BoxingLibrary

Generated by Doxygen 1.6.1

Wed Aug 29 13:27:20 2018

Contents

1 Boxing Library

Functions for decoding analog and digital data. The boxing library has functions for coding (boxing) and decoding (unboxing) digital and analog data.

The Boxer (boxing/boxer.h) is the top level API for boxing data. It takes a byte array as input and converts it to raw-images. The layout of the raw-image and the coding of the digital data is defined by the Configuration.

The Unboxer (boxing/unboxer.h) takes sampled input images, and decodes in two steps: extract and decode. The extract step locates the frame within the image, decodes the metadata in the bottom border of the frame, then tracks the pixels within the frame. The tracked pixels are then quantified for digital data. The decode step is different for analog and digital data. Digital decode is applying the codec defined by the Configuration. Analoge decode is applying the LUT created from the frame top border calibration bar.

1.1 Definitions

1.1.1 sampled-image

2 dimensional digitized version of image stored on analog storage medium. Must be sampled with higher resoultion than original resolution used when writing the image.

1.1.2 raw-image

Two dimensional digital image to be written on analog storage medium. The image represents a rendered 2D barcode image with a frame and a data container. The resoultion of the raw image per printed pixel is from1 to 8 bits.

1.1.3 metadata

Generic information stored in the border of the frame of a raw image. Examples can be frame number and checksums.

1.1.4 boxing

Coding analog and digital data into raw images.

1.1.5 unboxing

Decoding sampled images and restoring the original content written to the raw image.

1.1.6 boxing-format

Parameters describing the raw image geometry and the methods used for coding the digital data into the frame.

1.1.7 extract-step

First step in unboxing a sampled-image. The process consists of locating the frame within the image, then decoding the metadata in the bottom border of the frame, then tracking the pixels within the frame. The tracked pixels are then quantified for digital data.

1.1.8 decode-step

Second step in unboxing a sampled-image. The step is different for analog and digital data. Digital decode is applying the codec defined by the boxing_format. Analoge decode is applying the LUT created from the frame top border calibration bar.

1.2 Sample Applications

1.2.1 Boxer

Command line application for coding digital data, see tests\boxer\main.c

1.2.2 Unboxer

Command line application for decoding digital data, see tests\unboxer\main.c

2 Todo List

Global boxing_config_set_property_uint consider optimizing

3 Module Documentation

3.1 Boxer

Box data into frames.

The boxer is responsible for coding digital and analoge data into frames. This provides the top level API for coding data.

Data Structures

- struct boxing_boxer_parameters_s

  Boxing parameters.

Functions

- boxing_boxer_parameters * boxing_boxer_parameters_create (const boxing_config *config)

  Create boxer parameters.

- void boxing_boxer_parameters_free (boxing_boxer_parameters *params)

Free boxer parameters.

- void boxing_boxer_parameters_init (boxing_boxer_parameters *params, const boxing_config *config)

  Initialize boxer parameters.

- boxing_boxer * boxing_boxer_create (boxing_boxer_parameters *params)

  Create boxer.

- void boxing_boxer_free (boxing_boxer *boxer)

  Free boxer.

3.1.1 Function Documentation

3.1.1.1 boxing_boxer* boxing_boxer_create (boxing_boxer_parameters * *params*)

Create boxer with given parameters.

Parameters:

params Boxing configuration.

3.1.1.2 void boxing_boxer_free (boxing_boxer * *boxer*)

Free data owned by boxer structure, but not structure itself.

Parameters:

boxer Boxer instance.

3.1.1.3 boxing_boxer_parameters* boxing_boxer_parameters_create (const boxing_config * *config*)

Create boxer parameters.

Parameters:

config Boxing configuration.

3.1.1.4 void boxing_boxer_parameters_free (boxing_boxer_parameters * *params*)

Free boxer parameters.

Parameters:

params Boxer parameters instance.

3.1.1.5 void boxing_boxer_parameters_init (boxing_boxer_parameters * *params*, const boxing_config * *config*)

Initialize boxer parameters.

Parameters:

params Boxing parameters.

config Boxing configuration.

3.2 Unboxing

Modules

- Boxer

  Box data into frames.
  The boxer is responsible for coding digital and analoge data into frames. This provides the top level API for coding data.

- Unboxer

  Unbox sampled frames.
  The unboxer is responsible for decoding digital and analoge data from sampled frames. This provides the top level API for decoding frames.

