libgac

v0.1.0

Generated by Doxygen 1.8.17

Chapter 1

GAC - Gaze Analysis C Library

This is a pure C library to perform basic gaze analysis.

Features:

- Sample filtering with moving average
- Sample gap fill-in through linear interpolation (lerp)
- Fixation detection with I-DT algorithm
- Saccade detection with I-VT algorithm

Quick Start

Initialise the gaze analysis handler:

```
gac_t h;
gac_init( &h, NULL );
```

To parse gaze data for fixations and saccades, for each new sample do the following:

```
gac_sample_window_update( &h, sample.origin.x, sample.origin.y, sample.oridin.z,
        sample.point.x, sample.point.y, sample.point.z, sample.timestamp );
// check for fixation
res = gac_sample_window_fixation_filter( &h, &fixation );
if( res == true )
{
    // new fixation deteced
}
// check for saccade
res = gac_sample_window_saccade_filter( &h, &saccade );
if( res == true )
{
    // new saccade detected
}
```

At the end, destroy the gaze analysis handler:

```
gac_destroy( &h );
```

Building the library

In order to build the library the following packages are required:

```
sudo apt install build-essential
sudo apt install autoconf autogen libtool
```

To build the library use the command

```
make
```

To run tests use

```
make test
```

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 gac_filter_fixation_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- double normalized_dispersion_threshold
- double duration_threshold
- bool is_collecting
- gac_queue_t window
- double duration
- vec3 point

4.1.1 Detailed Description

The fixation filter structure holding filter parameters.

gac_filter_fixation_s

4.1.2 Field Documentation

4.1.2.1 duration

```
double gac_filter_fixation_t::duration
```

The fixation duration

4.1.2.2 duration_threshold

```
double gac_filter_fixation_t::duration_threshold
```

The duration threashold

4.1.2.3 is_collecting

```
bool gac_filter_fixation_t::is_collecting
```

A flag indicating whether a fixation is ongoing

4.1.2.4 is_heap

```
bool gac_filter_fixation_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.1.2.5 normalized_dispersion_threshold

```
double gac_filter_fixation_t::normalized_dispersion_threshold
```

The pre-computed dispersion threshold at unit distance

4.1.2.6 point

```
vec3 gac_filter_fixation_t::point
```

The fixation point

4.1.2.7 window

```
gac_queue_t gac_filter_fixation_t::window
```

A pointer to the sample queue

The documentation for this struct was generated from the following file:

- include/gac.h

4.2 gac_filter_gap_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- bool is_enabled
- double max_gap_length
- double sample_period

4.2.1 Detailed Description

The gap fill-in filter structure.

gac_filter_gap_s

4.2.2 Field Documentation

4.2.2.1 is_enabled

```
bool gac_filter_gap_t::is_enabled
```

A flag indicating whether the filter is active or not

4.2.2.2 is_heap

```
bool gac_filter_gap_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.2.2.3 max_gap_length

```
double gac_filter_gap_t::max_gap_length
```

The maximal allowed gap length to be filled-in

4.2.2.4 sample_period

```
double gac_filter_gap_t::sample_period
```

The sample period to compute the number of required fill-in samples

The documentation for this struct was generated from the following file:

- include/gac.h

4.3 gac_filter_noise_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- bool is_enabled
- gac_queue_t window
- uint32_t mid
- gac_filter_noise_type_t type

4.3.1 Detailed Description

The noise filter parameters.

gac_filter_noise_s

4.3.2 Field Documentation

4.3.2.1 is_enabled

```
bool gac_filter_noise_t::is_enabled
```

A flag indicating whether the noise filter is active or not

4.3.2.2 is_heap

```
bool gac_filter_noise_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.3.2.3 mid

```
uint32_t gac_filter_noise_t::mid
```

The mid-point counter

4.3.2.4 type

```
gac_filter_noise_type_t gac_filter_noise_t::type
```

The noise filter type

4.3.2.5 window

```
gac_queue_t gac_filter_noise_t::window
```

The noise filter window

The documentation for this struct was generated from the following file:

- include/gac.h

4.4 gac_filter_parameter_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- struct {
  double max_gap_length
  double sample_period
  } gap
- struct {
  gac_filter_noise_type_t type
  uint32_t mid_idx
  } noise
- struct {
  float velocity_threshold
  } saccade
- struct {
  double duration_threshold
  float dispersion_threshold
  } fixation

4.4.1 Detailed Description

The filter parameter structure to initialise the gaze analysis handeler.

gac_filter_parameter_s

4.4.2 Field Documentation

4.4.2.1 dispersion_threshold

```
float gac_filter_parameter_t::dispersion_threshold
```

The dispersion threshold in degrees.

4.4.2.2 duration_threshold

```
double gac_filter_parameter_t::duration_threshold
```

The duration threshold in milliseconds.

4.4.2.3 fixation

```
struct { ...  } gac_filter_parameter_t::fixation
```

Fixation detection.

4.4.2.4 gap

```
struct { ...  } gac_filter_parameter_t::gap
```

The gap filter parameter

4.4.2.5 is_heap

```
bool gac_filter_parameter_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.4.2.6 max_gap_length

```
double gac_filter_parameter_t::max_gap_length
```

The maximal allowed gap length to be filled-in. Set to zero to disable gap fill-in filter.

