// offset to each processed video frame.
CONST_A_OFFSET,
/// Instant (for one frame) horizontal image offset in pixels. The library
/// should add this offset to next processed video frame.
INSTANT_X_OFFSET,

```

```

    /// Instant (for one frame) vertical image offset in pixels. The library
    /// should add this offset to next processed video frame.
    INSTANT_Y_OFFSET,
    /// Instant (for one frame) rotational angle in radians. The library
    /// should add this offset to next processed video frame.
    INSTANT_A_OFFSET,
    /// Algorithm type. Default values:
    /// 0 - 2D type 1. Stabilisation only on horizontal and vertical.
    /// 1 - 2D type 2. Stabilisation only on horizontal and vertical.
    /// 2 - 3D. Stabilisation on horizontal and vertical + rotation.
    /// Particular implementation can have unique values.
    TYPE,
    /// Cut frequency, Hz. Stabiliser will block vibrations with frequency
    /// > CUT_FREQUENCY_HZ.
    CUT_FREQUENCY_HZ,
    /// Frames per second of input video.
    FPS,
    /// Processing time, mks. Processing time for last video frame.
    PROCESSING_TIME_MKS,
    /// Logging mode. Values: 0 - Disable, 1 - Only file, 2 - Only terminal,
    /// 3 - File and terminal.
    LOG_MODE
};

```

**Table 2** - Video stabiliser parameters description.

Parameter	Description
SCALE_FACTOR	Scale factor. Value depends on implementation. Default: If 1 the library will process original frame size, if 2 the library will scale original frame size by 2, if 3 - by 3.
X_OFFSET_LIMIT	Maximum horizontal image shift in pixels per video frame. If image shift bigger than this limit the library should compensate only X_OFFSET_LIMIT shift.
Y_OFFSET_LIMIT	Maximum vertical image shift in pixels per video frame. If image shift bigger than this limit the library should compensate only Y_OFFSET_LIMIT shift.
A_OFFSET_LIMIT	Maximum rotational image angle in radians per video frame. If image absolute rotational angle bigger than this limit the library should compensate only A_OFFSET_LIMIT angle.
X_FILTER_COEFF	Horizontal smoothing coefficient of constant camera movement. The range of values depends on the specific implementation of the stabilisation algorithm. Default values [0-1]: 0 - the library will not compensate for constant camera motion, video will not be stabilized, 1 - no smoothing of constant camera motion (the library will compensate for the current picture drift completely without considering constant motion).

Parameter	Description
Y_FILTER_COEFF	Vertical smoothing coefficient of constant camera movement. The range of values depends on the specific implementation of the stabilisation algorithm. Default values [0-1]: 0 - the library will not compensate for constant camera motion, video will not be stabilized, 1 - no smoothing of constant camera motion (the library will compensate for the current picture drift completely without considering constant motion).
A_FILTER_COEFF	Rotational smoothing coefficient of constant camera movement. The range of values depends on the specific implementation of the stabilisation algorithm. Default values [0-1]: 0 - the library will not compensate for constant camera motion, video will not be stabilized, 1 - no smoothing of constant camera motion (the library will compensate for the current picture drift completely without considering constant motion).
MODE	Stabilisation mode: 0 - Stabilisation off. The library should just copy input image. 1 - Stabilisation on.
TRANSPARENT_BORDER	Transparent border mode: 0 - Not transparent borders (black borders). 1 - Transparent borders (parts of previous images). Particular implementation can have additional modes.
CONST_X_OFFSET	Constant horizontal image offset in pixels. The library should add this offset to each processed video frame.
CONST_Y_OFFSET	Constant vertical image offset in pixels. The library should add this offset to each processed video frame.
CONST_A_OFFSET	Constant rotational angle in radians. The library should add this offset to each processed video frame.
INSTANT_X_OFFSET	Instant (for one frame) horizontal image offset in pixels. The library should add this offset to next processed video frame.
INSTANT_Y_OFFSET	Instant (for one frame) vertical image offset in pixels. The library should add this offset to next processed video frame.
INSTANT_A_OFFSET	Instant (for one frame) rotational angle in radians. The library should add this offset to next processed video frame.
TYPE	Algorithm type. Particular implementation can have unique values. Default values: 0 - 2D. Stabilisation only on horizontal and vertical directions. 1 - 3D. Stabilisation on horizontal and vertical directions + rotation.
CUT_FREQUENCY_HZ	Cut frequency, Hz. Stabiliser will block vibrations with frequency CUT_FREQUENCY_HZ.
FPS	Frames per second of input video.

Parameter	Description
PROCESSING_TIME_MKS (Read only parameter)	Processing time, mks. Processing time for last video frame.
LOG_MODE	Logging mode. Values: 0 - Disable, 1 - Only file, 2 - Only terminal, 3 - File and terminal.

## VStabiliserCommand enum

**VStabiliserCommand** enum lists video stabiliser commands. Enum declaration.

