

libgac

v0.1.0

Generated by Doxygen 1.8.17

1 GAC - Gaze Analysis C Library	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 gac_filter_fixation_t Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Field Documentation	7
4.1.2.1 duration	7
4.1.2.2 duration_threshold	8
4.1.2.3 is_collecting	8
4.1.2.4 is_heap	8
4.1.2.5 normalized_dispersion_threshold	8
4.1.2.6 point	8
4.1.2.7 window	8
4.2 gac_filter_gap_t Struct Reference	8
4.2.1 Detailed Description	9
4.2.2 Field Documentation	9
4.2.2.1 is_enabled	9
4.2.2.2 is_heap	9
4.2.2.3 max_gap_length	9
4.2.2.4 sample_period	9
4.3 gac_filter_noise_t Struct Reference	10
4.3.1 Detailed Description	10
4.3.2 Field Documentation	10
4.3.2.1 is_enabled	10
4.3.2.2 is_heap	10
4.3.2.3 mid	10
4.3.2.4 type	10
4.3.2.5 window	11
4.4 gac_filter_parameter_t Struct Reference	11
4.4.1 Detailed Description	11
4.4.2 Field Documentation	11
4.4.2.1 dispersion_threshold	12
4.4.2.2 duration_threshold	12
4.4.2.3 fixation	12
4.4.2.4 gap	12
4.4.2.5 is_heap	12
4.4.2.6 max_gap_length	12

4.4.2.7 mid_idx	12
4.4.2.8 noise	13
4.4.2.9 saccade	13
4.4.2.10 sample_period	13
4.4.2.11 type	13
4.4.2.12 velocity_threshold	13
4.5 gac_filter_saccade_t Struct Reference	13
4.5.1 Detailed Description	14
4.5.2 Field Documentation	14
4.5.2.1 is_collecting	14
4.5.2.2 is_heap	14
4.5.2.3 velocity_threshold	14
4.5.2.4 window	14
4.6 gac_fixation_t Struct Reference	14
4.6.1 Detailed Description	15
4.6.2 Field Documentation	15
4.6.2.1 duration	15
4.6.2.2 is_heap	15
4.6.2.3 point	15
4.6.2.4 timestamp	15
4.7 gac_queue_item_t Struct Reference	16
4.7.1 Detailed Description	16
4.7.2 Field Documentation	16
4.7.2.1 data	16
4.7.2.2 next	16
4.7.2.3 prev	16
4.8 gac_queue_t Struct Reference	16
4.8.1 Detailed Description	17
4.8.2 Field Documentation	17
4.8.2.1 count	17
4.8.2.2 head	17
4.8.2.3 is_heap	17
4.8.2.4 length	17
4.8.2.5 rm	18
4.8.2.6 tail	18
4.9 gac_saccade_t Struct Reference	18
4.9.1 Detailed Description	18
4.9.2 Field Documentation	18
4.9.2.1 duration	18
4.9.2.2 is_heap	19
4.9.2.3 point_dest	19
4.9.2.4 point_start	19

4.9.2.5 timestamp	19
4.10 gac_sample_t Struct Reference	19
4.10.1 Detailed Description	19
4.10.2 Field Documentation	20
4.10.2.1 is_heap	20
4.10.2.2 origin	20
4.10.2.3 point	20
4.10.2.4 timestamp	20
4.11 gac_t Struct Reference	20
4.11.1 Detailed Description	21
4.11.2 Field Documentation	21
4.11.2.1 fixation	21
4.11.2.2 gap	21
4.11.2.3 is_heap	21
4.11.2.4 noise	21
4.11.2.5 parameter	21
4.11.2.6 saccade	21
4.11.2.7 samples	21
5 File Documentation	23
5.1 include/gac.h File Reference	23
5.1.1 Typedef Documentation	25
5.1.1.1 gac_filter_noise_type_t	25
5.1.1.2 gac_filter_step_action_t	25
5.1.2 Enumeration Type Documentation	25
5.1.2.1 gac_filter_noise_type_e	25
5.1.2.2 gac_filter_step_action_e	26
5.1.3 Function Documentation	26
5.1.3.1 gac_create()	26
5.1.3.2 gac_destroy()	26
5.1.3.3 gac_filter_fixation()	26
5.1.3.4 gac_filter_fixation_create()	27
5.1.3.5 gac_filter_fixation_destroy()	27
5.1.3.6 gac_filter_fixation_init()	28
5.1.3.7 gac_filter_fixation_step()	28
5.1.3.8 gac_filter_gap()	28
5.1.3.9 gac_filter_gap_create()	29
5.1.3.10 gac_filter_gap_destroy()	29
5.1.3.11 gac_filter_gap_init()	30
5.1.3.12 gac_filter_noise()	30
5.1.3.13 gac_filter_noise_average()	30
5.1.3.14 gac_filter_noise_create()	31