4.4.2.7 mid_idx

```
uint32_t gac_filter_parameter_t::mid_idx
```

The mid index of the window. This is used to compute the length of the window: window_length = mid_idx $*$ 2 + 1. Set to zero to disable noise filtering.

4.4.2.8 noise

```
struct { ... } gac_filter_parameter_t::noise
```

Noise filter parameter

4.4.2.9 saccade

```
struct { ... } gac_filter_parameter_t::saccade
```

Saccade detection.

4.4.2.10 sample_period

```
double gac_filter_parameter_t::sample_period
```

The sample period to compute the number of required fill-in samples

4.4.2.11 type

```
gac_filter_noise_type_t gac_filter_parameter_t::type
```

The noise filter type.

4.4.2.12 velocity_threshold

```
float gac_filter_parameter_t::velocity_threshold
```

The velocity threshold in degrees per seconds.

The documentation for this struct was generated from the following file:

- include/gac.h

4.5 gac_filter_saccade_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- float velocity_threshold
- bool is_collecting
- gac_queue_t window

4.5.1 Detailed Description

The saccade filter structure holding filter parameters.

gac_filter_saccade_s

4.5.2 Field Documentation

4.5.2.1 is_collecting

```
bool gac_filter_saccade_t::is_collecting
```

A flag indicating whether a saccade is ongoing

4.5.2.2 is_heap

```
bool gac_filter_saccade_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.5.2.3 velocity_threshold

```
float gac_filter_saccade_t::velocity_threshold
```

The velocity threshold

4.5.2.4 window

```
gac_queue_t gac_filter_saccade_t::window
```

A pointer to the sample queue

The documentation for this struct was generated from the following file:

- include/gac.h

4.6 gac_fixation_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- vec3 point
- double duration
- double timestamp

4.6.1 Detailed Description

A fixation sample.

gac_fixation_s

4.6.2 Field Documentation

4.6.2.1 duration

```
double gac_fixation_t::duration
```

The fixation duration in milliseconds

4.6.2.2 is_heap

```
bool gac_fixation_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.6.2.3 point

```
vec3 gac_fixation_t::point
```

The fixation gaze point

4.6.2.4 timestamp

```
double gac_fixation_t::timestamp
```

The timestamp of the fixation start

The documentation for this struct was generated from the following file:

- include/gac.h

4.7 gac_queue_item_t Struct Reference

```
#include <gac.h>
```

Data Fields

- gac_queue_item_t ∗ next
- gac_queue_item_t ∗ prev
- void ∗ data

4.7.1 Detailed Description

A generic queue item.

gac_queue_item_s

4.7.2 Field Documentation

4.7.2.1 data

```
void* gac_queue_item_t::data
```

A pointer to the arbitrary data structure

4.7.2.2 next

```
gac_queue_item_t* gac_queue_item_t::next
```

A pointer to the next queue item (towards the head).

4.7.2.3 prev

```
gac_queue_item_t* gac_queue_item_t::prev
```

A pointer to the previous queue item (towards the tail).

The documentation for this struct was generated from the following file:

- include/gac.h

4.8 gac_queue_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- gac_queue_item_t ∗ tail
- gac_queue_item_t ∗ head
- uint32_t count
- uint32_t length
- void(∗ rm)(void ∗)

4.8.1 Detailed Description

A generic queue structure.

gac_queue_s

4.8.2 Field Documentation

4.8.2.1 count

```
uint32_t gac_queue_t::count
```

The number of occupied spaces.

4.8.2.2 head

```
gac_queue_item_t* gac_queue_t::head
```

A pointer to the tail to write to

4.8.2.3 is_heap

```
bool gac_queue_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.8.2.4 length

```
uint32_t gac_queue_t::length
```

The number of total available spaces

4.8.2.5 rm

```
void( * gac_queue_t::rm) (void *)
```

The handler to remove data items

4.8.2.6 tail

```
gac_queue_item_t* gac_queue_t::tail
```

A pointer to the head of the queue to read from.

The documentation for this struct was generated from the following file:

- include/gac.h

4.9 gac_saccade_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- vec3 point_start
- vec3 point_dest
- double duration
- double timestamp

4.9.1 Detailed Description

A saccade sample.

gac_saccade_s

4.9.2 Field Documentation

4.9.2.1 duration

```
double gac_saccade_t::duration
```

The sacacde duration

4.9.2.2 is_heap

```
bool gac_saccade_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.9.2.3 point_dest

```
vec3 gac_saccade_t::point_dest
```

