

libgac

v0.2.2

Generated by Doxygen 1.8.17

1 GAC - Gaze Analysis C Library	1
2 Changelog	5
3 Data Structure Index	7
3.1 Data Structures	7
4 File Index	9
4.1 File List	9
5 Data Structure Documentation	11
5.1 gac_aoi_analysis_t Struct Reference	11
5.1.1 Detailed Description	11
5.1.2 Field Documentation	11
5.1.2.1 _me	11
5.1.2.2 aoi_visited_before_count	11
5.1.2.3 dwell_time	11
5.1.2.4 dwell_time_relative	12
5.1.2.5 enter_saccade_count	12
5.1.2.6 first_fixation	12
5.1.2.7 first_saccade	12
5.1.2.8 fixation_count	12
5.1.2.9 fixation_count_relative	12
5.2 gac_aoi_collection_analysis_result_t Struct Reference	12
5.2.1 Detailed Description	12
5.2.2 Field Documentation	12
5.2.2.1 _me	13
5.2.2.2 aois	13
5.2.2.3 count	13
5.2.2.4 items	13
5.2.2.5 trial_id	13
5.3 gac_aoi_collection_analysis_t Struct Reference	13
5.3.1 Detailed Description	13
5.3.2 Field Documentation	13
5.3.2.1 _me	13
5.3.2.2 aoi_visited_count	13
5.3.2.3 dwell_time	14
5.3.2.4 fixation_count	14
5.3.2.5 trial_id	14
5.4 gac_aoi_collection_t Struct Reference	14
5.4.1 Detailed Description	14
5.4.2 Field Documentation	14
5.4.2.1 _me	14
5.4.2.2 analysis	14

5.4.2.3 aois	14
5.4.2.4 count	15
5.4.2.5 items	15
5.5 gac_aoi_t Struct Reference	15
5.5.1 Detailed Description	15
5.5.2 Field Documentation	15
5.5.2.1 _me	15
5.5.2.2 analysis	15
5.5.2.3 avg_edge_len	16
5.5.2.4 bounding_box	16
5.5.2.5 count	16
5.5.2.6 items	16
5.5.2.7 label	16
5.5.2.8 points	16
5.5.2.9 ray_origin	16
5.5.2.10 resolution_x	16
5.5.2.11 resolution_y	16
5.6 gac_filter_fixation_t Struct Reference	16
5.6.1 Detailed Description	17
5.6.2 Field Documentation	17
5.6.2.1 _me	17
5.6.2.2 duration	17
5.6.2.3 duration_threshold	17
5.6.2.4 is_collecting	17
5.6.2.5 new_samples	17
5.6.2.6 normalized_dispersion_threshold	17
5.6.2.7 point	17
5.6.2.8 screen_point	17
5.6.2.9 window	17
5.7 gac_filter_gap_t Struct Reference	18
5.7.1 Detailed Description	18
5.7.2 Field Documentation	18
5.7.2.1 _me	18
5.7.2.2 is_enabled	18
5.7.2.3 max_gap_length	18
5.7.2.4 sample_period	18
5.8 gac_filter_noise_t Struct Reference	18
5.8.1 Detailed Description	18
5.8.2 Field Documentation	19
5.8.2.1 _me	19
5.8.2.2 is_enabled	19
5.8.2.3 mid	19

5.8.2.4 type	19
5.8.2.5 window	19
5.9 gac_filter_parameter_t Struct Reference	19
5.9.1 Detailed Description	20
5.9.2 Field Documentation	20
5.9.2.1 _me	20
5.9.2.2 dispersion_threshold	20
5.9.2.3 duration_threshold	20
5.9.2.4 fixation	20
5.9.2.5 gap	20
5.9.2.6 max_gap_length	20
5.9.2.7 mid_idx	20
5.9.2.8 noise	20
5.9.2.9 saccade	20
5.9.2.10 sample_period	20
5.9.2.11 type	21
5.9.2.12 velocity_threshold	21
5.10 gac_filter_saccade_t Struct Reference	21
5.10.1 Detailed Description	21
5.10.2 Field Documentation	21
5.10.2.1 _me	21
5.10.2.2 is_collecting	21
5.10.2.3 new_samples	21
5.10.2.4 velocity_threshold	21
5.10.2.5 window	21
5.11 gac_fixation_t Struct Reference	22
5.11.1 Detailed Description	22
5.11.2 Field Documentation	22
5.11.2.1 _me	22
5.11.2.2 duration	22
5.11.2.3 first_sample	22
5.11.2.4 point	22
5.11.2.5 screen_point	22
5.12 gac_plane_t Struct Reference	22
5.12.1 Detailed Description	23
5.12.2 Field Documentation	23
5.12.2.1 _me	23
5.12.2.2 e1	23
5.12.2.3 e2	23
5.12.2.4 m	23
5.12.2.5 norm	23
5.12.2.6 p1	23

5.12.2.7 p2	23
5.12.2.8 p3	24
5.13 gac_queue_item_t Struct Reference	24
5.13.1 Detailed Description	24
5.13.2 Field Documentation	24
5.13.2.1 data	24
5.13.2.2 next	24
5.13.2.3 prev	24
5.14 gac_queue_t Struct Reference	24
5.14.1 Detailed Description	25
5.14.2 Field Documentation	25
5.14.2.1 _me	25
5.14.2.2 count	25
5.14.2.3 head	25
5.14.2.4 length	25
5.14.2.5 rm	25
5.14.2.6 tail	25
5.15 gac_saccade_t Struct Reference	25
5.15.1 Detailed Description	25
5.15.2 Field Documentation	25
5.15.2.1 _me	26
5.15.2.2 first_sample	26
5.15.2.3 last_sample	26
5.16 gac_sample_t Struct Reference	26
5.16.1 Detailed Description	26
5.16.2 Field Documentation	26
5.16.2.1 _me	26
5.16.2.2 label	26
5.16.2.3 label_onset	26
5.16.2.4 origin	27
5.16.2.5 point	27
5.16.2.6 screen_point	27
5.16.2.7 timestamp	27
5.16.2.8 trial_id	27
5.16.2.9 trial_onset	27
5.17 gac_screen_t Struct Reference	27
5.17.1 Detailed Description	27
5.17.2 Field Documentation	27
5.17.2.1 _me	27
5.17.2.2 height	28
5.17.2.3 origin	28
5.17.2.4 plane	28

5.17.2.5 resolution_x	28
5.17.2.6 resolution_y	28
5.17.2.7 width	28
5.18 gac_t Struct Reference	28
5.18.1 Detailed Description	28
5.18.2 Field Documentation	29
5.18.2.1 _me	29
5.18.2.2 aoic	29
5.18.2.3 fixation	29
5.18.2.4 gap	29
5.18.2.5 label_timestamp	29
5.18.2.6 last_sample	29
5.18.2.7 noise	29
5.18.2.8 parameter	29
5.18.2.9 saccade	29
5.18.2.10 samples	29
5.18.2.11 screen	29
5.18.2.12 trial_timestamp	30
6 File Documentation	31
6.1 include/gac.h File Reference	31
6.1.1 Detailed Description	32
6.1.2 Function Documentation	32
6.1.2.1 gac_add_aoi()	32
6.1.2.2 gac_create()	32
6.1.2.3 gac_destroy()	33
6.1.2.4 gac_finalise()	33
6.1.2.5 gac_get_filter_parameter()	33
6.1.2.6 gac_get_filter_parameter_default()	34
6.1.2.7 gac_init()	34
6.1.2.8 gac_sample_window_cleanup()	34
6.1.2.9 gac_sample_window_fixation_filter()	35
6.1.2.10 gac_sample_window_saccade_filter()	35
6.1.2.11 gac_sample_window_update()	35
6.1.2.12 gac_sample_window_update_screen()	36
6.1.2.13 gac_sample_window_update_vec()	37
6.1.2.14 gac_set_screen()	37
6.1.2.15 gac_version()	38
6.2 include/gac_aoi.h File Reference	38
6.2.1 Detailed Description	40
6.2.2 Macro Definition Documentation	40
6.2.2.1 GAC_AOI_MAX	40

6.2.2.2 GAC_AOI_MAX_LABEL_LEN	40
6.2.2.3 GAC_AOI_MAX_POINTS	40
6.2.3 Typedef Documentation	40
6.2.3.1 gac_aoi_orientation_t	40
6.2.4 Enumeration Type Documentation	40
6.2.4.1 gac_aoi_orientation_e	40
6.2.5 Function Documentation	41
6.2.5.1 gac_aoi_add_point()	41
6.2.5.2 gac_aoi_add_point_res()	41
6.2.5.3 gac_aoi_add_rect()	41
6.2.5.4 gac_aoi_add_rect_res()	42
6.2.5.5 gac_aoi_copy()	42
6.2.5.6 gac_aoi_copy_to()	42
6.2.5.7 gac_aoi_create()	43
6.2.5.8 gac_aoi_destroy()	43
6.2.5.9 gac_aoi_includes_point()	43
6.2.5.10 gac_aoi_includes_point_res()	43
6.2.5.11 gac_aoi_init()	44
6.2.5.12 gac_aoi_intersect()	44
6.2.5.13 gac_aoi_orientation_triplet()	44
6.2.5.14 gac_aoi_point_on_segment()	45
6.2.5.15 gac_aoi_set_resolution()	45
6.3 include/gac_aoi_analysis.h File Reference	46
6.3.1 Detailed Description	47
6.3.2 Function Documentation	48
6.3.2.1 gac_aoi_analysis_clear()	48
6.3.2.2 gac_aoi_analysis_copy()	48
6.3.2.3 gac_aoi_analysis_copy_to()	48
6.3.2.4 gac_aoi_analysis_create()	49
6.3.2.5 gac_aoi_analysis_destroy()	49
6.3.2.6 gac_aoi_analysis_init()	49
6.4 include/gac_aoi_collection.h File Reference	49
6.4.1 Detailed Description	51
6.4.2 Function Documentation	51
6.4.2.1 gac_aoi_collection_add()	51
6.4.2.2 gac_aoi_collection_analyse_clear()	51
6.4.2.3 gac_aoi_collection_analyse_finalise()	52
6.4.2.4 gac_aoi_collection_analyse_fixation()	52
6.4.2.5 gac_aoi_collection_analyse_saccade()	52
6.4.2.6 gac_aoi_collection_assign()	53
6.4.2.7 gac_aoi_collection_create()	53
6.4.2.8 gac_aoi_collection_destroy()	53