5.1.3.15 gac_filter_noise_destroy()	31
5.1.3.16 gac_filter_noise_init()	31
5.1.3.17 gac_filter_saccade()	32
5.1.3.18 gac_filter_saccade_create()	32
5.1.3.19 gac_filter_saccade_destroy()	33
5.1.3.20 gac_filter_saccade_init()	33
5.1.3.21 gac_filter_saccade_step()	33
5.1.3.22 gac_fixation_create()	34
5.1.3.23 gac_fixation_destroy()	34
5.1.3.24 gac_fixation_init()	35
5.1.3.25 gac_fixation_normalised_dispersion_threshold()	35
5.1.3.26 gac_get_filter_parameter()	35
5.1.3.27 gac_get_filter_parameter_default()	36
5.1.3.28 gac_init()	36
5.1.3.29 gac_queue_clear()	37
5.1.3.30 gac_queue_create()	37
5.1.3.31 gac_queue_destroy()	38
5.1.3.32 gac_queue_grow()	38
5.1.3.33 gac_queue_init()	38
5.1.3.34 gac_queue_pop()	39
5.1.3.35 gac_queue_push()	39
5.1.3.36 gac_queue_remove()	39
5.1.3.37 gac_queue_set_rm_handler()	40
5.1.3.38 gac_saccade_create()	40
5.1.3.39 gac_saccade_destroy()	41
5.1.3.40 gac_saccade_init()	41
5.1.3.41 gac_sample_create()	41
5.1.3.42 gac_sample_destroy()	42
5.1.3.43 gac_sample_init()	42
5.1.3.44 gac_sample_window_cleanup()	43
5.1.3.45 gac_sample_window_fixation_filter()	43
5.1.3.46 gac_sample_window_saccade_filter()	43
5.1.3.47 gac_sample_window_update()	44
5.1.3.48 gac_samples_average_origin()	44
5.1.3.49 gac_samples_average_point()	45
5.1.3.50 gac_samples_dispersion()	45

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.ordin.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 detected  
}  
// 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:

gac_filter_fixation_t	7
gac_filter_gap_t	8
gac_filter_noise_t	10
gac_filter_parameter_t	11
gac_filter_saccade_t	13
gac_fixation_t	14
gac_queue_item_t	16
gac_queue_t	16
gac_saccade_t	18
gac_sample_t	19
gac_t	20

Chapter 3

File Index

3.1 File List

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

include/ gac.h	23
------------------------------------------	----

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 threshold

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 handler.

[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: $\text{window_length} = \text{mid_idx} * 2 + 1$. 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 saccade 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](#)

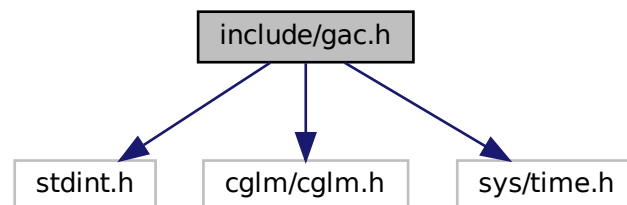
Chapter 5

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

<code>GAC_FILTER_NOISE_TYPE_AVERAGE</code>	Moving average filtering
<code>GAC_FILTER_NOISE_TYPE_MEDIAN</code>	[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

<i>parameter</i>	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

<i>h</i>	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

<i>filter</i>	The gap filter structure holding the configuration parameters.
<i>sample</i>	The lastes sample
<i>fixation</i>	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

<i>dispersion_threshold</i>	The dispersion thresholad in degrees.
<i>duration_threshold</i>	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