- Configuration

  Frame configuration.
  The configuration describes the frame layout and the codecs used to code and decode the digital data. The configuration is a set of key / value strings organized into groups.

3.3 Unboxer

Unbox sampled frames.

The unboxer is responsible for decoding digital and analoge data from sampled frames. This provides the top level API for decoding frames.

Data Structures

- struct boxing_unboxer_parameters_s

  Unboxer configuration.

- struct boxing_unboxing_codec_info_s

  Codec info.

Typedefs

- typedef boxing_image8 *(* boxing_sample_cb)(void *user, const boxing_image8 *frame, const boxing_matrixf *location_matrix, DBOOL *state)

  Boxing sample callback function.

- typedef gvector *(* boxing_quantize_cb)(void *user, const boxing_image8 *image, int block_width, int block_height, int bins)

  Boxing quantize callback function.

- typedef int(* boxing_tracker_created_cb)(void *user, int *res, boxing_tracker *tracker)

  Boxing tracker created callback function.

- typedef int(* boxing_reference_bar_complete_cb)(void *user, int *res, boxing_stats_mtf *stats)

  Boxing reference bar complete callback function.

- typedef int(* boxing_metadata_complete_cb)(void *user, int *res, boxing_metadata_list *metadata)

  Boxing metadata complete callback function.

- typedef int(* boxing_content_sampled_cb)(void *user, int *res, boxing_image8 *image, void *sampler_list)

  Boxing content sampled callback function.

- typedef int(* boxing_content_quantized_cb)(void *user, int *res, char *data, int size)

  Boxing content quantized callback function.

- typedef int(* boxing_training_complete_cb)(void *user, int *res, void *training_result)

  Boxing training complete callback function.

- typedef int(* boxing_decode_step_cb)(void *user, int *res, void *data, int data_size, int step, int is_codec_errorcorrecting, int parity_size, int block_size)

  Boxing decode step callback function.

- typedef int(* boxing_all_complete_cb)(void *user, int *res, boxing_stats_decode *stats)

  Unoxing complete callback function.

Enumerations

- enum boxing_unboxer_result

  Unboxing result.

- enum boxing_process_callback_result

  Processing callback result.

Functions

- boxing_unboxer * boxing_unboxer_create (boxing_unboxer_parameters *parameters)

  Create unboxer.

- void boxing_unboxer_free (boxing_unboxer *unboxer)

  Free unboxer data.

- void boxing_unboxer_set_raw_input (boxing_unboxer *unboxer, int is_raw)
  Enable/disable RAW input images.
- int boxing_unboxer_is_raw_input (boxing_unboxer *unboxer)
  Get RAW mode.
- void boxing_unboxer_codec_info (const boxing_unboxer *unboxer, int step, boxing_codec_info *info)
  Get the codec information.
- size_t boxing_unboxer_decoding_steps (const boxing_unboxer *unboxer)
  Get the number of decoding steps.
- void boxing_unboxer_reset (const boxing_unboxer *unboxer)
  Reset the unboxer.
- enum boxing_unboxer_result boxing_unboxer_unbox_extract_container (gvector *data, boxing_metadata_list *metadata, boxing_image8 *image, boxing_unboxer *unboxer, void *user_data)
  Extract data container.
- int boxing_unboxer_decode (boxing_unboxer *unboxer, gvector *data, boxing_metadata_list *metadata, boxing_stats_decode *decode_stats, unsigned int step, void *user_data)
  Decode data.
- enum boxing_unboxer_result boxing_unboxer_unbox (gvector *data, boxing_metadata_list *metadata, boxing_image8 *image, boxing_unboxer *unboxer, int *extract_result, void *user_data)
  Decode image.
- boxing_codecdispatcher * boxing_unboxer_dispatcher (boxing_unboxer *unboxer, const char *coding_scheme)
  Dispatcher function.
- void boxing_unboxer_parameters_init (boxing_unboxer_parameters *parameters)
  Initialize unboxer parameters.
- void boxing_unboxer_parameters_free (boxing_unboxer_parameters *parameters)
  Free unboxing parameters data.

3.3.1 Typedef Documentation

3.3.1.1 int(* boxing_all_complete_cb)(void *user, int *res, boxing_stats_decode *stats)

Parameters:

↔ ***user*** User data.
← ***res*** Result.
← ***stats*** Decode statistic.