6.4.2.9 gac_aoi_collection_init()	53
6.5 include/gac_aoi_collection_analysis.h File Reference	54
6.5.1 Detailed Description	55
6.5.2 Function Documentation	55
6.5.2.1 gac_aoi_collection_analysis_clear()	55
6.5.2.2 gac_aoi_collection_analysis_create()	56
6.5.2.3 gac_aoi_collection_analysis_destroy()	56
6.5.2.4 gac_aoi_collection_analysis_init()	56
6.6 include/gac_filter_fixation.h File Reference	56
6.6.1 Detailed Description	58
6.6.2 Function Documentation	58
6.6.2.1 gac_filter_fixation()	58
6.6.2.2 gac_filter_fixation_create()	58
6.6.2.3 gac_filter_fixation_destroy()	59
6.6.2.4 gac_filter_fixation_init()	59
6.7 include/gac_filter_gap.h File Reference	59
6.7.1 Detailed Description	60
6.7.2 Function Documentation	61
6.7.2.1 gac_filter_gap()	61
6.7.2.2 gac_filter_gap_create()	61
6.7.2.3 gac_filter_gap_destroy()	62
6.7.2.4 gac_filter_gap_init()	62
6.8 include/gac_filter_noise.h File Reference	62
6.8.1 Detailed Description	64
6.8.2 Typedef Documentation	64
6.8.2.1 gac_filter_noise_type_t	64
6.8.3 Enumeration Type Documentation	64
6.8.3.1 gac_filter_noise_type_e	64
6.8.4 Function Documentation	64
6.8.4.1 gac_filter_noise()	64
6.8.4.2 gac_filter_noise_average()	65
6.8.4.3 gac_filter_noise_create()	65
6.8.4.4 gac_filter_noise_destroy()	65
6.8.4.5 gac_filter_noise_init()	66
6.9 include/gac_filter_saccade.h File Reference	66
6.9.1 Detailed Description	68
6.9.2 Function Documentation	68
6.9.2.1 gac_filter_saccade()	68
6.9.2.2 gac_filter_saccade_create()	68
6.9.2.3 gac_filter_saccade_destroy()	69
6.9.2.4 gac_filter_saccade_init()	69
6.10 include/gac_fixation.h File Reference	69

6.10.1 Detailed Description	71
6.10.2 Function Documentation	72
6.10.2.1 gac_fixation_copy()	72
6.10.2.2 gac_fixation_copy_to()	72
6.10.2.3 gac_fixation_create()	72
6.10.2.4 gac_fixation_destroy()	73
6.10.2.5 gac_fixation_init()	73
6.10.2.6 gac_fixation_normalised_dispersion_threshold()	73
6.11 include/gac_plane.h File Reference	74
6.11.1 Detailed Description	75
6.11.2 Function Documentation	75
6.11.2.1 gac_plane_create()	75
6.11.2.2 gac_plane_destroy()	75
6.11.2.3 gac_plane_init()	76
6.11.2.4 gac_plane_intersection()	76
6.11.2.5 gac_plane_point()	76
6.12 include/gac_queue.h File Reference	77
6.12.1 Detailed Description	78
6.12.2 Function Documentation	79
6.12.2.1 gac_queue_clear()	79
6.12.2.2 gac_queue_create()	79
6.12.2.3 gac_queue_destroy()	79
6.12.2.4 gac_queue_grow()	80
6.12.2.5 gac_queue_init()	80
6.12.2.6 gac_queue_pop()	80
6.12.2.7 gac_queue_push()	80
6.12.2.8 gac_queue_remove()	81
6.12.2.9 gac_queue_set_rm_handler()	81
6.13 include/gac_saccade.h File Reference	82
6.13.1 Detailed Description	83
6.13.2 Function Documentation	84
6.13.2.1 gac_saccade_copy()	84
6.13.2.2 gac_saccade_copy_to()	84
6.13.2.3 gac_saccade_create()	84
6.13.2.4 gac_saccade_destroy()	85
6.13.2.5 gac_saccade_init()	85
6.14 include/gac_sample.h File Reference	85
6.14.1 Detailed Description	87
6.14.2 Macro Definition Documentation	87
6.14.2.1 GAC_SAMPLE_MAX_LABEL_LEN	87
6.14.3 Function Documentation	87
6.14.3.1 gac_sample_copy()	87

6.14.3.2 gac_sample_copy_to()	88
6.14.3.3 gac_sample_create()	88
6.14.3.4 gac_sample_destroy()	88
6.14.3.5 gac_sample_get_label_timestamp()	89
6.14.3.6 gac_sample_get_onset()	89
6.14.3.7 gac_sample_get_trial_timestamp()	89
6.14.3.8 gac_sample_init()	90
6.14.3.9 gac_samples_average_origin()	90
6.14.3.10 gac_samples_average_point()	90
6.14.3.11 gac_samples_average_screen_point()	91
6.14.3.12 gac_samples_dispersion()	91
6.15 include/gac_screen.h File Reference	91
6.15.1 Detailed Description	92
6.15.2 Function Documentation	93
6.15.2.1 gac_screen_create()	93
6.15.2.2 gac_screen_destroy()	93
6.15.2.3 gac_screen_init()	93
6.15.2.4 gac_screen_point()	94
6.15.2.5 gac_screen_point_res()	94
6.15.2.6 gac_screen_set_resolution()	94
Index	97

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
- Area of interest (AOI) analysis

Quick Start

Initialise the gaze analysis handler:

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

To perform an AOI analysis, add some AOIs to the gaze analysis handler:

```
gac_aoi_t aoi;  
gac_aoi_init( &aoi, "my_circular_aoi" );  
gac_aoi_add_point( &aoi, 0.5, 0.4 );  
gac_aoi_add_point( &aoi, 0.5, 0.3 );  
gac_aoi_add_point( &aoi, 0.6, 0.2 );  
gac_aoi_add_point( &aoi, 0.7, 0.2 );  
gac_aoi_add_point( &aoi, 0.8, 0.3 );  
gac_aoi_add_point( &aoi, 0.8, 0.4 );  
gac_aoi_add_point( &aoi, 0.7, 0.5 );  
gac_aoi_add_point( &aoi, 0.6, 0.5 );  
gac_add_aoi( &h, &aoi );  
gac_aoi_init( &aoi, "my_rectangular_aoi" );  
gac_aoi_add_rect( &aoi, 0.3, 0.45, 0.1, 0.1 );  
gac_add_aoi( &h, &aoi );
```

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

```
int i, count;  
bool res;  
gac_fixation_t fixation;  
gac_saccade_t saccade;  
gac_aoi_collection_analysis_result_t analysis;  
// update the sample window with data.  
count = 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 );  
// updating the sample window may add multiple new samples (due to gap  
// filtering). Hence, we need to filter for each sample added.
```

```

for( i = 0; i < count; i++ )
{
    // check for saccade
    res = gac_sample_window_saccade_filter( &h, &saccade );
    if( res == true )
    {
        // perform AOI analysis on saccade data
        gac_aoi_collection_analyse_saccade( &h.aoic, &saccade );
        // saccade structures must be destroyed once they are no longer needed.
        gac_saccade_destroy( &saccade );
    }
    // check for fixation
    res = gac_sample_window_fixation_filter( &h, &fixation );
    if( res == true )
    {
        // perform AOI analysis on fixation data.
        res = gac_aoi_collection_analyse_fixation( &h->aoic, &fixation,
            &analysis );
        if( res == true )
        {
            // An AOI analysis entry is ready, do something with the analysis
            // data
        }
        // fixation structures must be destroyed once they are no longer needed.
        gac_fixation_destroy( &fixation );
    }
    // remove samples from the sample window which are no longer used
    gac_sample_window_cleanup( h );
}

```

After all samples were analysed, a finalization step is required to conclude the AOI analysis:

```

bool res = gac_finalise( &h, &analysis );
if( res )
{
    // An AOI analysis entry is ready, do something with the analysis data
}

```

Finally, destroy the gaze analysis handler:

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

Basic Concept

The library provides several functions to work with gaze data. The easiest approach is to use the functions `gac*_sample_window_*` as these maintain their own sample window and noise and gap filters can be configured through the `gac_filter_paramter_t` structure.

Alternatively it is possible to manually maintain a sample window and work with each filter individually. This means filter structures have to be created and destroyed manually and filtering has to be applied manually to a custom sample window. Refer to the API for more information.

Detection Algorithm

Fixations are detected with the I-DT algorithm (Salvucci & Goldberg 2000). Saccades are detected with the I-VT algorithm (Salvucci & Goldberg 2000).

Note that the resulting fixations and saccades will **not** fit together perfectly (e.g. a saccade follows a fixation and vice versa) because

1. both algorithms work with their own parameters which will most likely lead to gaps (data which is neither classified as part of a fixation nor saccade)
2. gaze data may be a recording of a smooth pursuit
3. gaps in the gaze data because of blinks or other data loss

For more details on the filter parameter options refer to the API documentation.