The end point of the saccade

4.9.2.4 point_start

```
vec3 gac_saccade_t::point_start
```

The start point of the saccade

4.9.2.5 timestamp

```
double gac_saccade_t::timestamp
```

The timestamp of the first saccade point

The documentation for this struct was generated from the following file:

- include/gac.h

4.10 gac_sample_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- vec3 point
- vec3 origin
- double timestamp

4.10.1 Detailed Description

The gaze data sample.

gac_sample_s

4.10.2 Field Documentation

4.10.2.1 is_heap

```
bool gac_sample_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.10.2.2 origin

```
vec3 gac_sample_t::origin
```

The gaze origin.

4.10.2.3 point

```
vec3 gac_sample_t::point
```

The gaze point.

4.10.2.4 timestamp

```
double gac_sample_t::timestamp
```

The sample timestamp.

The documentation for this struct was generated from the following file:

- include/gac.h

4.11 gac_t Struct Reference

```
#include <gac.h>
```

Data Fields

- bool is_heap
- gac_queue_t samples
- gac_filter_fixation_t fixation
- gac_filter_gap_t gap
- gac_filter_saccade_t saccade
- gac_filter_noise_t noise
- gac_filter_parameter_t parameter

4.11.1 Detailed Description

The gaze analysis handler structure.

gac_s

4.11.2 Field Documentation

4.11.2.1 fixation

```
gac_filter_fixation_t gac_t::fixation
```

The fixation filter structure

4.11.2.2 gap

```
gac_filter_gap_t gac_t::gap
```

The gap filter structure

4.11.2.3 is_heap

```
bool gac_t::is_heap
```

Flag to indicate whether the struct was allocated on the heap.

4.11.2.4 noise

```
gac_filter_noise_t gac_t::noise
```

The noise filter structure

4.11.2.5 parameter

```
gac_filter_parameter_t gac_t::parameter
```

The parameters passed during configuration

4.11.2.6 saccade

```
gac_filter_saccade_t gac_t::saccade
```

The saccade filetr structure

4.11.2.7 samples

```
gac_queue_t gac_t::samples
```

The sample queue

The documentation for this struct was generated from the following file:

- include/gac.h

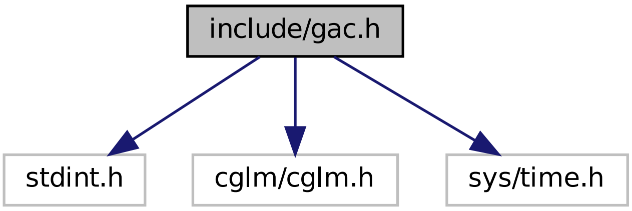
Chapter 5

File Documentation

5.1 include/gac.h File Reference