Last callback from unboxing.

3.3.1.2 int(∗ boxing_content_quantized_cb)(void ∗user, int ∗res, char ∗data, int size)

Parameters:

↔ ***user*** User data.

← ***res*** Result.

← ***data*** Data.

← ***size*** Size.

Called when iamge quantize (conversion from sampled pixels to symbols) is complete.

3.3.1.3 int(∗ boxing_content_sampled_cb)(void ∗user, int ∗res, boxing_image8 ∗image, void ∗sampler_list)

Parameters:

↔ ***user*** User data.

← ***res*** Result.

← ***image*** Image.

← ***sampler_list*** Sampler list.

Called when image content is sampled (location of each content pixel is found).

3.3.1.4 int(∗ boxing_decode_step_cb)(void ∗user, int ∗res, void ∗data, int data_size, int step, int is_codec_errorcorrecting, int parity_size, int block_size)

Parameters:

↔ ***user*** User data.

← ***res*** Result.

← ***data*** Data.

← ***data_size*** Data size.

← ***step*** Current step.

← ***is_codec_errorcorrecting*** Codec error correcting sign.

← ***parity_size*** Parity size.

← ***block_size*** Block size.

Codec decode step complete callback.

3.3.1.5 int(∗ boxing_metadata_complete_cb)(void ∗user, int ∗res, boxing_metadata_list ∗meta_data)

Parameters:

↔ ***user*** User data.
← ***res*** Result.
← ***meta_data*** Decoded metadata items.

Called when metadata decoding is complete.

3.3.1.6 gvector ∗(∗ boxing_quantize_cb)(void ∗user, const boxing_image8 ∗image, int block_width, int block_height, int bins)

Parameters:

↔ ***user*** User data.
← ***image*** Image.
← ***block_width*** Block width.
← ***block_height*** Block height.
← ***bins*** Bins.

Called before quantizing starts.

3.3.1.7 int(∗ boxing_reference_bar_complete_cb)(void ∗user, int ∗res, boxing_stats_mtf ∗stats)

Parameters:

↔ ***user*** User data.
← ***res*** Result.
← ***stats*** MTF statistics.

Called when reference bar tracking is complete.

3.3.1.8 boxing_image8 ∗(∗ boxing_sample_cb)(void ∗user, const boxing_image8 ∗frame, const boxing_matrixf ∗location_matrix, DBOOL ∗state)

Parameters:

↔ ***user*** User data.
← ***frame*** Frame.
← ***location_matrix*** Location matrix.
← ***state*** State.

Called when image sampling is complete..

3.3.1.9 int(∗ boxing_tracker_created_cb)(void ∗user, int ∗res, boxing_tracker ∗tracker)

Parameters:

↔ ***user*** User data.

← ***res*** Result.

← ***tracker*** Tracker.

Called after the frame tracker object is created. Allows for implementation spesific modifications to the tracker.

3.3.1.10 int(∗ boxing_training_complete_cb)(void ∗user, int ∗res, void ∗training_result)

Parameters:

↔ ***user*** User data.

← ***res*** Result.

← ***training_result*** Training result.

Called when boxing training is complete.

3.3.2 Enumeration Type Documentation

3.3.2.1 enum boxing_process_callback_result

unboxer.h

Parameters:

BOXING_PROCESS_CALLBACK_OK (0) Process callback OK.

BOXING_PROCESS_CALLBACK_ABORT (1) Process callback abort.

A callback should return BOXING_PROCESS_CALLBACK_ABORT if the unbox process should be aborted or BOXING_PROCESS_CALLBACK_OK is the unbox process should continue.

3.3.2.2 enum boxing_unboxer_result

unboxer.h

Parameters:

BOXING_UNBOXER_OK (0) Unboxing OK.

BOXING_UNBOXER_METADATA_ERROR (1) Metadata error.

BOXING_UNBOXER_BORDER_TRACKING_ERROR (2) Border tracking error.

BOXING_UNBOXER_DATA_DECODE_ERROR (3) Data decode error.

BOXING_UNBOXER_CRC_MISMATCH_ERROR (4) CRC mismatch error.

BOXING_UNBOXER_CONFIG_ERROR (5) Configuration error.

BOXING_UNBOXER_PROCESS_CALLBACK_ABORT (6) Process callback abort.