Filters

Optionally the gaze data is processed by

1. a moving average filter which computes the average of all samples in the filters own sliding window. Sample annotations (e.g. the label, trial ID, and timestamps) are copied from the data sample in the middle of the sliding window.
2. a gap fill-in filter where data samples are filled into gaps using linear interpolation.

For more details on the filter parameter options refer to the API documentation.

3d vs 2d Data

All calculations are performed on 3d data. If only 2d data is available this library cannot be used (yet). The reason for this is that with 3d data it is possible to compute an accurate dispersion and velocity threshold based on the distance of the gaze origin to the gaze point. For 2d data the dispersion and velocity threshold would need to be estimated based on the measured data which is not (yet) supported by the library.

However, it is possible to provide 2d data alongside 3d data for each data sample which will propagated to fixation and saccade result structures. To add 2d data for each sample instead of the function `gac_sample_window_update()` use `gac_sample_window_update_screen()`.

If 2d data is not available it is possible to compute it from 3d data. `gac_sample_window_update()` does this automatically if the screen location is defined. To define the screen location use the function `gac_set_screen()`.

Sample annotations

Each sample has two fields available for custom data annotation:

- `trial_id`: expects an integer number and can be used to e.g. associate a data point to a trial.
- `label`: expects a string and can be used to e.g. describe the currently displayed stimuli.

The annotations are propagated to the fixation and saccade result structures.

Further, each sample has two additional timestamp fields for onset information of the annotations:

- `trial_onset`: the amount of milliseconds since the last change in the field `trial_id`.
- `label_onset`: the amount of milliseconds since the last change in the field `label`.

Area of Interest (AOI) Analysis

The area of interest (AOI) analysis is performed based on fixations. Saccade information can also be used to extend the analysis but fixations are always required. For each distinct trial ID block an analysis of each AOI is performed.

To decide whether a sample point is inside an AOI a ray casting method is used where a virtual ray is drawn from an arbitrary point outside the AOI to the sample point. Then, every intersection with segments of the AOI contour is counted. If an even number of intersection is detected, the point lies outside of the AOI, otherwise the point lies inside the AOI. To improve performance, a coarse detection using a rectangular a bounding box is performed (if the sample point lies outside the bounding box it also lies outside the AOI).

Building the library on Linux (Ubuntu)

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 commands

```
autoreconf --install
./configure
make
```

To build and run tests use

```
cd test
make
```

To build and run the example use

```
cd example
make
make run
```

Building the library on Windows

Build the library on windows with **msys2**. Once installed start `msys2.exe`.

Some dependencies need to be installed. To do this type the following commands:

```
pacman -Syyu
pacman -Sy mingw-w64-x86_64-gcc
pacman -Sy autogen autoconf automake libtool
```

Finally, to build the library type

```
make
```

Build the example with the following commands:

```
cd example
make
```

To run the example make sure that the system knows the location of `msys2.dll` (either by adding the location to the PATH or by copying the file to the example folder). Run the example by starting `example.exe`.

Chapter 2

Changelog

v0.2.2 (latest)

New Features

- Add helper functions to compute timestamps and onsets.

Improvements

- Add non-rectangular AOI to the example.
- Improve reported AOI timestamp information.

Bug Fixes

- Fix trial ID and label onset calculations.
-

v0.2.1

Improvements

- Add aoic structure to gaze handler.

Bug Fixes

- Fix memory leak.
-

v0.2.0

New Features

- Add support for screen resolution (internally 2d coordinates are still stored as normalized values).
- Allow to define area of interests (AOI) and perform a basic analysis based on fixations and saccades.

Improvements

- Add MPL license.
- Add this changelog.
- Use a minimalistic include approach with cglm instead of including everything.
- Split code into individual file pairs (.c and .h) to separate concerns.

Bug Fixes

- Fix doxygen configuration.
-

v0.1.1

Initial release.

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

gac_aoi_analysis_t	11
gac_aoi_collection_analysis_result_t	12
gac_aoi_collection_analysis_t	13
gac_aoi_collection_t	14
gac_aoi_t	15
gac_filter_fixation_t	16
gac_filter_gap_t	18
gac_filter_noise_t	18
gac_filter_parameter_t	19
gac_filter_saccade_t	21
gac_fixation_t	22
gac_plane_t	22
gac_queue_item_t	24
gac_queue_t	24
gac_saccade_t	25
gac_sample_t	26
gac_screen_t	27
gac_t	28

Chapter 4

File Index

4.1 File List

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

include/ gac.h	31
include/ gac_aoi.h	38
include/ gac_aoi_analysis.h	46
include/ gac_aoi_collection.h	49
include/ gac_aoi_collection_analysis.h	54
include/ gac_filter_fixation.h	56
include/ gac_filter_gap.h	59
include/ gac_filter_noise.h	62
include/ gac_filter_saccade.h	66
include/ gac_fixation.h	69
include/ gac_plane.h	74
include/ gac_queue.h	77
include/ gac_saccade.h	82
include/ gac_sample.h	85
include/ gac_screen.h	91

Chapter 5

Data Structure Documentation

5.1 gac_aoi_analysis_t Struct Reference

```
#include <gac_aoi_analysis.h>
```

Data Fields

- void * [_me](#)
- uint32_t [aoi_visited_before_count](#)
- uint32_t [fixation_count](#)
- uint32_t [enter_saccade_count](#)
- double [fixation_count_relative](#)
- double [dwell_time](#)
- double [dwell_time_relative](#)
- gac_fixation_t [first_fixation](#)
- gac_saccade_t [first_saccade](#)

5.1.1 Detailed Description

A structure holding the AOI analysis results.

[gac_aoi_analysis_s](#)

5.1.2 Field Documentation

5.1.2.1 _me

```
void* gac_aoi_analysis_t::_me
```

Self-pointer to allocated structure for memory management.

5.1.2.2 aoi_visited_before_count

```
uint32_t gac_aoi_analysis_t::aoi_visited_before_count
```

The number of different AOIs visited before the first fixation hit this AOI.

5.1.2.3 dwell_time

```
double gac_aoi_analysis_t::dwell_time
```

The sum of all fixation durations on the AOI.

5.1.2.4 dwell_time_relative

`double gac_aoi_analysis_t::dwell_time_relative`

The relative trial time spent on the AOI. 1 is the sum of all fixation durations within the trial interest period. The trial interest period corresponds to all samples with the same trial ID.

5.1.2.5 enter_saccade_count

`uint32_t gac_aoi_analysis_t::enter_saccade_count`

The number of saccades entering the AOI.

5.1.2.6 first_fixation

`gac_fixation_t gac_aoi_analysis_t::first_fixation`

The first fixation on the AOI.

5.1.2.7 first_saccade

`gac_saccade_t gac_aoi_analysis_t::first_saccade`

The first saccade on the AOI.

5.1.2.8 fixation_count

`uint32_t gac_aoi_analysis_t::fixation_count`

The number of fixations in this AOI.

5.1.2.9 fixation_count_relative

`double gac_aoi_analysis_t::fixation_count_relative`

The relative number of fixations in this AOI where 1 is the number of all fixations within the trial interest period. The trial interest period corresponds to all samples with the same trial ID.

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

- [include/gac_aoi_analysis.h](#)

5.2 gac_aoi_collection_analysis_result_t Struct Reference

```
#include <gac_aoi_collection_analysis.h>
```

Data Fields

- `void * _me`
- `struct {`
 `struct {`
 `char label [GAC_AOI_MAX_LABEL_LEN]`
 `gac_aoi_analysis_t analysis`
 `} items [GAC_AOI_MAX]`
 `uint32_t count`
 `} aois`
- `uint32_t trial_id`

5.2.1 Detailed Description

A collection of AOIs.

[gac_aoi_collection_analysis_result_s](#)

5.2.2 Field Documentation

5.2.2.1 _me

`void* gac_aoi_collection_analysis_result_t::_me`
Self-pointer to allocated structure for memory management.

5.2.2.2 aois

`struct { ... } gac_aoi_collection_analysis_result_t::aois`
The collection of individual AOIs.

5.2.2.3 count

`uint32_t gac_aoi_collection_analysis_result_t::count`
The number of AOIs in the list.

5.2.2.4 items

`struct { ... } gac_aoi_collection_analysis_result_t::items[GAC_AOI_MAX]`
The aoi analysis list.

5.2.2.5 trial_id

`uint32_t gac_aoi_collection_analysis_result_t::trial_id`
The trial ID associated to the analysis.
The documentation for this struct was generated from the following file:

- `include/gac_aoi_collection_analysis.h`

5.3 gac_aoi_collection_analysis_t Struct Reference

```
#include <gac_aoi_collection_analysis.h>
```

Data Fields

- `void * _me`
- `uint32_t fixation_count`
- `uint32_t aoi_visited_count`
- `uint32_t trial_id`
- `double dwell_time`

5.3.1 Detailed Description

The AOI collection analysis data structure.
[gac_aoi_collection_analysis_s](#)

5.3.2 Field Documentation

5.3.2.1 _me

`void* gac_aoi_collection_analysis_t::_me`
Self-pointer to allocated structure for memory management.

5.3.2.2 aoi_visited_count

`uint32_t gac_aoi_collection_analysis_t::aoi_visited_count`
The number of visited aois.

5.3.2.3 dwell_time

`double gac_aoi_collection_analysis_t::dwell_time`
The summed duration of all fixations.

5.3.2.4 fixation_count

`uint32_t gac_aoi_collection_analysis_t::fixation_count`
The total fixation count.