```
#include <stdint.h>
#include <cglm/cglm.h>
#include <sys/time.h>
```

Include dependency graph for gac.h:



Data Structures

- struct gac_sample_t
- struct gac_fixation_t
- struct gac_saccade_t
- struct gac_queue_item_t
- struct gac_queue_t
- struct gac_filter_fixation_t
- struct gac_filter_saccade_t
- struct gac_filter_noise_t
- struct gac_filter_gap_t
- struct gac_filter_parameter_t
- struct gac_t

Typedefs

- typedef enum gac_filter_noise_type_e gac_filter_noise_type_t
- typedef enum gac_filter_step_action_e gac_filter_step_action_t

Enumerations

- enum gac_filter_noise_type_e { GAC_FILTER_NOISE_TYPE_AVERAGE, GAC_FILTER_NOISE_TYPE_MEDIAN }
- enum gac_filter_step_action_e { **GAC_FILTER_STEP_ACTION_SHRINK**, **GAC_FILTER_STEP_ACTION_CLEAR**, **GAC_FILTER_STEP_ACTION_NONE** }

Functions

- gac_t ∗ gac_create (gac_filter_parameter_t ∗parameter)
- void gac_destroy (gac_t ∗h)
- bool gac_init (gac_t ∗h, gac_filter_parameter_t ∗parameter)
- bool gac_get_filter_parameter (gac_t ∗h, gac_filter_parameter_t ∗parameter)
- bool gac_get_filter_parameter_default (gac_filter_parameter_t ∗parameter)
- bool gac_filter_fixation (gac_filter_fixation_t ∗filter, gac_sample_t ∗sample, gac_fixation_t ∗fixation)
- gac_filter_fixation_t ∗ gac_filter_fixation_create (float dispersion_threshold, double duration_threshold)
- void gac_filter_fixation_destroy (gac_filter_fixation_t ∗filter)
- bool gac_filter_fixation_init (gac_filter_fixation_t ∗filter, float dispersion_threshold, double duration_threshold)
- bool gac_filter_fixation_step (gac_filter_fixation_t ∗filter, gac_sample_t ∗sample, gac_fixation_t ∗fixation, gac_filter_step_action_t ∗action)
- uint32_t gac_filter_gap (gac_filter_gap_t ∗filter, gac_queue_t ∗samples, gac_sample_t ∗sample)
- gac_filter_gap_t ∗ gac_filter_gap_create (double max_gap_length, double sample_period)
- void gac_filter_gap_destroy (gac_filter_gap_t ∗filter)
- bool gac_filter_gap_init (gac_filter_gap_t ∗filter, double max_gap_length, double sample_period)
- gac_sample_t ∗ gac_filter_noise (gac_filter_noise_t ∗filter, gac_sample_t ∗sample)
- gac_filter_noise_t ∗ gac_filter_noise_create (gac_filter_noise_type_t type, uint32_t mid_idx)
- void gac_filter_noise_destroy (gac_filter_noise_t ∗filter)
- bool gac_filter_noise_init (gac_filter_noise_t ∗filter, gac_filter_noise_type_t type, uint32_t mid_idx)
- gac_sample_t ∗ gac_filter_noise_average (gac_filter_noise_t ∗filter)
- bool gac_filter_saccade (gac_filter_saccade_t ∗filter, gac_sample_t ∗sample, gac_saccade_t ∗saccade)
- gac_filter_saccade_t ∗ gac_filter_saccade_create (float velocity_threshold)
- void gac_filter_saccade_destroy (gac_filter_saccade_t ∗filter)
- bool gac_filter_saccade_init (gac_filter_saccade_t ∗filter, float velocity_threshold)
- bool gac_filter_saccade_step (gac_filter_saccade_t ∗filter, gac_sample_t ∗sample, gac_saccade_t ∗saccade, gac_filter_step_action_t ∗action)
- gac_fixation_t ∗ gac_fixation_create (vec3 ∗point, double timestamp, double duration)
- void gac_fixation_destroy (gac_fixation_t ∗fixation)
- bool gac_fixation_init (gac_fixation_t ∗fixation, vec3 ∗point, double timestamp, double duration)
- float gac_fixation_normalised_dispersion_threshold (float angle)
- bool gac_queue_clear (gac_queue_t ∗queue)
- gac_queue_t ∗ gac_queue_create (uint32_t length)
- void gac_queue_destroy (gac_queue_t ∗queue)
- bool gac_queue_grow (gac_queue_t ∗queue, uint32_t count)
- bool gac_queue_init (gac_queue_t ∗queue, uint32_t length)
- bool gac_queue_pop (gac_queue_t ∗queue, void ∗∗data)
- bool gac_queue_push (gac_queue_t ∗queue, void ∗data)
- bool gac_queue_remove (gac_queue_t ∗queue)
- bool gac_queue_set_rm_handler (gac_queue_t ∗queue, void(∗rm)(void ∗))

- gac_saccade_t ∗ gac_saccade_create (vec3 ∗point_start, vec3 ∗point_dest, double timestamp, double duration)
- void gac_saccade_destroy (gac_saccade_t ∗saccade)
- bool gac_saccade_init (gac_saccade_t ∗saccade, vec3 ∗point_start, vec3 ∗point_dest, double timestamp, double duration)
- gac_sample_t ∗ gac_sample_create (vec3 ∗origin, vec3 ∗point, double timestamp)
- void gac_sample_destroy (void ∗sample)
- bool gac_sample_init (gac_sample_t ∗sample, vec3 ∗origin, vec3 ∗point, double timestamp)
- bool gac_sample_window_cleanup (gac_t ∗h)
- bool gac_sample_window_fixation_filter (gac_t ∗h, gac_fixation_t ∗fixation)
- bool gac_sample_window_saccade_filter (gac_t ∗h, gac_saccade_t ∗saccade)
- void gac_sample_window_update (gac_t ∗h, float ox, float oy, float oz, float px, float py, float pz, double timestamp)
- bool gac_samples_average_point (gac_queue_t ∗samples, vec3 ∗avg, uint32_t count)
- bool gac_samples_average_origin (gac_queue_t ∗samples, vec3 ∗avg, uint32_t count)
- bool gac_samples_dispersion (gac_queue_t ∗samples, float ∗dispersion, uint32_t count)

5.1.1 Typedef Documentation

5.1.1.1 gac_filter_noise_type_t

```
typedef enum gac_filter_noise_type_e gac_filter_noise_type_t
```

gac_filter_noise_type_e

5.1.1.2 gac_filter_step_action_t

```
typedef enum gac_filter_step_action_e gac_filter_step_action_t
```

gac_filter_step_action_e

5.1.2 Enumeration Type Documentation

5.1.2.1 gac_filter_noise_type_e

```
enum gac_filter_noise_type_e
```

The available noise filter types

Enumerator

GAC_FILTER_NOISE_TYPE_AVERAGE	Moving average filtering
GAC_FILTER_NOISE_TYPE_MEDIAN	[not implemented] Moving median filtering

5.1.2.2 gac_filter_step_action_e

```
enum gac_filter_step_action_e
```

Actions to perform on the fixation sample window after fixation step.

5.1.3 Function Documentation

5.1.3.1 gac_create()