```
enum class VStabiliserCommand
{
    /// Reset stabilisation algorithm.
    RESET = 1,
    /// Enable stabilisation. After execution parameter MODE must be set to 1.
    ON,
    /// Disable stabilisation. After execution parameter MODE must be set to 0.
    OFF
};
```

**Table 3** - Video stabiliser commands description.

Parameter	Description
RESET	Reset stabilisation algorithm.
ON	Enable stabilisation. After execution parameter <b>MODE</b> must be set to 1.
OFF	Disable stabilisation. After execution parameter <b>MODE</b> must be set to 0.

## VStabiliserParams class description

**VStabiliserParams** class used for video stabiliser initialization (**initVStabiliser(...)** method) or to get all actual params (**getParams()** method). Also **VStabiliserParams** provide structure to write/read params from JSON files (**JSON\_READABLE** macro) and provide methos to encode and decode params.

## VStabiliserParams class declaration

**VStabiliser.h** file contains **VStabiliserParams** class declaration. Class declaration:

```
class VStabiliserParams
{
public:
    /// Scale factor. value depends on implementation. Default:
    /// If 1 the library will process original frame size, if 2
    /// the library will scale original frame size by 2, if 3 - by 3.
```

```

int scaleFactor{1};
/// Maximum horizontal image shift in pixels per video frame. If image shift
/// bigger than this limit the library should compensate only xOffsetLimit
/// shift.
int xOffsetLimit{150};
/// Maximum vertical image shift in pixels per video frame. If image shift
/// bigger than this limit the library should compensate only yOffsetLimit
/// shift.
int yOffsetLimit{150};
/// Maximum rotational image angle in radians per video frame. If image
/// absolute rotational angle bigger than this limit the library should
/// compensate only aOffsetLimit angle.
float aOffsetLimit{10.0f};
/// Horizontal smoothing coefficient of constant camera movement. The range
/// of values depends on the specific implementation of the stibilisation
/// algorithm. Default values [0-1]: 0 - the library will not compensate for
/// constant camera motion, video will not be stabilized, 1 - no smoothing
/// of constant camera motion (the library will compensate for the current
/// picture drift completely without considering constant motion).
float xFilterCoeff{0.9f};
/// Vertical smoothing coefficient of constant camera movement. The range
/// of values depends on the specific implementation of the stibilisation
/// algorithm. Default values [0-1]: 0 - the library will not compensate for
/// constant camera motion, video will not be stabilized, 1 - no smoothing
/// of constant camera motion (the library will compensate for the current
/// picture drift completely without considering constant motion).
float yFilterCoeff{0.9f};
/// Rotational smoothing coefficient of constant camera movement. The range
/// of values depends on the specific implementation of the stibilisation
/// algorithm. Default values [0-1]: 0 - the library will not compensate for
/// constant camera motion, video will not be stabilized, 1 - no smoothing
/// of constant camera motion (the library will compensate for the current
/// picture drift completely without considering constant motion).
float aFilterCoeff{0.9f};
/// Enable/disable stabilisation.
bool enable{true};
/// Enable/disable transparent borders.
bool transparentBorder{true};
/// Constant horizontal image offset in pixels. The library should add this
/// offset to each processed video frame.
int constXOffset{0};
/// Constant vertical image offset in pixels. The library should add this
/// offset to each processed video frame.
int constYOffset{0};
/// Constant rotational angle in radians. The library should add this
/// offset to each processed video frame.
float constAOffset{0.0f};
/// Instant (for one frame) horizontal image offset in pixels. The library
/// should add this offset to next processed video frame.
int instantXOffset{0};
/// Instant (for one frame) vertical image offset in pixels. The library
/// should add this offset to next processed video frame.
int instantYOffset{0};
/// Instant (for one frame) rotational angle in radians. The library
/// should add this offset to next processed video frame.

```