<i>filter</i>	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

<i>filter</i>	The filter structure to initialise.
<i>dispersion_threshold</i>	The dispersion threshold in degrees.
<i>duration_threshold</i>	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

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

<i>filter</i>	The gap filter structure holding the configuration parameters.
<i>samples</i>	The sample queue to be filled in
<i>sample</i>	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

<i>max_gap_length</i>	The maximal gap length in milliseconds to fil-in. Larger gaps are ignored. If set to 0 the filter is disabled.
<i>sample_period</i>	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

<i>filter</i>	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

<i>filter</i>	A pointer to the struct to be initialised.
<i>max_gap_length</i>	The maximal gap length in milliseconds to fil-in. Larger gaps are ignored. If set to 0 the filter is disabled.
<i>sample_period</i>	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

<i>filter</i>	The filter parameters.
<i>sample</i>	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

<i>filter</i>	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

<i>type</i>	The noise filter type.
<i>mid_idx</i>	The mid index of the window. This is used to compute the length of the window: <code>window_length = mid_idx * 2 + 1</code> . 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

<i>filter</i>	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

<i>filter</i>	A pointer to the structure to initialise.
<i>type</i>	The noise filter type.
<i>mid_idx</i>	The mid index of the window. This is used to compute the length of the window: <code>window_length = mid_idx * 2 + 1</code> . 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

<i>filter</i>	The filter parameters
<i>sample</i>	The lastes sample
<i>saccade</i>	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

<i>velocity_threshold</i>	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

<i>filter</i>	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

<i>filter</i>	A pointer to the filter structure to initialise.
<i>velocity_threshold</i>	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

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

<i>point</i>	The fixation point.
<i>timestamp</i>	The timestamp of the fixation start.
<i>duration</i>	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

<i>fixation</i>	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

<i>fixation</i>	The fixation structure to initialise.
<i>point</i>	The fixation point.
<i>timestamp</i>	The timestamp of the fixation start.
<i>duration</i>	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 threshold assuming a unit distance. To get the actual dispersion threshold multiply this by the distance of the gaze origin to the gaze point.

Parameters

<i>angle</i>	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

<i>h</i>	A pointer to the gaze analysis structure to initialise.
<i>parameter</i>	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

<i>parameter</i>	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

<i>h</i>	A pointer to the gaze analysis structure to initialise.
<i>parameter</i>	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

<i>queue</i>	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

<i>length</i>	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 its items and all data inside the items.

Parameters

<i>queue</i>	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

<i>queue</i>	A pointer to the queue to grow.
<i>count</i>	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

<i>queue</i>	A pointer to the queue to initialise.
<i>length</i>	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

<i>queue</i>	A pointer to the queue.
<i>data</i>	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

<i>queue</i>	A pointer to the queue.
<i>data</i>	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

<i>queue</i>	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

<i>queue</i>	A pointer to the queue.
<i>rm</i>	The remove 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

<i>point_start</i>	The first data point in a saccade.
<i>point_dest</i>	The last data point in a saccade.
<i>timestamp</i>	The timestamp of the beginning of the saccade.
<i>duration</i>	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

<i>saccade</i>	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

<i>saccade</i>	A pointer to the saccade structure to initialise.
<i>point_start</i>	The first data point in a saccade.
<i>point_dest</i>	The last data point in a saccade.
<i>timestamp</i>	The timestamp of the beginning of the saccade.
<i>duration</i>	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

<i>origin</i>	The gaze origin vector.
<i>point</i>	The gaze point vector.
<i>timestamp</i>	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

<i>sample</i>	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

<i>sample</i>	The sample structure to initialise.
<i>origin</i>	The gaze origin vector.
<i>point</i>	The gaze point vector.
<i>timestamp</i>	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

<i>h</i>	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

<i>h</i>	A pointer to the gaze analysis handler.
<i>fixation</i>	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

<i>h</i>	A pointer to the gaze analysis handler.
<i>saccade</i>	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

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

<i>samples</i>	A pointer to the sample window.
<i>avg</i>	A location to store the average gaze origin. This is only valid if the function returns true.
<i>count</i>	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

<i>samples</i>	A pointer to the sample window.
<i>avg</i>	A location to store the average gaze point. This is only valid if the function returns true.
<i>count</i>	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