```
gac_t* gac_create (
             gac_filter_parameter_t * parameter )
```

Allocate the gaze analysis structure on the heap. This must be freed. If no parameter structure is provided default values are used. Refer to gac_init() for more information.

Parameters

parameter	An optional filter parameter structure.

Returns

A pointer to the allocated structure or NULL on failure.

5.1.3.2 gac_destroy()

```
void gac_destroy (
             gac_t * h )
```

Destroy the gaze analysis handler.

Parameters

h	A pointer to the gaze analysis handler.

5.1.3.3 gac_filter_fixation()

```
bool gac_filter_fixation (
             gac_filter_fixation_t * filter,
             gac_sample_t * sample,
             gac_fixation_t * fixation )
```

The fixation detection algorithm I-DT.

Parameters

filter	The gap filter structure holding the configuration parameters.
sample	The lastes sample
fixation	A location where a detected fixation is stored. This is only valid if the function returns true.

Returns

True if a fixation was detected, false otherwise.

5.1.3.4 gac_filter_fixation_create()

```
gac_filter_fixation_t* gac_filter_fixation_create (
             float dispersion_threshold,
             double duration_threshold )
```

Allocate a new fixation filter structure on the heap. This structure must be freed.

Parameters

dispersion_threshold	The dispersion thresholad in degrees.
duration_threshold	The duration threshold in milliseconds.

Returns

The allocated fixation filter structure or NULL on failure.

5.1.3.5 gac_filter_fixation_destroy()

```
void gac_filter_fixation_destroy (
             gac_filter_fixation_t * filter )
```

Destroy the fixation filter structure.

Parameters

filter	A pointer to the structure to destroy.

5.1.3.6 gac_filter_fixation_init()

```
bool gac_filter_fixation_init (
            gac_filter_fixation_t * filter,
            float dispersion_threshold,
            double duration_threshold )
```

Initialise a fixation filter structure.

Parameters

filter	The filter structure to initialise.
dispersion_threshold	The dispersion thresholad in degrees.
duration_threshold	The duration threshold in milliseconds.

Returns

True on success, false on failure.

5.1.3.7 gac_filter_fixation_step()

```
bool gac_filter_fixation_step (
            gac_filter_fixation_t * filter,
            gac_sample_t * sample,
            gac_fixation_t * fixation,
            gac_filter_step_action_t * action )
```

Internal function to compute the fixation detection algorithm I-DT. Do not use this function. INstead use either the function gac_sample_window_fixation_filter() or gac_filter_fixation().

Parameters

filter	The gap filter structure holding the configuration parameters.
sample	The lastes sample
fixation	A location where a detected fixation is stored. This is only valid if the function returns true.
action	An action code indicating to the parent function which action to perform on the sample window.

Returns

True if a fixation was detected, false otherwise.

5.1.3.8 gac_filter_gap()

```
uint32_t gac_filter_gap (
            gac_filter_gap_t * filter,
            gac_queue_t * samples,
            gac_sample_t * sample )
```

Fill in gaps between the last sample and the current sample if any. The number of samples to be filled in depends on the sample period. To avoid filling up large gaps the gap filling is limited to a maximal gap length (in milliseconds). The sample passed to the function is added as well.

Parameters

filter	The gap filter structure holding the configuration parameters.
samples	The sample queue to be filled in
sample	The lastes sample

Returns

The number of samples added to the sample window.

5.1.3.9 gac_filter_gap_create()

```
gac_filter_gap_t* gac_filter_gap_create (
            double max_gap_length,
            double sample_period )
```

Allocate the filter gap structure on the heap. this needs to be freed.

Parameters

max_gap_length	The maximal gap length in milliseconds to fil-in. Larger gaps are ignored. If set to 0 the filter is disabled.
sample_period	The expected average sample period in milliseconds (1000 / sample_rate).

Returns

A pointer to the allocated filter gap structure.

5.1.3.10 gac_filter_gap_destroy()

```
void gac_filter_gap_destroy (
            gac_filter_gap_t * filter )
```

Destroy the gap filter structure.

Parameters

filter	A pointer to the structure to destroy.

5.1.3.11 gac_filter_gap_init()

```
bool gac_filter_gap_init (
             gac_filter_gap_t * filter,
             double max_gap_length,
             double sample_period )
```

Initialise a filter gap structure.

Parameters

filter	A pointer to the struct to be initialised.
max_gap_length	The maximal gap length in milliseconds to fil-in. Larger gaps are ignored. If set to 0 the filter is disabled.
sample_period	The expected average sample period in milliseconds (1000 / sample_rate).

Returns

True on success, false on failure.

5.1.3.12 gac_filter_noise()

```
gac_sample_t* gac_filter_noise (
             gac_filter_noise_t * filter,
             gac_sample_t * sample )
```

A noise filter. The filter consecutively collects samples into a window and returns a filtered value when the window is full, otherwise the passed sample is returned. The filter maintains its won sample window.

Parameters

filter	The filter parameters.
sample	The sample to add to the filter window.

Returns

A filtered sample if the filter window is full or the sample passed to the function otherwise.

5.1.3.13 gac_filter_noise_average()