BOXING_UNBOXER_INPUT_DATA_ERROR (7) Input data error.

Unboxer result codes.

3.3.3 Function Documentation

3.3.3.1 void boxing_unboxer_codec_info (const boxing_unboxer * *unboxer*, int *step*, boxing_codec_info * *info*)

Get the information of the current codec from the unboxer instance.

Parameters:

← ***unboxer*** Unboxer instance.

← ***step*** Step value.

→ ***info*** Codec information.

3.3.3.2 boxing_unboxer* boxing_unboxer_create (boxing_unboxer_parameters * *parameters*)

Create unboxer with given parameters.

Parameters:

parameters Unboxing configuration.

Returns:

Unboxer instance or NULL on error.

3.3.3.3 int boxing_unboxer_decode (boxing_unboxer * *unboxer*, gvector * *data*, boxing_metadata_list * *metadata*, boxing_stats_decode * *decode_stats*, unsigned int *step*, void * *user_data*)

Unbox metadata and data from quantized data container.

Parameters:

← ***unboxer*** Unboxer structure

↔ ***data*** In: Quantized data container. Out: Decoded data.

← ***metadata*** Decoded metadata

→ ***decode_stats*** Statistic of the decode process

← ***step*** Decoding step

← ***user_data*** User data.

Returns:

Unboxing status code

3.3.3.4 size_t boxing_unboxer_decoding_steps (const boxing_unboxer * *unboxer*)

Get the number of unboxer decoding steps.

Parameters:

← ***unboxer*** Unboxer instance.

Returns:

number of decoding steps or 0 on error.

3.3.3.5 boxing_codecdispatcher* boxing_unboxer_dispatcher (boxing_unboxer * *unboxer*, const char * *coding_scheme*)

Get the unboxer dispatcher. The dispatcher is responsible for calling the codecs in the codec chain.

Parameters:

← ***unboxer*** Unboxer.

← ***coding_scheme*** Coding scheme.

Returns:

boxing_codecdispatcher instance.

3.3.3.6 void boxing_unboxer_free (boxing_unboxer * *unboxer*)

Free data owned by unboxer structure, but not structure itself.

Parameters:

unboxer Unboxer instance.

3.3.3.7 int boxing_unboxer_is_raw_input (boxing_unboxer * *unboxer*)

Return is_raw parameter.

Parameters:

← ***unboxer*** Unboxer instance.

Returns:

is_raw value.

3.3.3.8 void boxing_unboxer_parameters_free (boxing_unboxer_parameters * *parameters*)

Free data owned by the parameters, but not the parameter instance itself.

Parameters:

← ***parameters*** Parameters to be freed.

3.3.3.9 void boxing_unboxer_parameters_init (boxing_unboxer_parameters * *parameters*)

Initialize unboxer parameters to default values.

Parameters:

← ***parameters*** Parameters to be initialized.

3.3.3.10 void boxing_unboxer_reset (const boxing_unboxer * *unboxer*)

Reset unboxer codec and metadata codec.

Parameters:

← ***unboxer*** Unboxer instance.

3.3.3.11 void boxing_unboxer_set_raw_input (boxing_unboxer * *unboxer*, int *is_raw*)

Set if input images should be interpreted as RAW images.

Parameters:

← ***unboxer*** Unboxer instance.

← ***is_raw*** false if 0, else true

3.3.3.12 enum boxing_unboxer_result boxing_unboxer_unbox (gvector * *data*, boxing_metadata_list * *metadata*, boxing_image8 * *image*, boxing_unboxer * *unboxer*, int * *extract_result*, void * *user_data*)

Decode image and return data and metadata on success.

Parameters:

→ ***data*** Decoded data.

→ ***metadata*** Decoded metadata.

← ***image*** Input image.

← ***unboxer*** Unboxer structure.

→ ***extract_result*** Result from data extraction phase of unboxing.

← ***user_data*** User data.

Returns:

Unboxing status code.

3.3.3.13 enum boxing_unboxer_result boxing_unboxer_unbox_extract_container (gvector * *data*, boxing_metadata_list * *metadata*, boxing_image8 * *image*, boxing_unboxer * *unboxer*, void * *user_data*)

Unbox metadata and extract a raw quantized copy of the data container.

Parameters:

→ ***data*** Quantized data container.