5.3.2.5 trial_id

`uint32_t gac_aoi_collection_analysis_t::trial_id`
A number distinguishing one trial from another
The documentation for this struct was generated from the following file:

- `include/gac_aoi_collection_analysis.h`

5.4 gac_aoi_collection_t Struct Reference

```
#include <gac_aoi_collection.h>
```

Data Fields

- `void * _me`
- `struct {`
 - `gac_aoi_t * ptrs [GAC_AOI_MAX]`
 - `gac_aoi_t items [GAC_AOI_MAX]`
 - `uint32_t count``} aois`
- `gac_aoi_collection_analysis_t analysis`

5.4.1 Detailed Description

A collection of AOIs.
[gac_aoi_collection_s](#)

5.4.2 Field Documentation

5.4.2.1 _me

`void* gac_aoi_collection_t::_me`
Self-pointer to allocated structure for memory management.

5.4.2.2 analysis

`gac_aoi_collection_analysis_t gac_aoi_collection_t::analysis`
The analysis data of the AOI collection.

5.4.2.3 aois

`struct { ... } gac_aoi_collection_t::aois`
The collection of individual AOIs.

5.4.2.4 count

```
uint32_t gac_aoi_collection_t::count
```

The number of AOIs in the list.

5.4.2.5 items

```
gac_aoi_t gac_aoi_collection_t::items[GAC_AOI_MAX]
```

The aoi list.

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

- include/[gac_aoi_collection.h](#)

5.5 gac_aoi_t Struct Reference

```
#include <gac_aoi.h>
```

Data Fields

- void * [_me](#)
- [vec2](#) [ray_origin](#)
- float [avg_edge_len](#)
- float [resolution_x](#)
- float [resolution_y](#)
- char [label](#) [[GAC_AOI_MAX_LABEL_LEN](#)]
- struct {
 - [vec2](#) [items](#) [[GAC_AOI_MAX_POINTS](#)]
 - [uint32_t](#) [count](#)
- [points](#)
- struct {
 - float [x_min](#)
 - float [x_max](#)
 - float [y_min](#)
 - float [y_max](#)
- [bounding_box](#)
- [gac_aoi_analysis_t](#) [analysis](#)

5.5.1 Detailed Description

An area of interest (AOI) structure.

[gac_aoi_s](#)

5.5.2 Field Documentation

5.5.2.1 _me

```
void* gac_aoi_t::_me
```

Self-pointer to allocated structure for memory management.

5.5.2.2 analysis

```
gac_aoi_analysis_t gac_aoi_t::analysis
```

The analysis data of the AOI.

5.5.2.3 avg_edge_len

```
float gac_aoi_t::avg_edge_len
```

The average length of an AOI edge.

5.5.2.4 bounding_box

```
struct { ... } gac_aoi_t::bounding_box
```

A axis aligned bounding box to quickly do a coars check if a point is outside the polygon.

5.5.2.5 count

```
uint32_t gac_aoi_t::count
```

The number of points defining the AOI.

5.5.2.6 items

```
vec2 gac_aoi_t::items[GAC_AOI_MAX_POINTS]
```

The point list.

5.5.2.7 label

```
char gac_aoi_t::label[GAC_AOI_MAX_LABEL_LEN]
```

A label describing the aoi.

5.5.2.8 points

```
struct { ... } gac_aoi_t::points
```

The points forming the AOI. At least 3 points are required for a valid AOI.

5.5.2.9 ray_origin

```
vec2 gac_aoi_t::ray_origin
```

An arbitrary point outside the AOI.

5.5.2.10 resolution_x

```
float gac_aoi_t::resolution_x
```

The width of the screen resolution.

5.5.2.11 resolution_y

```
float gac_aoi_t::resolution_y
```

The height of the screen resolution.

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

- [include/gac_aoi.h](#)

5.6 gac_filter_fixation_t Struct Reference

```
#include <gac_filter_fixation.h>
```

Data Fields

- void * [_me](#)
- double [normalized_dispersion_threshold](#)
- double [duration_threshold](#)
- bool [is_collecting](#)
- [gac_queue_t](#) [window](#)

- uint32_t [new_samples](#)
- double [duration](#)
- vec2 [screen_point](#)
- vec3 [point](#)

5.6.1 Detailed Description

The fixation filter structure holding filter parameters.

[gac_filter_fixation_s](#)

5.6.2 Field Documentation

5.6.2.1 `_me`

`void* gac_filter_fixation_t::_me`

Self-pointer to allocated structure for memory management.

5.6.2.2 `duration`

`double gac_filter_fixation_t::duration`

The fixation duration

5.6.2.3 `duration_threshold`

`double gac_filter_fixation_t::duration_threshold`

The duration threshold

5.6.2.4 `is_collecting`

`bool gac_filter_fixation_t::is_collecting`

A flag indicating whether a fixation is ongoing.

5.6.2.5 `new_samples`

`uint32_t gac_filter_fixation_t::new_samples`

Counter to keep track of new items in the parent queue.

5.6.2.6 `normalized_dispersion_threshold`

`double gac_filter_fixation_t::normalized_dispersion_threshold`

The pre-computed dispersion threshold at unit distance

5.6.2.7 `point`

`vec3 gac_filter_fixation_t::point`

The fixation point

5.6.2.8 `screen_point`

`vec2 gac_filter_fixation_t::screen_point`

The fixation screen point

5.6.2.9 `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_filter_fixation.h`

5.7 gac_filter_gap_t Struct Reference

```
#include <gac_filter_gap.h>
```

Data Fields

- void * [_me](#)
- bool [is_enabled](#)
- double [max_gap_length](#)
- double [sample_period](#)

5.7.1 Detailed Description

The gap fill-in filter structure.

[gac_filter_gap_s](#)

5.7.2 Field Documentation

5.7.2.1 [_me](#)

```
void* gac_filter_gap_t::_me
```

Self-pointer to allocated structure for memory management.

5.7.2.2 [is_enabled](#)

```
bool gac_filter_gap_t::is_enabled
```

A flag indicating whether the filter is active or not

5.7.2.3 [max_gap_length](#)

```
double gac_filter_gap_t::max_gap_length
```

The maximal allowed gap length to be filled-in

5.7.2.4 [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_filter_gap.h](#)

5.8 gac_filter_noise_t Struct Reference

```
#include <gac_filter_noise.h>
```

Data Fields

- void * [_me](#)
- bool [is_enabled](#)
- [gac_queue_t](#) [window](#)
- [uint32_t](#) [mid](#)
- [gac_filter_noise_type_t](#) [type](#)

5.8.1 Detailed Description

The noise filter parameters.

[gac_filter_noise_s](#)

5.8.2 Field Documentation

5.8.2.1 _me

```
void* gac_filter_noise_t::_me
```

Self-pointer to allocated structure for memory management.

5.8.2.2 is_enabled

```
bool gac_filter_noise_t::is_enabled
```

A flag indicating whether the noise filter is active or not

5.8.2.3 mid

```
uint32_t gac_filter_noise_t::mid
```

The mid-point counter

5.8.2.4 type

```
gac_filter_noise_type_t gac_filter_noise_t::type
```

The noise filter type

5.8.2.5 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_filter_noise.h](#)

5.9 gac_filter_parameter_t Struct Reference

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

Data Fields

- void * [_me](#)
- 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](#)

5.9.1 Detailed Description

The filter parameter structure to initialise the gaze analysis handler.

[gac_filter_parameter_s](#)

5.9.2 Field Documentation

5.9.2.1 `_me`

```
void* gac_filter_parameter_t::_me
```

Self-pointer to allocated structure for memory management.

5.9.2.2 `dispersion_threshold`

```
float gac_filter_parameter_t::dispersion_threshold
```

The dispersion threshold in degrees.

5.9.2.3 `duration_threshold`

```
double gac_filter_parameter_t::duration_threshold
```

The duration threshold in milliseconds.

5.9.2.4 `fixation`

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

Fixation detection.

5.9.2.5 `gap`

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

The gap filter parameter

5.9.2.6 `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.

5.9.2.7 `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.

5.9.2.8 `noise`

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

Noise filter parameter

5.9.2.9 `saccade`

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

Saccade detection.

5.9.2.10 `sample_period`

```
double gac_filter_parameter_t::sample_period
```

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

5.9.2.11 type

`gac_filter_noise_type_t gac_filter_parameter_t::type`

The noise filter type.

5.9.2.12 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](#)

5.10 gac_filter_saccade_t Struct Reference

```
#include <gac_filter_saccade.h>
```

Data Fields

- `void * _me`
- `float velocity_threshold`
- `bool is_collecting`
- `uint32_t new_samples`
- `gac_queue_t window`

5.10.1 Detailed Description

The saccade filter structure holding filter parameters.

[gac_filter_saccade_s](#)

5.10.2 Field Documentation

5.10.2.1 _me

`void* gac_filter_saccade_t::_me`

Self-pointer to allocated structure for memory management.

5.10.2.2 is_collecting

`bool gac_filter_saccade_t::is_collecting`

A flag indicating whether a saccade is ongoing

5.10.2.3 new_samples

`uint32_t gac_filter_saccade_t::new_samples`

Counter to keep track of new items in the parent queue.

5.10.2.4 velocity_threshold

`float gac_filter_saccade_t::velocity_threshold`

The velocity threshold

5.10.2.5 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_filter_saccade.h](#)

5.11 gac_fixation_t Struct Reference

```
#include <gac_fixation.h>
```

Data Fields

- void * [_me](#)
- vec2 [screen_point](#)
- vec3 [point](#)
- double [duration](#)
- gac_sample_t [first_sample](#)

5.11.1 Detailed Description

A fixation sample.

[gac_fixation_s](#)

5.11.2 Field Documentation

5.11.2.1 [_me](#)

```
void* gac_fixation_t::_me
```

Self-pointer to allocated structure for memory management.

5.11.2.2 [duration](#)

```
double gac_fixation_t::duration
```

The fixation duration in milliseconds.

5.11.2.3 [first_sample](#)

```
gac_sample_t gac_fixation_t::first_sample
```

The first sample of the fixation.

5.11.2.4 [point](#)

```
vec3 gac_fixation_t::point
```

The fixation gaze point.