```
gac_sample_t* gac_filter_noise_average (
             gac_filter_noise_t * filter )
```

A moving average noise filter. It computes the average sample point and origin from all samples in the filter window and assigns the timestamp of the median sample (the sample in the middle of the window) to the averaged sample.

Parameters

filter	The filter parameters

Returns

A new averaged sample if the filter window is full or the sample passed to the function otherwise.

5.1.3.14 gac_filter_noise_create()

```
gac_filter_noise_t* gac_filter_noise_create (
            gac_filter_noise_type_t type,
            uint32_t mid_idx )
```

Allocate the noise filter structure. This needs to be freed.

Parameters

type	The noise filter type.
mid_idx	The mid index of the window. This is used to compute the length of the window: window_length = mid_idx ∗ 2 + 1. If set to 0 the filter is disabled.

Returns

A pointer to the allocated structure or NULL on failure.

5.1.3.15 gac_filter_noise_destroy()

```
void gac_filter_noise_destroy (
            gac_filter_noise_t * filter )
```

Destroy the noise filter structure.

Parameters

filter	A pointer to the structure to destroy.

5.1.3.16 gac_filter_noise_init()

```
bool gac_filter_noise_init (
            gac_filter_noise_t * filter,
            gac_filter_noise_type_t type,
            uint32_t mid_idx )
```

Initialises a noise filter structure.

Parameters

filter	A pointer to the structure to initialise.
type	The noise filter type.
mid_idx	The mid index of the window. This is used to compute the length of the window: window_length = mid_idx ∗ 2 + 1. If set to 0 the filter is disabled.

Returns

True on success, false on failure.

5.1.3.17 gac_filter_saccade()

```
bool gac_filter_saccade (
            gac_filter_saccade_t * filter,
            gac_sample_t * sample,
            gac_saccade_t * saccade )
```

The saccade detection algorithm I-VT.

Parameters

filter	The filter parameters
sample	The lastes sample
saccade	A location where a detected saccade is stored. This is only valid if the function returns true.

Returns

True if a saccade was detected, false otherwise.

5.1.3.18 gac_filter_saccade_create()

```
gac_filter_saccade_t* gac_filter_saccade_create (
            float velocity_threshold )
```

Allocate a new saccade filter structure on the heap. This needs to be freed.

Parameters

velocity_threshold	The velocity threshold in degrees per second.

Returns

A pointer to the allocated filter structure or NUll on failure.

5.1.3.19 gac_filter_saccade_destroy()

```
void gac_filter_saccade_destroy (
             gac_filter_saccade_t * filter )
```

Destroy the saccade filter structure.

Parameters

filter	A pointer to the structure to destroy.

5.1.3.20 gac_filter_saccade_init()

```
bool gac_filter_saccade_init (
             gac_filter_saccade_t * filter,
             float velocity_threshold )
```

Initialise a saccade filter structure.

Parameters

filter	A pointer to the filter structure to initialise.
velocity_threshold	The velocity threshold in degrees per second.

Returns

True on success, false on failure.

5.1.3.21 gac_filter_saccade_step()

```
bool gac_filter_saccade_step (
             gac_filter_saccade_t * filter,
             gac_sample_t * sample,
             gac_saccade_t * saccade,
             gac_filter_step_action_t * action )
```

Internal function to compute the I-VT algorithm. Do not use this function. Instead use either the function gac_sample_window_saccade_filter() or gac_filter_saccade().

Parameters

filter	The filter parameters
sample	The lastes sample
saccade	A location where a detected saccade is stored. This is only valid if the function returns true.
action	An action code indicating to the parent function which action to perform on the sample window.

Returns

True if a saccade was detected, false otherwise.

5.1.3.22 gac_fixation_create()

```
gac_fixation_t* gac_fixation_create (
             vec3 * point,
             double timestamp,
             double duration )
```

Allocate a new fixation structure on the heap. This structure must be freed.

Parameters

point	The fixation point.
timestamp	The timestamp of the fixation start.
duration	The duration of the fixation.

Returns

The allocated fixation structure or NULL on failure.

5.1.3.23 gac_fixation_destroy()

```
void gac_fixation_destroy (
             gac_fixation_t * fixation )
```

Destroy a fixation structure.

Parameters

fixation	A pointer to the fixation structure to destroy.

5.1.3.24 gac_fixation_init()

```
bool gac_fixation_init (
            gac_fixation_t * fixation,
            vec3 * point,
            double timestamp,
            double duration )
```

Initialise a fixation structure.

Parameters

fixation	The fixation structure to initialise.
point	The fixation point.
timestamp	The timestamp of the fixation start.
duration	The duration of the fixation.

Returns

True on success, false on failure.

5.1.3.25 gac_fixation_normalised_dispersion_threshold()