<i>samples</i>	A pointer to the sample window.
<i>dispersion</i>	A location to store the dispersion value. This is only valid if the function returns true.
<i>count</i>	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

- count
 - [gac_queue_t](#), [17](#)
- data
 - [gac_queue_item_t](#), [16](#)
- dispersion_threshold
 - [gac_filter_parameter_t](#), [11](#)
- duration
 - [gac_filter_fixation_t](#), [7](#)
 - [gac_fixation_t](#), [15](#)
 - [gac_saccade_t](#), [18](#)
- duration_threshold
 - [gac_filter_fixation_t](#), [7](#)
 - [gac_filter_parameter_t](#), [12](#)
- fixation
 - [gac_filter_parameter_t](#), [12](#)
 - [gac_t](#), [21](#)
- gac.h
 - [gac_create](#), [26](#)
 - [gac_destroy](#), [26](#)
 - [gac_filter_fixation](#), [26](#)
 - [gac_filter_fixation_create](#), [27](#)
 - [gac_filter_fixation_destroy](#), [27](#)
 - [gac_filter_fixation_init](#), [27](#)
 - [gac_filter_fixation_step](#), [28](#)
 - [gac_filter_gap](#), [28](#)
 - [gac_filter_gap_create](#), [29](#)
 - [gac_filter_gap_destroy](#), [29](#)
 - [gac_filter_gap_init](#), [30](#)
 - [gac_filter_noise](#), [30](#)
 - [gac_filter_noise_average](#), [30](#)
 - [gac_filter_noise_create](#), [31](#)
 - [gac_filter_noise_destroy](#), [31](#)
 - [gac_filter_noise_init](#), [31](#)
 - [GAC_FILTER_NOISE_TYPE_AVERAGE](#), [25](#)
 - [gac_filter_noise_type_e](#), [25](#)
 - [GAC_FILTER_NOISE_TYPE_MEDIAN](#), [25](#)
 - [gac_filter_noise_type_t](#), [25](#)
 - [gac_filter_saccade](#), [32](#)
 - [gac_filter_saccade_create](#), [32](#)
 - [gac_filter_saccade_destroy](#), [33](#)
 - [gac_filter_saccade_init](#), [33](#)
 - [gac_filter_saccade_step](#), [33](#)
 - [gac_filter_step_action_e](#), [26](#)
 - [gac_filter_step_action_t](#), [25](#)
 - [gac_fixation_create](#), [34](#)
 - [gac_fixation_destroy](#), [34](#)
 - [gac_fixation_init](#), [34](#)
 - [gac_fixation_normalised_dispersion_threshold](#), [35](#)
 - [gac_get_filter_parameter](#), [35](#)
 - [gac_get_filter_parameter_default](#), [36](#)
 - [gac_init](#), [36](#)
 - [gac_queue_clear](#), [37](#)
 - [gac_queue_create](#), [37](#)
 - [gac_queue_destroy](#), [37](#)
 - [gac_queue_grow](#), [38](#)
 - [gac_queue_init](#), [38](#)
 - [gac_queue_pop](#), [39](#)
 - [gac_queue_push](#), [39](#)
 - [gac_queue_remove](#), [39](#)
 - [gac_queue_set_rm_handler](#), [40](#)
 - [gac_saccade_create](#), [40](#)
 - [gac_saccade_destroy](#), [41](#)
 - [gac_saccade_init](#), [41](#)
 - [gac_sample_create](#), [41](#)
 - [gac_sample_destroy](#), [42](#)
 - [gac_sample_init](#), [42](#)
 - [gac_sample_window_cleanup](#), [42](#)
 - [gac_sample_window_fixation_filter](#), [43](#)
 - [gac_sample_window_saccade_filter](#), [43](#)
 - [gac_sample_window_update](#), [44](#)
 - [gac_samples_average_origin](#), [44](#)
 - [gac_samples_average_point](#), [45](#)
 - [gac_samples_dispersion](#), [45](#)
- [gac_create](#)
 - [gac.h](#), [26](#)
- [gac_destroy](#)
 - [gac.h](#), [26](#)
- [gac_filter_fixation](#)
 - [gac.h](#), [26](#)
- [gac_filter_fixation_create](#)
 - [gac.h](#), [27](#)
- [gac_filter_fixation_destroy](#)
 - [gac.h](#), [27](#)
- [gac_filter_fixation_init](#)
 - [gac.h](#), [27](#)
- [gac_filter_fixation_step](#)
 - [gac.h](#), [28](#)
- [gac_filter_fixation_t](#), [7](#)
 - [duration](#), [7](#)
 - [duration_threshold](#), [7](#)
 - [is_collecting](#), [8](#)
 - [is_heap](#), [8](#)
 - [normalized_dispersion_threshold](#), [8](#)
 - [point](#), [8](#)
 - [window](#), [8](#)