5.11.2.5 [screen_point](#)

```
vec2 gac_fixation_t::screen_point
```

The 2d fixation gaze point on the screen.

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

- [include/gac_fixation.h](#)

5.12 gac_plane_t Struct Reference

```
#include <gac_plane.h>
```

Data Fields

- void * [_me](#)
- vec3 [p1](#)
- vec3 [p2](#)
- vec3 [p3](#)
- vec3 [e1](#)
- vec3 [e2](#)
- vec3 [norm](#)
- mat4 [m](#)

5.12.1 Detailed Description

A genaral plane definition.

[gac_plane_s](#)

5.12.2 Field Documentation

5.12.2.1 [_me](#)

```
void* gac_plane_t::_me
```

Self-pointer to allocated structure for memory management.

5.12.2.2 [e1](#)

```
vec3 gac_plane_t::e1
```

The vector pointing from p1 to p2.

5.12.2.3 [e2](#)

```
vec3 gac_plane_t::e2
```

The vector pointing from p1 to p3.

5.12.2.4 [m](#)

```
mat4 gac_plane_t::m
```

Transformation matrix to transform a 3d gaze point to a 2d gaze point.

5.12.2.5 [norm](#)

```
vec3 gac_plane_t::norm
```

The normal of the screen surface.

5.12.2.6 [p1](#)

```
vec3 gac_plane_t::p1
```

A point on the plane 3d space.

5.12.2.7 [p2](#)

```
vec3 gac_plane_t::p2
```

A point on the plane 3d space.

5.12.2.8 p3

`vec3 gac_plane_t::p3`

A point on the plane 3d space.

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

- [include/gac_plane.h](#)

5.13 gac_queue_item_t Struct Reference

```
#include <gac_queue.h>
```

Data Fields

- `gac_queue_item_t * next`
- `gac_queue_item_t * prev`
- `void * data`

5.13.1 Detailed Description

A generic queue item.

[gac_queue_item_s](#)

5.13.2 Field Documentation

5.13.2.1 data

`void* gac_queue_item_t::data`

A pointer to the arbitrary data structure

5.13.2.2 next

`gac_queue_item_t* gac_queue_item_t::next`

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

5.13.2.3 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_queue.h](#)

5.14 gac_queue_t Struct Reference

```
#include <gac_queue.h>
```

Data Fields

- `void * _me`
- `gac_queue_item_t * tail`
- `gac_queue_item_t * head`
- `uint32_t count`
- `uint32_t length`
- `void(* rm)(void *)`

5.14.1 Detailed Description

A generic queue structure.

[gac_queue_s](#)

5.14.2 Field Documentation

5.14.2.1 _me

```
void* gac_queue_t::_me
```

Self-pointer to allocated structure for memory management.

5.14.2.2 count

```
uint32_t gac_queue_t::count
```

The number of occupied spaces.

5.14.2.3 head

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

A pointer to the tail to write to

5.14.2.4 length

```
uint32_t gac_queue_t::length
```

The number of total available spaces

5.14.2.5 rm

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

The handler to remove data items

5.14.2.6 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_queue.h](#)

5.15 gac_saccade_t Struct Reference

```
#include <gac_saccade.h>
```

Data Fields

- void * [_me](#)
- gac_sample_t [first_sample](#)
- gac_sample_t [last_sample](#)

5.15.1 Detailed Description

A saccade sample.

[gac_saccade_s](#)

5.15.2 Field Documentation

5.15.2.1 `_me`

```
void* gac_saccade_t::_me
```

Self-pointer to allocated structure for memory management.

5.15.2.2 `first_sample`

```
gac_sample_t gac_saccade_t::first_sample
```

The first sample of the saccade.

5.15.2.3 `last_sample`

```
gac_sample_t gac_saccade_t::last_sample
```

The last sample of the saccade.

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

- [include/gac_saccade.h](#)

5.16 `gac_sample_t` Struct Reference

```
#include <gac_sample.h>
```

Data Fields

- void * [_me](#)
- uint32_t [trial_id](#)
- vec2 [screen_point](#)
- vec3 [point](#)
- vec3 [origin](#)
- double [timestamp](#)
- double [trial_onset](#)
- double [label_onset](#)
- char [label](#) [[GAC_SAMPLE_MAX_LABEL_LEN](#)]

5.16.1 Detailed Description

The gaze data sample.

[gac_sample_s](#)

5.16.2 Field Documentation

5.16.2.1 `_me`

```
void* gac_sample_t::_me
```

Self-pointer to allocated structure for memory management.

5.16.2.2 `label`

```
char gac_sample_t::label[GAC_SAMPLE_MAX_LABEL_LEN]
```

Arbitrary label to annotate the sample.

5.16.2.3 `label_onset`

```
double gac_sample_t::label_onset
```

The time in milliseconds since the last change of label.

5.16.2.4 origin

```
vec3 gac_sample_t::origin
```

The gaze origin.

5.16.2.5 point

```
vec3 gac_sample_t::point
```

The gaze point.

5.16.2.6 screen_point

```
vec2 gac_sample_t::screen_point
```

The 2d gaze point on the screen.

5.16.2.7 timestamp

```
double gac_sample_t::timestamp
```

The sample timestamp.

5.16.2.8 trial_id

```
uint32_t gac_sample_t::trial_id
```

The ID of a ongoing trial.

5.16.2.9 trial_onset

```
double gac_sample_t::trial_onset
```

The time in milliseconds since the last change of trial ID.

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

- [include/gac_sample.h](#)

5.17 gac_screen_t Struct Reference

```
#include <gac_screen.h>
```

Data Fields

- void * [_me](#)
- float [width](#)
- float [height](#)
- float [resolution_x](#)
- float [resolution_y](#)
- vec2 [origin](#)
- gac_plane_t [plane](#)

5.17.1 Detailed Description

Screen definition of the eye tracker.

[gac_screen_s](#)

5.17.2 Field Documentation

5.17.2.1 _me

```
void* gac_screen_t::_me
```

Self-pointer to allocated structure for memory management.

5.17.2.2 height

`float gac_screen_t::height`
The height of the screen.

5.17.2.3 origin

`vec2 gac_screen_t::origin`
The screen origin in 2d space.

5.17.2.4 plane

`gac_plane_t gac_screen_t::plane`
The underlying plane definition of the screen

5.17.2.5 resolution_x

`float gac_screen_t::resolution_x`
The width of the screen resolution.

5.17.2.6 resolution_y

`float gac_screen_t::resolution_y`
The height of the screen resolution.

5.17.2.7 width

`float gac_screen_t::width`
The width of the screen.
The documentation for this struct was generated from the following file:

- [include/gac_screen.h](#)

5.18 gac_t Struct Reference

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

Data Fields

- `void * _me`
- `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`
- `gac_screen_t * screen`
- `gac_sample_t * last_sample`
- `double trial_timestamp`
- `double label_timestamp`
- `gac_aoi_collection_t aoic`

5.18.1 Detailed Description

The gaze analysis handler structure.
[gac_s](#)

5.18.2 Field Documentation

5.18.2.1 _me

`void* gac_t::_me`

Self-pointer to allocated structure for memory management.

5.18.2.2 aoic

`gac_aoi_collection_t gac_t::aoic`

The AOI collection structure to handle AOIs.

5.18.2.3 fixation

`gac_filter_fixation_t gac_t::fixation`

The fixation filter structure

5.18.2.4 gap

`gac_filter_gap_t gac_t::gap`

The gap filter structure

5.18.2.5 label_timestamp

`double gac_t::label_timestamp`

The timestamp of the last label change.

5.18.2.6 last_sample

`gac_sample_t* gac_t::last_sample`

The last sample entered to the window. This remains even if the sample window is cleared.

5.18.2.7 noise

`gac_filter_noise_t gac_t::noise`

The noise filter structure

5.18.2.8 parameter

`gac_filter_parameter_t gac_t::parameter`

The parameters passed during configuration

5.18.2.9 saccade

`gac_filter_saccade_t gac_t::saccade`

The saccade filter structure

5.18.2.10 samples

`gac_queue_t gac_t::samples`

The sample queue

5.18.2.11 screen

`gac_screen_t* gac_t::screen`

The screen information.

5.18.2.12 trial_timestamp

`double gac_t::trial_timestamp`

The timestamp of the last trial ID change.