```

float instantAOffset{0.0f};
/// Algorithm type. Default values:
/// 0 - 2D type 1. Stabilisation only on horizonatal and vertical.
/// 1 - 2D type 2. Stabilisation only on horizonatal and vertical.
/// 2 - 3D. Stabilisation on horizontal and vertical + rotation.
/// Particular implementation can have unique values.
int type{2};
/// Cat frequency, Hz. Stabiliser will block vibrations with frequency
/// > cutFrequencyHz.
float cutFrequencyHz{2.0f};
/// Frames per second of input video.
float fps{30.0f};
/// Processing time, mks. Processing time for last video frame.
int processingTimeMks{0};
/// Logging mode. Values: 0 - Disable, 1 - Only file, 2 - Only terminal,
/// 3 - File and terminal.
int logMod{0};

JSON_READABLE(VStabiliserParams, scaleFactor, xOffsetLimit, yOffsetLimit,
              aoffsetLimit, xFilterCoeff, yFilterCoeff, aFilterCoeff,
              enable, transparentBorder, constXOffset, constYOffset,
              constAOffset, type, cutFrequencyHz, fps, logMod);

/**
 * @brief operator =
 * @param src Source object.
 * @return VStabiliserParams obect.
 */
VStabiliserParams& operator= (const VStabiliserParams& src);

/**
 * @brief Encode params.
 * @param data Pointer to data buffer.
 * @param size Size of data.
 * @param mask Pointer to params mask to include in data.
 */
void encode(uint8_t* data, int& size, VStabiliserParamsMask* mask = nullptr);

/**
 * @brief Decode params.
 * @param data Pointer to data.
 * @return TRUE is params decoded or FALSE if not.
 */
bool decode(uint8_t* data);
};

```

**Table 4** - VStabiliserParams class fields description is equivalent to **VStabiliserParam** enum description.

Field	type	Description
scaleFactor	int	Scale factor. Value depends on implementation. Default: If 1 the library will process original frame size, if 2 the library will scale original frane size by 2, if 3 - by 3.

Field	type	Description
xOffsetLimit	int	Maximum horizontal image shift in pixels per video frame. If image shift bigger than this limit the library should compensate only xOffsetLimit shift.
yOffsetLimit	int	Maximum vertical image shift in pixels per video frame. If image shift bigger than this limit the library should compensate only yOffsetLimit shift.
aOffsetLimit	float	Maximum rotational image angle in radians per video frame. If image absolute rotational angle bigger than this limit the library should compensate only aOffsetLimit angle.
xFilterCoeff	float	Horizontal smoothing coefficient of constant camera movement. The range of values depends on the specific implementation of the stabilisation algorithm. Default values [0-1]: 0 - the library will not compensate for constant camera motion, video will not be stabilized, 1 - no smoothing of constant camera motion (the library will compensate for the current picture drift completely without considering constant motion).
yFilterCoeff	float	Vertical smoothing coefficient of constant camera movement. The range of values depends on the specific implementation of the stabilisation algorithm. Default values [0-1]: 0 - the library will not compensate for constant camera motion, video will not be stabilized, 1 - no smoothing of constant camera motion (the library will compensate for the current picture drift completely without considering constant motion).
aFilterCoeff	float	Rotational smoothing coefficient of constant camera movement. The range of values depends on the specific implementation of the stabilisation algorithm. Default values [0-1]: 0 - the library will not compensate for constant camera motion, video will not be stabilized, 1 - no smoothing of constant camera motion (the library will compensate for the current picture drift completely without considering constant motion).
enable	bool	Enable/disable stabilisation.
transparentBorder	bool	Enable/disable transparent borders.
constXOffset	int	Constant horizontal image offset in pixels. The library should add this offset to each processed video frame.
constYOffset	int	Constant vertical image offset in pixels. The library should add this offset to each processed video frame.
constAOffset	float	Constant rotational angle in radians. The library should add this offset to each processed video frame.
instantXOffset	int	Instant (for one frame) horizontal image offset in pixels. The library should add this offset to next processed video frame.



Field	type	Description
instantYOffset	int	Instant (for one frame) vertical image offset in pixels. The library should add this offset to next processed video frame.
instantAOffset	int	Instant (for one frame) rotational angle in radians. The library should add this offset to next processed video frame.
type	int	Algorithm type. Default values: 0 - 2D type 1. Stabilisation only on horizontal and vertical. 1 - 2D type 2. Stabilisation only on horizontal and vertical. 2 - 3D. Stabilisation on horizontal and vertical + rotation. Particular implementation can have unique values.
cutFrequencyHz	float	Cut frequency, Hz. Stabiliser will block vibrations with frequency > cutFrequencyHz.
fps	float	Frames per second of input video.
processingTimeMks (read only parameter)	int	Processing time, mks. Processing time for last video frame.
logMod	int	Logging mode. Values: 0 - Disable, 1 - Only file, 2 - Only terminal, 3 - File and terminal.

**None:** *VStabiliserParams* class fields listed in Table 4 **must** reflect params set/get by methods *setParam(...)* and *getParam(...)*.

## Serialize video stabiliser params

**VStabiliserParams** class provides method **encode(...)** to serialize video stabiliser params (fields of *VStabiliserParams* class, see Table 4). Serialization of params necessary in case when you need to send params via communication channels. Method provide options to exclude particular parameters from serialization. To do this method inserts binary mask (3 bytes) where each bit represents particular parameter and **decode(...)** method recognizes it. Method declaration:

```
void encode(uint8_t* data, int& size, VStabiliserParamsMask* mask = nullptr);
```

Parameter	Value
data	Pointer to data buffer.
size	Size of encoded data.
mask	Parameters mask - pointer to <b>VStabiliserParamsMask</b> structure. <b>VStabiliserParamsMask</b> (declared in <i>VStabiliser.h</i> file) determines flags for each field (parameter) declared in <b>VStabiliserParams</b> class. If the user wants to exclude any parameters from serialization, he can put a pointer to the mask. If the user wants to exclude a particular parameter from serialization, he should set the corresponding flag in the <i>VStabiliserParamsMask</i> structure.

**VStabiliserParamsMask** structure declaration:

```

typedef struct VStabiliserParamsMask
{
    bool scaleFactor{true};
    bool xOffsetLimit{true};
    bool yOffsetLimit{true};
    bool aOffsetLimit{true};
    bool xFilterCoeff{true};
    bool yFilterCoeff{true};
    bool aFilterCoeff{true};
    bool enable{true};
    bool transparentBorder{true};
    bool constXOffset{true};
    bool constYOffset{true};
    bool constAOffset{true};
    bool instantXOffset{false};
    bool instantYOffset{false};
    bool instantAOffset{false};
    bool type{true};
    bool cutFrequencyHz{true};
    bool fps{true};
    bool processingTimeMks{true};
    bool logMod{true};
} VStabiliserParamsMask;

```

Example without parameters mask:

```

// Encode data.
VStabiliserParams in;
uint8_t data[1024];
int size = 0;
in.encode(data, size);
cout << "Encoded data size: " << size << " bytes" << endl;

```

Example without parameters mask:

```

// Prepare mask.
VStabiliserParamsMask mask;
mask.scaleFactor = true; // Exclude scaleFactor. Others by default.

// Encode data.
VStabiliserParams in;
uint8_t data[1024];
int size = 0;
in.encode(data, size, &mask);
cout << "Encoded data size: " << size << " bytes" << endl;

```

## Deserialize video stabiliser params

**VStabiliserParams** class provides method **decode(...)** to deserialize params (fields of VStabiliserParams class, see Table 4). Deserialization of params necessary in case when you need to receive params via communication channels. Method automatically recognizes which parameters were serialized by **encode(...)** method. Method declaration:

```
bool decode(uint8_t* data);
```

Parameter	Value
data	Pointer to data buffer.

**Returns:** TRUE if data decoded (deserialized) or FALSE if not.

Example:

```
// Encode data.
VStabiliserParams in;
uint8_t data[1024];
int size = 0;
in.encode(data, size);
cout << "Encoded data size: " << size << " bytes" << endl;

// Decode data.
VStabiliserParams out;
if (!out.decode(data))
    cout << "Can't decode data" << endl;
```

## Read params from JSON file and write to JSON file

**VStabiliser** interface class library depends on **ConfigReader** library which provides method to read params from JSON file and to write params to JSON file. Example of writing and reading params to JSON file:

```
// Prepare random params.
VStabiliserParams in;
in.scaleFactor = rand() % 255;
in.xOffsetLimit = rand() % 255;
in.yOffsetLimit = rand() % 255;
in.aOffsetLimit = static_cast<float>(rand() % 255);