- [gac_filter_gap](#)

- gac.h, [28](#)
- gac_filter_gap_create
 - gac.h, [29](#)
- gac_filter_gap_destroy
 - gac.h, [29](#)
- gac_filter_gap_init
 - gac.h, [30](#)
- gac_filter_gap_t, [8](#)
 - is_enabled, [9](#)
 - is_heap, [9](#)
 - max_gap_length, [9](#)
 - sample_period, [9](#)
- gac_filter_noise
 - gac.h, [30](#)
- gac_filter_noise_average
 - gac.h, [30](#)
- gac_filter_noise_create
 - gac.h, [31](#)
- gac_filter_noise_destroy
 - gac.h, [31](#)
- gac_filter_noise_init
 - gac.h, [31](#)
- gac_filter_noise_t, [10](#)
 - is_enabled, [10](#)
 - is_heap, [10](#)
 - mid, [10](#)
 - type, [10](#)
 - window, [10](#)
- GAC_FILTER_NOISE_TYPE_AVERAGE
 - gac.h, [25](#)
- gac_filter_noise_type_e
 - gac.h, [25](#)
- GAC_FILTER_NOISE_TYPE_MEDIAN
 - gac.h, [25](#)
- gac_filter_noise_type_t
 - gac.h, [25](#)
- gac_filter_parameter_t, [11](#)
 - dispersion_threshold, [11](#)
 - duration_threshold, [12](#)
 - fixation, [12](#)
 - gap, [12](#)
 - is_heap, [12](#)
 - max_gap_length, [12](#)
 - mid_idx, [12](#)
 - noise, [12](#)
 - saccade, [13](#)
 - sample_period, [13](#)
 - type, [13](#)
 - velocity_threshold, [13](#)
- gac_filter_saccade
 - gac.h, [32](#)
- gac_filter_saccade_create
 - gac.h, [32](#)
- gac_filter_saccade_destroy
 - gac.h, [33](#)
- gac_filter_saccade_init
 - gac.h, [33](#)
- gac_filter_saccade_step
 - gac.h, [33](#)
- gac_filter_saccade_t, [13](#)
 - is_collecting, [14](#)
 - is_heap, [14](#)
 - velocity_threshold, [14](#)
 - window, [14](#)
- gac_filter_step_action_e
 - gac.h, [26](#)
- gac_filter_step_action_t
 - gac.h, [25](#)
- gac_fixation_create
 - gac.h, [34](#)
- gac_fixation_destroy
 - gac.h, [34](#)
- gac_fixation_init
 - gac.h, [34](#)
- gac_fixation_normalised_dispersion_threshold
 - gac.h, [35](#)
- gac_fixation_t, [14](#)
 - duration, [15](#)
 - is_heap, [15](#)
 - point, [15](#)
 - timestamp, [15](#)
- gac_get_filter_parameter
 - gac.h, [35](#)
- gac_get_filter_parameter_default
 - gac.h, [36](#)
- gac_init
 - gac.h, [36](#)
- gac_queue_clear
 - gac.h, [37](#)
- gac_queue_create
 - gac.h, [37](#)
- gac_queue_destroy
 - gac.h, [37](#)
- gac_queue_grow
 - gac.h, [38](#)
- gac_queue_init
 - gac.h, [38](#)
- gac_queue_item_t, [16](#)
 - data, [16](#)
 - next, [16](#)
 - prev, [16](#)
- gac_queue_pop
 - gac.h, [39](#)
- gac_queue_push
 - gac.h, [39](#)
- gac_queue_remove
 - gac.h, [39](#)
- gac_queue_set_rm_handler
 - gac.h, [40](#)
- gac_queue_t, [16](#)
 - count, [17](#)
 - head, [17](#)
 - is_heap, [17](#)
 - length, [17](#)
 - rm, [17](#)
 - tail, [18](#)