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

- [include/gac.h](#)

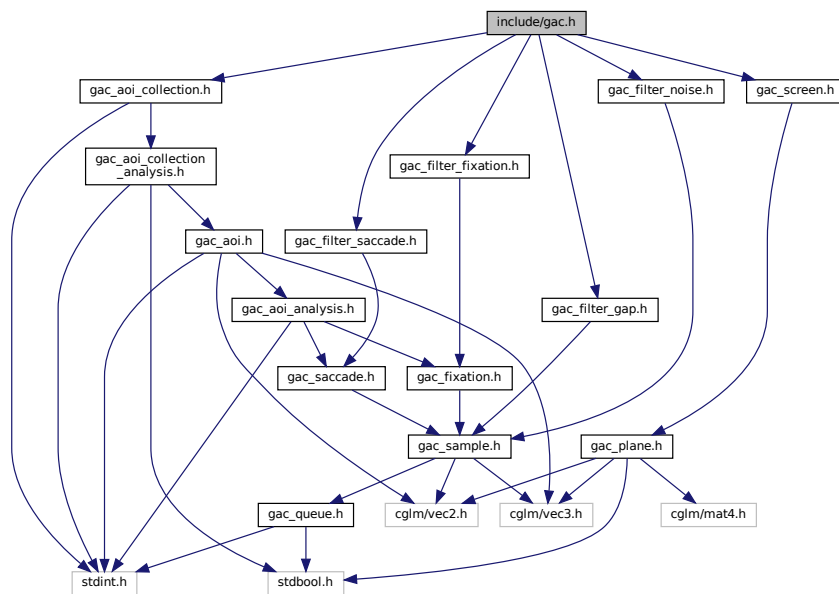
Chapter 6

File Documentation

6.1 include/gac.h File Reference

```
#include "gac_aoi_collection.h"
#include "gac_filter_fixation.h"
#include "gac_filter_gap.h"
#include "gac_filter_noise.h"
#include "gac_filter_saccade.h"
#include "gac_screen.h"
```

Include dependency graph for gac.h:



Data Structures

- struct [gac_filter_parameter_t](#)
- struct [gac_t](#)

Functions

- bool [gac_add_aoi](#) ([gac_t](#) *h, [gac_aoi_t](#) *aoi)
- [gac_t](#) * [gac_create](#) ([gac_filter_parameter_t](#) *parameter)
- void [gac_destroy](#) ([gac_t](#) *h)

- bool [gac_finalise](#) (gac_t *h, gac_aoi_collection_analysis_result_t *analysis)
- 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_set_screen](#) (gac_t *h, float top_left_x, float top_left_y, float top_left_z, float top_right_x, float top_right_y, float top_right_z, float bottom_left_x, float bottom_left_y, float bottom_left_z)
- 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)
- uint32_t [gac_sample_window_update](#) (gac_t *h, float ox, float oy, float oz, float px, float py, float pz, double timestamp, uint32_t trial_id, const char *label)
- uint32_t [gac_sample_window_update_vec](#) (gac_t *h, vec2 *screen_point, vec3 *origin, vec3 *point, double timestamp, uint32_t trial_id, const char *label)
- uint32_t [gac_sample_window_update_screen](#) (gac_t *h, float ox, float oy, float oz, float px, float py, float pz, float sx, float sy, double timestamp, uint32_t trial_id, const char *label)
- const char * [gac_version](#) ()

6.1.1 Detailed Description

Gaze analysis library for fixation and saccade detection in raw gaze data.

[gac.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.1.2 Function Documentation

6.1.2.1 gac_add_aoi()

```
bool gac_add_aoi (
    gac_t * h,
    gac_aoi_t * aoi )
```

Allows to add an AOI to the gaze analysis handler. This enables the AOI analysis.

Parameters

<i>h</i>	A pointer to the gaze analysis handler.
<i>aoi</i>	A pointer to the AOI structure to add.

Returns

True on success, false otherwise.

6.1.2.2 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.

6.1.2.3 gac_destroy()

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

Destroy the gaze analysis handler.

Parameters

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

6.1.2.4 gac_finalise()

```
bool gac_finalise (
    gac_t * h,
    gac_aoi_collection_analysis_result_t * analysis )
```

Finalise the AOI analysis.

Parameters

<i>h</i>	A pointer to the gaze analysis handler.
<i>analysis</i>	A location to store the analysis result. This structure is only valid if the function returns true.

Returns

True on success, false otherwise.

6.1.2.5 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.

6.1.2.6 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.

6.1.2.7 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.

6.1.2.8 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.

6.1.2.9 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.

6.1.2.10 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.

6.1.2.11 gac_sample_window_update()

```
uint32_t gac_sample_window_update (
    gac_t * h,
    float ox,
    float oy,
    float oz,
    float px,
    float py,
    float pz,
    double timestamp,
    uint32_t trial_id,
    const char * label )
```

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.
<i>trial_id</i>	The ID of the ongoing trial.
<i>label</i>	An optional arbitrary label annotating the sample.

Returns

The number of new samples added to the window.

6.1.2.12 `gac_sample_window_update_screen()`

```
uint32_t gac_sample_window_update_screen (
    gac_t * h,
    float ox,
    float oy,
    float oz,
    float px,
    float py,
    float pz,
    float sx,
    float sy,
    double timestamp,
    uint32_t trial_id,
    const char * label )
```

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>sx</i>	The x coordinate of the screen gaze point.
<i>sy</i>	The y coordinate of the screen gaze point.
<i>timestamp</i>	The timestamp of the sample.
<i>trial_id</i>	The ID of the ongoing trial.
<i>label</i>	An optional arbitrary label annotating the sample.

Returns

The number of new samples added to the window.

6.1.2.13 gac_sample_window_update_vec()

```
uint32_t gac_sample_window_update_vec (
    gac_t * h,
    vec2 * screen_point,
    vec3 * origin,
    vec3 * point,
    double timestamp,
    uint32_t trial_id,
    const char * label )
```

Update sample window with a new sample.

Parameters

<i>h</i>	A pointer to the gaze analysis handler.
<i>screen_point</i>	The 2d screen gaze point
<i>origin</i>	The gaze origin.
<i>point</i>	The gaze point.
<i>timestamp</i>	The timestamp of the sample.
<i>trial_id</i>	The ID of the ongoing trial.
<i>label</i>	An optional arbitrary label annotating the sample.

Returns

The number of new samples added to the window.

6.1.2.14 gac_set_screen()

```
bool gac_set_screen (
    gac_t * h,
    float top_left_x,
    float top_left_y,
    float top_left_z,
    float top_right_x,
    float top_right_y,
    float top_right_z,
    float bottom_left_x,
    float bottom_left_y,
    float bottom_left_z )
```

Configure the screen position in 3d space. This allows to compute normalized 2d gaze point coordinates.

Parameters

<i>h</i>	A pointer to the gaze analysis handler.
<i>top_left_x</i>	The x coordinate of the top left screen corner.
<i>top_left_y</i>	The y coordinate of the top left screen corner.
<i>top_left_z</i>	The z coordinate of the top left screen corner.
<i>top_right_x</i>	The x coordinate of the top right screen corner.
<i>top_right_y</i>	The y coordinate of the top right screen corner.

Parameters

<i>top_right_z</i>	The z coordinate of the top right screen corner.
<i>bottom_left_x</i>	The x coordinate of the bottom left screen corner.
<i>bottom_left_y</i>	The y coordinate of the bottom left screen corner.
<i>bottom_left_z</i>	The z coordinate of the bottom left screen corner.

Returns

True on success, false on failure.

6.1.2.15 gac_version()

```
const char* gac_version ( )
```

Returns the version of the library.

Returns

A version number string of the form <major>.<minor>.<revision>.

6.2 include/gac_aoi.h File Reference

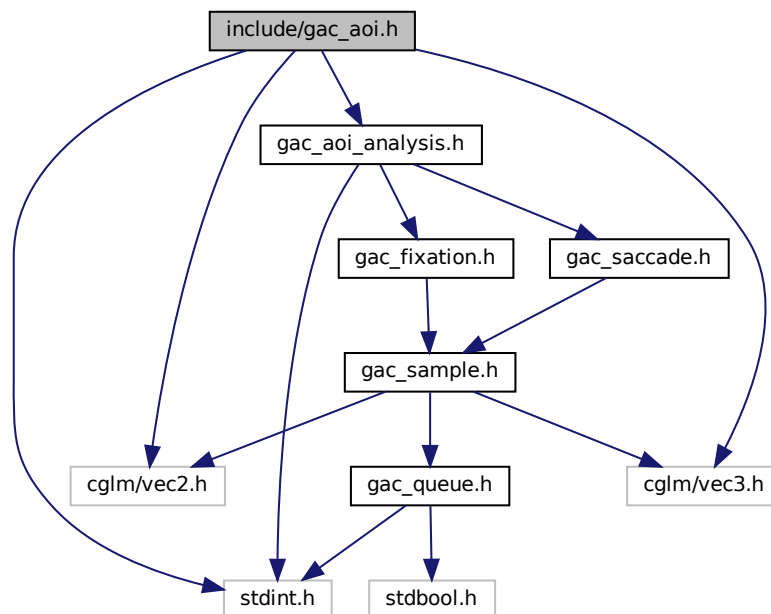
```
#include "gac_aoi_analysis.h"
```

```
#include <stdint.h>
```

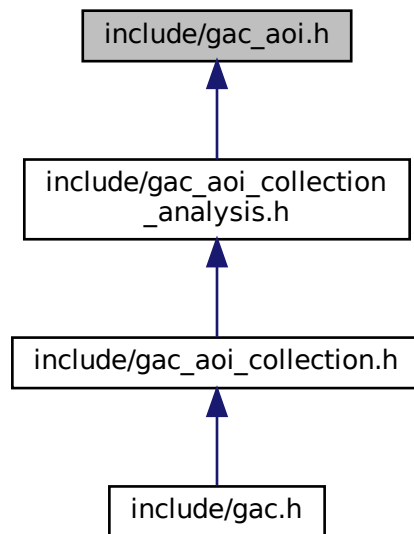
```
#include <cglm/vec2.h>
```

```
#include <cglm/vec3.h>
```

Include dependency graph for gac_aoi.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_aoi_t](#)

Macros

- #define [GAC_AOI_MAX](#) 100
- #define [GAC_AOI_MAX_POINTS](#) 100
- #define [GAC_AOI_MAX_LABEL_LEN](#) 100

Typedefs

- typedef enum [gac_aoi_orientation_e](#) [gac_aoi_orientation_t](#)

Enumerations

- enum [gac_aoi_orientation_e](#) { [GAC_AOI_ORIENTATION_COLINEAR](#), [GAC_AOI_ORIENTATION_CLOCKWISE](#), [GAC_AOI_ORIENTATION_COUNTER_CLOCKWISE](#) }

Functions

- bool [gac_aoi_add_point](#) ([gac_aoi_t](#) *aoi, float x, float y)
- bool [gac_aoi_add_point_res](#) ([gac_aoi_t](#) *aoi, float x_res, float y_res)
- bool [gac_aoi_add_rect](#) ([gac_aoi_t](#) *aoi, float x, float y, float width, float height)
- bool [gac_aoi_add_rect_res](#) ([gac_aoi_t](#) *aoi, float x, float y, float width, float height)
- [gac_aoi_t](#) * [gac_aoi_copy](#) ([gac_aoi_t](#) *aoi)
- bool [gac_aoi_copy_to](#) ([gac_aoi_t](#) *tgt, [gac_aoi_t](#) *src)
- [gac_aoi_t](#) * [gac_aoi_create](#) (const char *label)
- void [gac_aoi_destroy](#) ([gac_aoi_t](#) *aoi)
- bool [gac_aoi_includes_point](#) ([gac_aoi_t](#) *aoi, float x, float y)

- bool [gac_aoi_includes_point_res](#) (gac_aoi_t *aoi, float x_res, float y_res)
- bool [gac_aoi_init](#) (gac_aoi_t *aoi, const char *label)
- bool [gac_aoi_intersect](#) (vec2 *p1, vec2 *q1, vec2 *p2, vec2 *q2)
- bool [gac_aoi_point_on_segment](#) (vec2 *p, vec2 *s1, vec2 *s2)
- [gac_aoi_orientation_t gac_aoi_orientation_triplet](#) (vec2 *p, vec2 *q, vec2 *r)
- bool [gac_aoi_set_resolution](#) (gac_aoi_t *aoi, float resolution_x, float resolution_y)

6.2.1 Detailed Description

Area of interest (AOI) structure and helper functions.