// Write params to file.
cr::utils::ConfigReader inConfig;
inConfig.set(in, "VStabiliserParams");
inConfig.writeToFile("TestVStabiliserParams.json");

// Read params from file.
cr::utils::ConfigReader outConfig;
if (!outConfig.readFromFile("TestVStabiliserParams.json"))
{
    cout << "Can't open config file" << endl;
```

```

        return false;
    }

    VStabiliserParams out;
    if(!outConfig.get(out, "VStabiliserParams"))
    {
        cout << "Can't read params from file" << endl;
        return false;
    }

```

**TestVStabiliserParams.json** will look like:

```

{
  "VStabiliserParams": {
    "aFilterCoeff": 5.0,
    "aOffsetLimit": 13.0,
    "constAOffset": 16.0,
    "constXOffset": 57,
    "constYOffset": 33,
    "cutFrequencyHz": 161.0,
    "enable": false,
    "fps": 90.0,
    "logMod": 99,
    "scaleFactor": 53,
    "transparentBorder": true,
    "type": 208,
    "xFilterCoeff": 76.0,
    "xOffsetLimit": 51,
    "yFilterCoeff": 230.0,
    "yOffsetLimit": 244
  }
}

```

## Build and connect to your project

Typical commands to build **VStabiliser** library:

```

git clone https://github.com/ConstantRobotics-Ltd/VStabiliser.git
cd VStabiliser
git submodule update --init --recursive
mkdir build
cd build
cmake ..
make

```

If you want connect **VStabiliser** library to your CMake project as source code you can make follow. For example, if your repository has structure:

```
CMakeLists.txt
src
  CMakeList.txt
  yourLib.h
  yourLib.cpp
```

You can add repository **VStabiliser** as submodule by commands:

```
cd <your repository folder>
git submodule add https://github.com/ConstantRobotics-Ltd/VStabiliser.git
3rdparty/VStabiliser
git submodule update --init --recursive
```

In you repository folder will be created folder **3rdparty/VStabiliser** which contains files of **VStabiliser** repository with subrepositories **Frame** and **ConfigReader**. New structure of your repository:

```
CMakeLists.txt
src
  CMakeList.txt
  yourLib.h
  yourLib.cpp
3rdparty
  vstabiliser
```

Create CMakeLists.txt file in **3rdparty** folder. CMakeLists.txt should contain:

```
cmake_minimum_required(VERSION 3.13)

#####
## 3RD-PARTY
## dependencies for the project
#####
project(3rdparty LANGUAGES CXX)

#####
## SETTINGS
## basic 3rd-party settings before use
#####
# To inherit the top-level architecture when the project is used as a submodule.
SET(PARENT ${PARENT}_YOUR_PROJECT_3RDPARTY)
# Disable self-overwriting of parameters inside included subdirectories.
SET(${PARENT}_SUBMODULE_CACHE_OVERWRITE OFF CACHE BOOL "" FORCE)

#####
## CONFIGURATION
## 3rd-party submodules configuration
#####
SET(${PARENT}_SUBMODULE_VSTABILISER ON CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VSTABILISER)
  SET(${PARENT}_VSTABILISER ON CACHE BOOL "" FORCE)
  SET(${PARENT}_VSTABILISER_TEST OFF CACHE BOOL "" FORCE)
endif()
```

```
#####
## INCLUDING SUBDIRECTORIES
## Adding subdirectories according to the 3rd-party configuration
#####
if (${PARENT}_SUBMODULE_VSTABILISER)
    add_subdirectory(Vstabiliser)
endif()
```

File **3rdparty/CMakeLists.txt** adds folder **Vstabiliser** to your project and excludes test application (Vstabiliser class test applications) from compiling. Your repository new structure will be:

```
CMakeLists.txt
src
  CMakeList.txt
  yourLib.h
  yourLib.cpp
3rdparty
  CMakeLists.txt
  vstabiliser
```

Next you need include folder 3rdparty in main **CMakeLists.txt** file of your repository. Add string at the end of your main **CMakeLists.txt**:

```
add_subdirectory(3rdparty)
```

Next you have to include Vstabiliser library in your **src/CMakeLists.txt** file:

```
target_link_libraries(${PROJECT_NAME} vstabiliser)
```

Done!