```
float gac_fixation_normalised_dispersion_threshold (
            float angle )
```

Compute a dispersion threashold assuming a unit distance. To get the actual dispersion threshold multiply this by the distance of the gaze origin to the gaze point.

Parameters

angle	The angel in degrees for which the dispersion threshold is computetd. Usual values range from 0.5 to 1 degree.

Returns

The normalized dispersion threshold.

5.1.3.26 gac_get_filter_parameter()

```
bool gac_get_filter_parameter (
            gac_t * h,
            gac_filter_parameter_t * parameter )
```

Get the filter parameters.

Parameters

h	A pointer to the gaze analysis structure to initialise.
parameter	A location where the filter parameter values can be stored.

Returns

True on success, false on failure.

5.1.3.27 gac_get_filter_parameter_default()

```
bool gac_get_filter_parameter_default (
            gac_filter_parameter_t * parameter )
```

Get the default filter parameter values.

Parameters

parameter	A location where the filter parameter values can be stored.

Returns

True on success, false on failure.

5.1.3.28 gac_init()

```
bool gac_init (
            gac_t * h,
            gac_filter_parameter_t * parameter )
```

Initialise the gaze analysis structure.

If no parameter structure is provided the following default values are set:

- fixation.dispersion_threshold = 0.5;
- fixation.duration_threshold = 100;
- saccade.velocity_threshold = 20;
- noise.mid_idx = 1;
- noise.type = GAC_FILTER_NOISE_TYPE_AVERAGE;
- gap.max_gap_length = 50;
- gap.sample_period = 16.67;

Parameters

h	A pointer to the gaze analysis structure to initialise.
parameter	An optional filter parameter structure.

Returns

True on success, false on failure.

5.1.3.29 gac_queue_clear()

```
bool gac_queue_clear (
             gac_queue_t * queue )
```

Remove all data items from the queue. The queue remove handler is used to free the data.

Parameters

queue	The queue to clear

Returns

True on success, false on failure.

5.1.3.30 gac_queue_create()

```
gac_queue_t* gac_queue_create (
             uint32_t length )
```

Allocate a new queue structure. This needs to be freed.

Parameters

length	The length of the queue.

Returns

The allocated queue structure.

5.1.3.31 gac_queue_destroy()

```
void gac_queue_destroy (
            gac_queue_t * queue )
```

Destroy a queue, all ist items and all data inside the items.

Parameters

queue	A pointer to the queue to destroy

5.1.3.32 gac_queue_grow()

```
bool gac_queue_grow (
            gac_queue_t * queue,
            uint32_t count )
```

Grow the queue.

Parameters

queue	A pointer to the queue to grow.
count	The number of spaces to add.

Returns

True on success, false on failure.

5.1.3.33 gac_queue_init()

```
bool gac_queue_init (
            gac_queue_t * queue,
            uint32_t length )
```

Initialise a queue structure.

Parameters

queue	A pointer to the queue to initialise.
length	The length of the queue

Returns

True on success, false on failure.

5.1.3.34 gac_queue_pop()

```
bool gac_queue_pop (
            gac_queue_t * queue,
            void ** data )
```

Remove a the data from the head of the queue and link the the now free space to the tail of the queue.

Parameters

queue	A pointer to the queue.
data	An optional location to store the popped data.

Returns

True on success, false on failure.

5.1.3.35 gac_queue_push()

```
bool gac_queue_push (
            gac_queue_t * queue,
            void * data )
```

Add a new item to the tail of the queue. If no more space is available, the queue is grown by one.

Parameters

queue	A pointer to the queue.
data	The data sample to be added to the tail of the queue.

Returns

True on success, false on failure.

5.1.3.36 gac_queue_remove()

```
bool gac_queue_remove (
            gac_queue_t * queue )
```

The same as gac_queue_pop() but also freeing the data item with the configured remove handler.

Parameters

queue	A pointer to the queue.

Returns

True on success, false on failure.

5.1.3.37 gac_queue_set_rm_handler()

```
bool gac_queue_set_rm_handler (
            gac_queue_t * queue,
            void(*)(void *) rm )
```

Set a remove handler which will be called whenever an item is removed from the queue.

Parameters

queue	A pointer to the queue.
rm	The renmove handler.

Returns

True on success, false on failure.

5.1.3.38 gac_saccade_create()

```
gac_saccade_t* gac_saccade_create (
            vec3 * point_start,
            vec3 * point_dest,
            double timestamp,
            double duration )
```

Allocate a new saccade structure on the heap. This needs to be freed.

Parameters

point_start	The first data point in a saccade.
point_dest	The last data point in a saccade.
timestamp	The timestamp of the beggining of the saccade.
duration	The duration of the saccade.

Returns

The allocated saccade structure on success or NULL on failure.

5.1.3.39 gac_saccade_destroy()

```
void gac_saccade_destroy (
            gac_saccade_t * saccade )
```

Destroy a saccade structure.

Parameters

saccade	A pointer to the saccade structure to destroy.

5.1.3.40 gac_saccade_init()