[gac_aoi.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.2.2 Macro Definition Documentation

6.2.2.1 GAC_AOI_MAX

```
#define GAC_AOI_MAX 100
```

The maximal allowed area of intersts to analyse.

6.2.2.2 GAC_AOI_MAX_LABEL_LEN

```
#define GAC_AOI_MAX_LABEL_LEN 100
```

The maximal label length

6.2.2.3 GAC_AOI_MAX_POINTS

```
#define GAC_AOI_MAX_POINTS 100
```

The maximal allowed points definig an area of interest.

6.2.3 Typedef Documentation

6.2.3.1 gac_aoi_orientation_t

```
typedef enum gac\_aoi\_orientation\_e gac\_aoi\_orientation\_t
gac\_aoi\_orientation\_e
```

6.2.4 Enumeration Type Documentation

6.2.4.1 gac_aoi_orientation_e

```
enum gac\_aoi\_orientation\_e
```

The order of point triplets. This is used for checking whetehr a point lies within an AOI.

Enumerator

GAC_AOI_ORIENTATION_COLINEAR	Points are colinear.
GAC_AOI_ORIENTATION_CLOCKWISE	Points are ordered clockwise.
GAC_AOI_ORIENTATION_COUNTER_CLOCKWISE	Points are ordered counter clockwise.

6.2.5 Function Documentation

6.2.5.1 gac_aoi_add_point()

```
bool gac_aoi_add_point (
    gac_aoi_t * aoi,
    float x,
    float y )
```

Add a point the AOE definition. An AOE requires at least 3 points to be valid. In addition to attaching the point to the internal array, this function computes a point which is guaranteed to be outside of the AOI at a reasonable distance from the AOI.

Parameters

<i>aoi</i>	A pointer to an AOI structure.
<i>x</i>	The normalised x coordinate of the AOI point to add.
<i>y</i>	The normalised y coordinate of the AOI point to add.

Returns

True on success, false on failure.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.2.5.2 gac_aoi_add_point_res()

```
bool gac_aoi_add_point_res (
    gac_aoi_t * aoi,
    float x_res,
    float y_res )
```

The same as [gac_aoi_add_point\(\)](#) but accepting the input coordinates in pixels instead of normalized values. Note that this function will always return false if [gac_aoi_set_resolution\(\)](#) was never called.

6.2.5.3 gac_aoi_add_rect()

```
bool gac_aoi_add_rect (
    gac_aoi_t * aoi,
    float x,
    float y,
    float width,
    float height )
```

Add four points describing a rectangle to the AOI, given the top left point, a width and a height.

Parameters

<i>aoi</i>	A pointer to the AOI structure.
<i>x</i>	The normalized x coordinate of the top left point of the rectangle.
<i>y</i>	The normalized y coordinate of the top left point of the rectangle.
<i>width</i>	The normalized width of the recatngle.
<i>height</i>	The normalized height of the recatngle.

Returns

True on success, false on failure.

6.2.5.4 gac_aoi_add_rect_res()

```
bool gac_aoi_add_rect_res (
    gac_aoi_t * aoi,
    float x,
    float y,
    float width,
    float height )
```

The same as [gac_aoi_add_rect\(\)](#) but accepting the input coordinates in pixels instead of normalized values. Note that this function will always return false if [gac_aoi_set_resolution\(\)](#) was never called.

6.2.5.5 gac_aoi_copy()

```
gac_aoi_t* gac_aoi_copy (
    gac_aoi_t * aoi )
```

Create a deep copy of an AOI.

Parameters

<i>aoi</i>	A pointer to the AOI to be copied.
------------	------------------------------------

Returns

A newly allocated copy of the input AOI.

6.2.5.6 gac_aoi_copy_to()

```
bool gac_aoi_copy_to (
    gac_aoi_t * tgt,
    gac_aoi_t * src )
```

Copy an AOI structure.

Parameters

<i>tgt</i>	A pointer to an AOI to copy to.
<i>src</i>	A pointer to the AOI to be copied.

Returns

True on success, false otherwise.

6.2.5.7 gac_aoi_create()

```
gac_aoi_t* gac_aoi_create (
    const char * label )
```

Allocate a new AOI structure. This must be freed with [gac_aoi_destroy\(\)](#).

Parameters

<i>label</i>	An arbitrary label, describing the AOI.
--------------	---

Returns

A pointer to the allocated structure or NULL on failure.

6.2.5.8 gac_aoi_destroy()

```
void gac_aoi_destroy (
    gac_aoi_t * aoi )
```

Destroys a AOI structure. This works for structures created with [gac_aoi_create\(\)](#) as well as [gac_aoi_init\(\)](#).

Parameters

<i>aoi</i>	A pointer to a AOI structure to destroy.
------------	--

6.2.5.9 gac_aoi_includes_point()

```
bool gac_aoi_includes_point (
    gac_aoi_t * aoi,
    float x,
    float y )
```

Checks whether a point is inside of an AOI. This function uses the ray casting method where a virtual ray is drawn from an arbitrary point outside the AOI to the point. Then, every intersection with segments of the AOI contour is counted. If an even number of intersection is detected, the point lies outside of the AOI, otherwise the point lies inside the AOI.

Parameters

<i>aoi</i>	A pointer to an AOI structure.
<i>x</i>	The normalised x coordinate of the point to check.
<i>y</i>	The normalised y coordinate of the point to check.

Returns

True if the point is inside the AOI, false otherwise.

6.2.5.10 gac_aoi_includes_point_res()

```
bool gac_aoi_includes_point_res (
    gac_aoi_t * aoi,
    float x_res,
    float y_res )
```

The same as [gac_aoi_includes_point\(\)](#) but accepting the input coordinates in pixels instead of normalized values. Note that this function will always return false if [gac_aoi_set_resolution\(\)](#) was never called.

6.2.5.11 gac_aoi_init()

```
bool gac_aoi_init (
    gac_aoi_t * aoi,
    const char * label )
```

Initialise the AOI structure.

Parameters

<i>aoi</i>	A pointer to the aoi structure to initialise.
<i>label</i>	An arbitrary label, describing the AOI.

Returns

True on success, false otherwise.

6.2.5.12 gac_aoi_intersect()

```
bool gac_aoi_intersect (
    vec2 * p1,
    vec2 * q1,
    vec2 * p2,
    vec2 * q2 )
```

Checks whether the line segment p1q1 intersects with the line segment p2q2.

Parameters

<i>p1</i>	A pointer to the starting point of the first segment.
<i>q1</i>	A pointer to the end point of the first segment.
<i>p2</i>	A pointer to the starting point of the second segment.
<i>q2</i>	A pointer to the end point of the second segment.

Returns

True if the two segments intersect, false otherwise.

6.2.5.13 gac_aoi_orientation_triplet()

```
gac_aoi_orientation_t gac_aoi_orientation_triplet (
    vec2 * p,
    vec2 * q,
    vec2 * r )
```

Given three ordered points p, q, and r, this function detects whether the points are colinear, ordered clockwise or counter clockwise.

Parameters

<i>p</i>	A pointer to point p.
<i>q</i>	A pointer to point q.
<i>r</i>	A pointer to point r.

Returns

The orientation of the three points.

6.2.5.14 gac_aoi_point_on_segment()

```
bool gac_aoi_point_on_segment (
    vec2 * p,
    vec2 * s1,
    vec2 * s2 )
```

Given three colinear points, this function checks if a point *p* lies on a segment *s1s2*.

Parameters

<i>p</i>	A pointer to the point to check.
<i>s1</i>	The starting point of the segment.
<i>s2</i>	The end point of the segment.

Returns

True if the point lies on the segment, false otherwise.

6.2.5.15 gac_aoi_set_resolution()

```
bool gac_aoi_set_resolution (
    gac_aoi_t * aoi,
    float resolution_x,
    float resolution_y )
```

Set the screen resolution. This allows to use all functions with an *res* suffix. These functions will act exactly like their counter part function without the *res* suffix but use 2d points expressed in the screen resolution.

Parameters

<i>aoi</i>	A pointer to an aoi structure.
<i>resolution</i> ↔ <i>_x</i>	The width of the screen resolution.
<i>resolution</i> ↔ <i>_y</i>	The height of the screen resolution.