- gac_saccade_create
 - gac.h, [40](#)
- gac_saccade_destroy
 - gac.h, [41](#)
- gac_saccade_init
 - gac.h, [41](#)
- gac_saccade_t, [18](#)
 - duration, [18](#)
 - is_heap, [18](#)
 - point_dest, [19](#)
 - point_start, [19](#)
 - timestamp, [19](#)
- gac_sample_create
 - gac.h, [41](#)
- gac_sample_destroy
 - gac.h, [42](#)
- gac_sample_init
 - gac.h, [42](#)
- gac_sample_t, [19](#)
 - is_heap, [20](#)
 - origin, [20](#)
 - point, [20](#)
 - timestamp, [20](#)
- gac_sample_window_cleanup
 - gac.h, [42](#)
- gac_sample_window_fixation_filter
 - gac.h, [43](#)
- gac_sample_window_saccade_filter
 - gac.h, [43](#)
- gac_sample_window_update
 - gac.h, [44](#)
- gac_samples_average_origin
 - gac.h, [44](#)
- gac_samples_average_point
 - gac.h, [45](#)
- gac_samples_dispersion
 - gac.h, [45](#)
- gac_t, [20](#)
 - fixation, [21](#)
 - gap, [21](#)
 - is_heap, [21](#)
 - noise, [21](#)
 - parameter, [21](#)
 - saccade, [21](#)
 - samples, [21](#)
- gap
 - gac_filter_parameter_t, [12](#)
 - gac_t, [21](#)
- head
 - gac_queue_t, [17](#)
- include/gac.h, [23](#)
- is_collecting
 - gac_filter_fixation_t, [8](#)
 - gac_filter_saccade_t, [14](#)
- is_enabled
 - gac_filter_gap_t, [9](#)
 - gac_filter_noise_t, [10](#)
- is_heap
 - gac_filter_fixation_t, [8](#)
 - gac_filter_gap_t, [9](#)
 - gac_filter_noise_t, [10](#)
 - gac_filter_parameter_t, [12](#)
 - gac_filter_saccade_t, [14](#)
 - gac_fixation_t, [15](#)
 - gac_queue_t, [17](#)
 - gac_saccade_t, [18](#)
 - gac_sample_t, [20](#)
 - gac_t, [21](#)
- length
 - gac_queue_t, [17](#)
- max_gap_length
 - gac_filter_gap_t, [9](#)
 - gac_filter_parameter_t, [12](#)
- mid
 - gac_filter_noise_t, [10](#)
- mid_idx
 - gac_filter_parameter_t, [12](#)
- next
 - gac_queue_item_t, [16](#)
- noise
 - gac_filter_parameter_t, [12](#)
 - gac_t, [21](#)
- normalized_dispersion_threshold
 - gac_filter_fixation_t, [8](#)
- origin
 - gac_sample_t, [20](#)
- parameter
 - gac_t, [21](#)
- point
 - gac_filter_fixation_t, [8](#)
 - gac_fixation_t, [15](#)
 - gac_sample_t, [20](#)
- point_dest
 - gac_saccade_t, [19](#)
- point_start
 - gac_saccade_t, [19](#)
- prev
 - gac_queue_item_t, [16](#)
- rm
 - gac_queue_t, [17](#)
- saccade
 - gac_filter_parameter_t, [13](#)
 - gac_t, [21](#)
- sample_period
 - gac_filter_gap_t, [9](#)
 - gac_filter_parameter_t, [13](#)
- samples
 - gac_t, [21](#)
- tail

- [gac_queue_t](#), [18](#)
- timestamp
 - [gac_fixation_t](#), [15](#)
 - [gac_saccade_t](#), [19](#)
 - [gac_sample_t](#), [20](#)
- type
 - [gac_filter_noise_t](#), [10](#)
 - [gac_filter_parameter_t](#), [13](#)
- velocity_threshold
 - [gac_filter_parameter_t](#), [13](#)
 - [gac_filter_saccade_t](#), [14](#)
- window
 - [gac_filter_fixation_t](#), [8](#)
 - [gac_filter_noise_t](#), [10](#)
 - [gac_filter_saccade_t](#), [14](#)