```
bool gac_saccade_init (
            gac_saccade_t * saccade,
            vec3 * point_start,
            vec3 * point_dest,
            double timestamp,
            double duration )
```

Initialise a saccade structure.

Parameters

saccade	A pointer to the saccade structure to initialise.
point_start	The first data point in a saccade.
point_dest	The last data point in a saccade.
timestamp	The timestamp of the beggining of the saccade.
duration	The duration of the saccade.

Returns

True on success, false on failure.

5.1.3.41 gac_sample_create()

```
gac_sample_t* gac_sample_create (
            vec3 * origin,
            vec3 * point,
            double timestamp )
```

Allocate a new sample structure on the heap. This needs to be freed.

Parameters

origin	The gaze origin vector.
point	The gaze point vector.
timestamp	The timestamp of the sample.

Returns

The allocated sample structure or NULL on failure.

5.1.3.42 gac_sample_destroy()

```
void gac_sample_destroy (
             void * sample )
```

Destroy a sample structure.

Parameters

sample	A pointer to the structure to be destroyed.

5.1.3.43 gac_sample_init()

```
bool gac_sample_init (
             gac_sample_t * sample,
             vec3 * origin,
             vec3 * point,
             double timestamp )
```

Initialise a sample structure.

Parameters

sample	The sample structure to initialise.
origin	The gaze origin vector.
point	The gaze point vector.
timestamp	The timestamp of the sample.

Returns

True on success, false on failure.

5.1.3.44 gac_sample_window_cleanup()

```
bool gac_sample_window_cleanup (
            gac_t * h )
```

Cleanup the sample window. This removes all sample data from the sample window which is no longer used for the gaze analysis.

Parameters

h	A pointer to the gaze analysis handler.

Returns

True on success, false on failure.

5.1.3.45 gac_sample_window_fixation_filter()

```
bool gac_sample_window_fixation_filter (
            gac_t * h,
            gac_fixation_t * fixation )
```

The fixation detection algorithm I-DT. This acts on the sample window managed by the functions gac_sample_window_update() and gac_sample_window_cleanup().

Parameters

h	A pointer to the gaze analysis handler.
fixation	A location where a detected fixation is stored. This is only valid if the function returns true.

Returns

True if a fixation was detected, false otherwise.

5.1.3.46 gac_sample_window_saccade_filter()

```
bool gac_sample_window_saccade_filter (
            gac_t * h,
            gac_saccade_t * saccade )
```

The saccade detection algorithm I-VT. This acts on the sample window managed by the functions gac_sample_window_update() and gac_sample_window_cleanup().

Parameters

h	A pointer to the gaze analysis handler.
saccade	A location where a detected saccade is stored. This is only valid if the function returns true.

Returns

True if a saccade was detected, false otherwise.

5.1.3.47 gac_sample_window_update()

```
void gac_sample_window_update (
            gac_t * h,
            float ox,
            float oy,
            float oz,
            float px,
            float py,
            float pz,
            double timestamp )
```

Update the sample window with a new sample. If noise filtering is enabled the filtered data is added to the sample window and the raw sample is dismissed. If gap filtering is enabled, sample gaps are filled-in with interpolated data samples.

Parameters

h	A pointer to the gaze analysis handler.
ox	The x coordinate of the gaze origin.
oy	The y coordinate of the gaze origin.
oz	The z coordinate of the gaze origin.
px	The x coordinate of the gaze point.
py	The y coordinate of the gaze point.
pz	The z coordinate of the gaze point.
timestamp	The timestamp of the sample.

5.1.3.48 gac_samples_average_origin()

```
bool gac_samples_average_origin (
            gac_queue_t * samples,
            vec3 * avg,
            uint32_t count )
```

Compute the average gaze origin of samples in the sample window.

Parameters

samples	A pointer to the sample window.
avg	A location to store the average gaze origin. This is only valid if the function returns true.
count	The number of samples to perform the computation on, starting by the queue tail (newest first). If 0 is passed, all samples are included.

5.1.3.49 gac_samples_average_point()

```
bool gac_samples_average_point (
            gac_queue_t * samples,
            vec3 * avg,
            uint32_t count )
```

Compute the average gaze point of samples in the sample window.

Parameters

samples	A pointer to the sample window.
avg	A location to store the average gaze point. This is only valid if the function returns true.
count	The number of samples to perform the computation on, starting by the queue tail (newest first). If 0 is passed, all samples are included.

5.1.3.50 gac_samples_dispersion()

```
bool gac_samples_dispersion (
            gac_queue_t * samples,
            float * dispersion,
            uint32_t count )
```

Compute the gaze point dispersion of samples in the sample window.

Parameters

samples	A pointer to the sample window.
dispersion	A location to store the dispersion value. This is only valid if the function returns true.
count	The number of samples to perform the computation on, starting by the queue tail (newest first). If 0 is passed, all samples are included.

Index