→ ***metadata*** Decoded metadata.

← ***image*** Image to be decoded.

← ***unboxer*** Unboxer structure.

← ***user_data*** User data.

Returns:

Unboxing status code

3.4 Configuration

Frame configuration.

The configuration describes the frame layout and the codecs used to code and decode the digital data. The configuration is a set of key / value strings organized into groups.

Data Structures

- struct boxing_config_s

  Frame configuration.

Functions

- boxing_config * boxing_config_create ()

  Create config object.

- void boxing_config_free (boxing_config *config)

  Frees occupied memory of boxing_config structure.

- boxing_config * boxing_config_clone (const boxing_config *config)

  Clone boxing_config structure.

- DBOOL boxing_config_is_equal (const boxing_config *a, const boxing_config *b)

  Function checks two instances of the boxing_config structures on the identity.

- boxing_config * boxing_config_instance ()

  Get global config instance.

- void boxing_config_set_property (boxing_config *config, const char *group, const char *key, const char *value)

Set string property.

- void boxing_config_set_property_uint (boxing_config *config, const char *name, const char *key, unsigned int value)

  Set unsinged int property.

- void boxing_config_properties (const boxing_config *config, const char *name, const GHashTable **properties)

  Get all group properties.

- const char * boxing_config_property (const boxing_config *config, const char *name, const char *key)

  Get string property.

- int boxing_config_property_int (const boxing_config *config, const char *name, const char *key)

  Get integer property.

- unsigned int boxing_config_property_uint (const boxing_config *config, const char *name, const char *key)

  Get unsigned integer property.

- boxing_pointi boxing_config_property_pointi (const boxing_config *config, const char *name, const char *key, DBOOL *found)

  Get integer point property.

- boxing_pointf boxing_config_property_pointf (const boxing_config *config, const char *name, const char *key, DBOOL *was_found)

  Get float point property.

- DBOOL boxing_config_is_set (const boxing_config *config, const char *name, const char *key)

  Check if property exists.

- gvector * boxing_config_parse_list_properties (const boxing_config *config, const char *name, const char *key)

  Get comma separated property value.

3.4.1 Function Documentation

3.4.1.1 boxing_config* boxing_config_clone (const boxing_config * *config*)

Creates a copy of input boxing_config structure and returns it. If boxing config pointer is NULL function returns NULL.

Parameters:

← ***config*** boxing_config pointer.

Returns:

new copy of boxing_config structure or NULL.

3.4.1.2 boxing_config* boxing_config_create ()

Allocate memory for the boxing_config type and initializes internal hash tables.

Returns:

New config instance.

3.4.1.3 void boxing_config_free (boxing_config * *config*)

Frees occupied memory of all internal hash tables.

Parameters:

← ***config*** boxing_config pointer.

3.4.1.4 boxing_config* boxing_config_instance ()

Ig global configuration is not initialized, than initializes internal hash tables and set is_instance_initialized variable.

Note:

Local config objects can be created using boxing_config_create.

Returns:

pointer to the global config instance.

3.4.1.5 DBOOL boxing_config_is_equal (const boxing_config * *a*, const boxing_config * *b*)

Function checks two instances of the boxing_config structures on the identity. Return DTRUE if groups (data) and aliaseses (aliases) are equal.

Parameters:

← ***a*** Pointer to the first instance of the boxing_config structure.

← ***b*** Pointer to the second instance of the boxing_config structure.

Returns:

sign of identity of the input structures.

3.4.1.6 DBOOL boxing_config_is_set (const boxing_config * *config*, const char * *name*, const char * *key*)

Check if property exists. Return DTRUE if required property exist.

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Name.

← ***key*** Key.

Returns:

DTRUE if property exist.

3.4.1.7 gvector* boxing_config_parse_list_properties (const boxing_config * *config*, const char * *name*, const char * *key*)

Get comma separated property value

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Name.

← ***key*** Key.

Returns:

Vector of values.

3.4.1.8 void boxing_config_properties (const boxing_config * *config*, const char * *name*, const GHashTable ** *properties*)

Sets properties pointer.

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Property name.

→ ***properties*** Properties hash table.

3.4.1.9 const char* boxing_config_property (const boxing_config * *config*, const char * *name*, const char * *key*)

Get string property.

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Name.

← ***key*** Key.

Returns:

required string property or NULL.

3.4.1.10 int boxing_config_property_int (const boxing_config * *config*, const char * *name*, const char * *key*)

Get integer property.

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Name.

← ***key*** Key.

Returns:

required integer property or NULL.

3.4.1.11 boxing_pointf boxing_config_property_pointf (const boxing_config * *config*, const char * *name*, const char * *key*, DBOOL * *was_found*)

Get float point property.

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Name.

← ***key*** Key.

→ ***was_found*** A sign of the success of the search for the required property.

Returns:

required float point property or NULL.

3.4.1.12 boxing_pointi boxing_config_property_pointi (const boxing_config * *config*, const char * *name*, const char * *key*, DBOOL * *found*)

Get integer point property.

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Name.

← ***key*** Key.

→ ***found*** A sign of the success of the search for the required property.

Returns:

required integer point property or NULL.

3.4.1.13 unsigned int boxing_config_property_uint (const boxing_config * *config*, const char * *name*, const char * *key*)

Get unsigned integer property.

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Name.

← ***key*** Key.

Returns:

required unsigned integer property or NULL.

3.4.1.14 void boxing_config_set_property (boxing_config * *config*, const char * *group*, const char * *key*, const char * *value*)

The function sets a string property. If the property group does not exist, a new group is created. If the property does not exist, a new property is added. If the property exists, its value is replaced by the new value.

Parameters:

← ***config*** boxing_config pointer.

← ***group*** Group name.

← ***key*** Property key.

← ***value*** New property value.

3.4.1.15 void boxing_config_set_property_uint (boxing_config * *config*, const char * *name*, const char * *key*, unsigned int *value*)

The function sets a new unsigned integer property. If this property does not exist, a new one is added. If this property exists, it is replaced by a new value.

Parameters:

← ***config*** Pointer to the boxing_config structure.

← ***name*** Property name.

← ***key*** Property key.

← ***value*** New unsigned integer property value.

Todo

consider optimizing

4 Data Structure Documentation

4.1 boxing_boxer_parameters_s Struct Reference

Boxing parameters.

4.1.1 Detailed Description

Parameters:

codec_cb Codec callback functions.

format Boxing format.

Boxing input parameters.

4.2 boxing_config_s Struct Reference

Frame configuration.

4.2.1 Detailed Description

Parameters:

groups Hash map of configuration groups.

aliases Group aliases.

The config object has a list of property groups. Each property group has a set of parameters (key / value pairs) where the key must be unique for the group.

A config property value is accessed using the group and property name (key).

The alias consept allows to have alternative names for a group. The goal is to remove this feature in a future release.

4.3 boxing_unboxer_parameters_s Struct Reference

Unboxer configuration.

4.3.1 Detailed Description

Parameters:

is_raw True if input data is in RAW format.

codec_cb Codec callback function.

format Boxing format, defines frame geometry and codecs.

training_mode True if unboxer should be run in training mode.

training_mode_reference Original RAW frame.

training_result Output from training mode.

pre_filter 5x5 sharpness filter that will be executed on images to be unboxed. Default filter function is used if set to NULL. Provide a direct assignemement function (i.e. destination = source) if you want to skip filtering.

decoding_filters Codec decoding functions.

sample_contents Boxing sample function.

quantize_contents Boxing quantize function.

on_tracker_created Boxing tracker created callback function.

on_content_sampled Boxing content sampled callback function.

on_content_quantized Boxing content quantized callback function.

on_metadata_complete Boxing metadata complete callback function.

on_reference_bar_complete Boxing reference bar complete callback function.

on_corner_mark_complete Boxing corner mark complete callback function.

on_training_complete Boxing training complete callback function.

on_decode_step Boxing decode step callback function.

on_all_complete Boxing all complete callback function.

orig_image Original image.

Configure unboxer. Note that the callback interface is only avaliable if BOXINGLIB_CALLBACK is defined.

Note 1: Training mode is an experimental feature for improving the unboxer by unboxing a known frame and using it's characteristics when decoding the rest of the frames. Could be removed in future versions in the library. Note 2: Pre-filtering of the image is really the responsibility of the reading device. This function will be removed in future versions of the library.

4.4 boxing_unboxing_codec_info_s Struct Reference

Codec info.

4.4.1 Detailed Description

Parameters:

name Name of the codec.

reentrant Is the codec reentrant.

Codec information.

Index