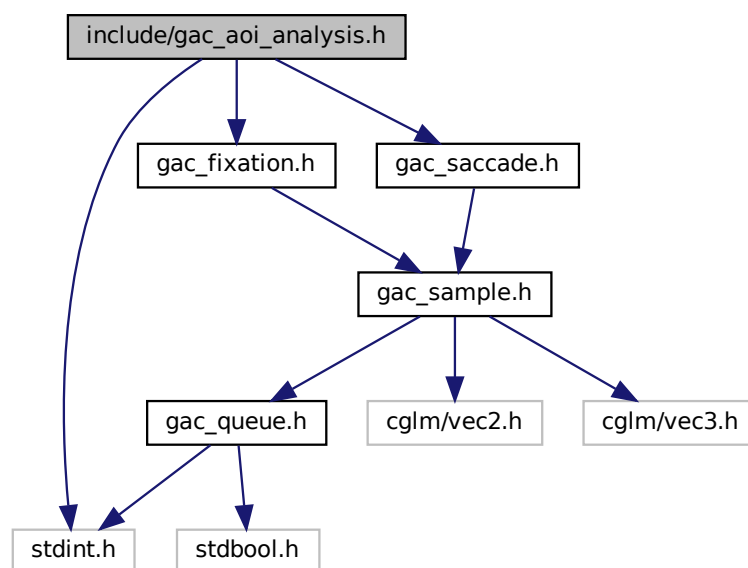
Returns

True on success, false on failure.

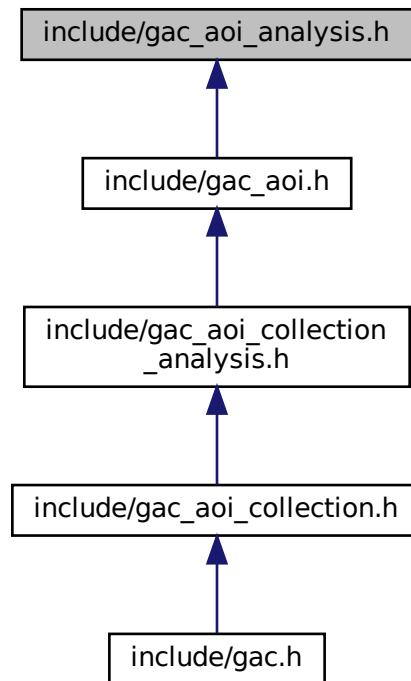
6.3 include/gac_aoi_analysis.h File Reference

```
#include <stdint.h>
#include "gac_fixation.h"
#include "gac_saccade.h"
```

Include dependency graph for gac_aoi_analysis.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_aoi_analysis_t](#)

Functions

- bool [gac_aoi_analysis_clear](#) (gac_aoi_analysis_t *analysis)
- gac_aoi_analysis_t * [gac_aoi_analysis_copy](#) (gac_aoi_analysis_t *analysis)
- bool [gac_aoi_analysis_copy_to](#) (gac_aoi_analysis_t *tgt, gac_aoi_analysis_t *src)
- gac_aoi_analysis_t * [gac_aoi_analysis_create](#) ()
- void [gac_aoi_analysis_destroy](#) (gac_aoi_analysis_t *analysis)
- bool [gac_aoi_analysis_init](#) (gac_aoi_analysis_t *analysis)

6.3.1 Detailed Description

The analysis structure definition of an AOI.
[gac_aoi_analysis.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.3.2 Function Documentation

6.3.2.1 gac_aoi_analysis_clear()

```
bool gac_aoi_analysis_clear (
    gac_aoi_analysis_t * analysis )
```

Clear an AIO analysis structure.

Parameters

<i>analysis</i>	A pointer to the structure to clear.
-----------------	--------------------------------------

Returns

True on success, false on failure.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.3.2.2 gac_aoi_analysis_copy()

```
gac_aoi_analysis_t* gac_aoi_analysis_copy (
    gac_aoi_analysis_t * analysis )
```

Create a deep copy of the AOI analysis structure.

Parameters

<i>analysis</i>	A pointer to the analysis structure to be copied.
-----------------	---

Returns

A newly allocated copy of the input structure.

6.3.2.3 gac_aoi_analysis_copy_to()

```
bool gac_aoi_analysis_copy_to (
    gac_aoi_analysis_t * tgt,
    gac_aoi_analysis_t * src )
```

Copy an AOI analysis structure.

Parameters

<i>tgt</i>	A pointer to the analysis structure to copy to.
<i>src</i>	A pointer to the analysis structure to be copied.

Returns

True on success, false otherwise.

6.3.2.4 gac_aoi_analysis_create()

```
gac_aoi_analysis_t* gac_aoi_analysis_create ( )
```

Allocate a new AOI analysis structure on the heap. This needs to be freed with [gac_aoi_analysis_destroy\(\)](#).

Returns

A pointer to the newly allocated structure.

6.3.2.5 gac_aoi_analysis_destroy()

```
void gac_aoi_analysis_destroy (
    gac_aoi_analysis_t * analysis )
```

Destroy an AOI analysis structure.

Parameters

<i>analysis</i>	A pointer to the analysis structure to destroy.
-----------------	---

6.3.2.6 gac_aoi_analysis_init()

```
bool gac_aoi_analysis_init (
    gac_aoi_analysis_t * analysis )
```

Initialise an AIO analysis structure.

Parameters

<i>analysis</i>	A pointer to the structure to initialise.
-----------------	---

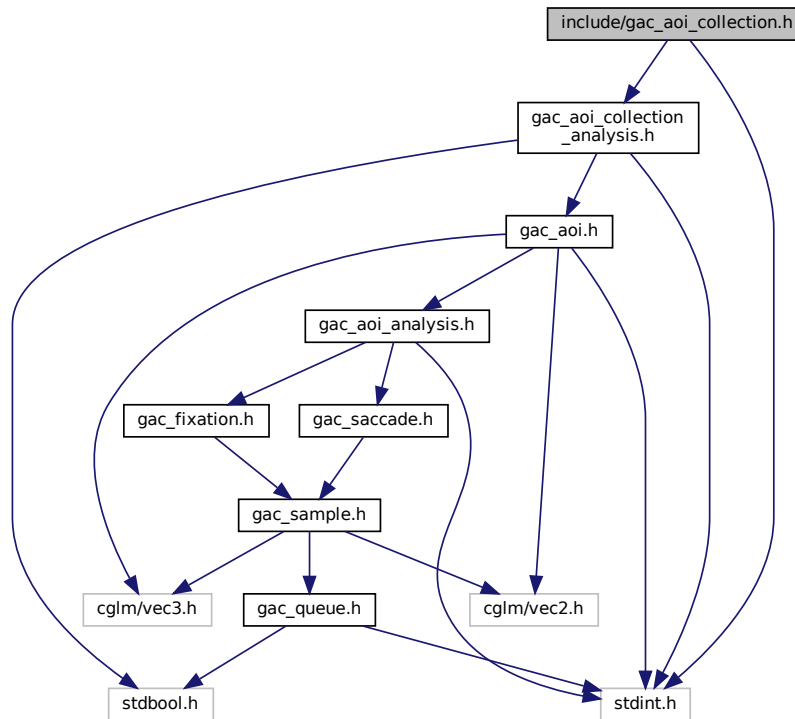
Returns

True on success, false on failure.

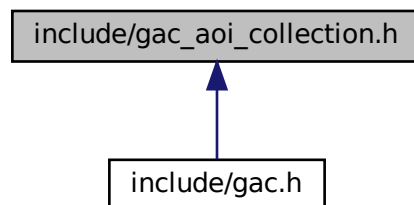
6.4 include/gac_aoi_collection.h File Reference

```
#include "gac_aoi_collection_analysis.h"
#include <stdint.h>
```

Include dependency graph for `gac_aoi_collection.h`:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_aoi_collection_t](#)

Functions

- bool [gac_aoi_collection_add](#) ([gac_aoi_collection_t](#) *aoic, [gac_aoi_t](#) *aoi)
- bool [gac_aoi_collection_analyse_clear](#) ([gac_aoi_collection_t](#) *aoic)
- bool [gac_aoi_collection_analyse_finalise](#) ([gac_aoi_collection_t](#) *aoic, [gac_aoi_collection_analysis_result_t](#) *analysis)

- bool [gac_aoi_collection_analyse_fixation](#) (gac_aoi_collection_t *aoic, gac_fixation_t *fixation, gac_aoi_collection_analysis_result_t *analysis)
- bool [gac_aoi_collection_analyse_saccade](#) (gac_aoi_collection_t *aoic, gac_saccade_t *saccade)
- bool [gac_aoi_collection_assign](#) (gac_aoi_collection_t *aoic, gac_aoi_t *aoi)
- gac_aoi_collection_t * [gac_aoi_collection_create](#) ()
- void [gac_aoi_collection_destroy](#) (gac_aoi_collection_t *aoic)
- bool [gac_aoi_collection_init](#) (gac_aoi_collection_t *aoic)

6.4.1 Detailed Description

The AOI collection structure and associated functions. This is used to aggregate information during the AOI analysis. [gac_aoi_collection.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.4.2 Function Documentation

6.4.2.1 gac_aoi_collection_add()

```
bool gac_aoi_collection_add (
    gac_aoi_collection_t * aoic,
    gac_aoi_t * aoi )
```

Add an AOI to an AOI collection. Do **not** destroy an AOI which was added to the collection. Memory management is taken care of by the collection.

Parameters

<i>aoic</i>	A pointer to the AOI collection
<i>aoi</i>	A pointer to the AOI to add.

Returns

True on success, false otherwise.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.4.2.2 gac_aoi_collection_analyse_clear()

```
bool gac_aoi_collection_analyse_clear (
    gac_aoi_collection_t * aoic )
```

Clear all analysis structures of all AOIS in the AOI collection.

Parameters

<i>aoic</i>	A pointer to the AOI collection structure to clear.
-------------	---

Returns

True on success, false otherwise.

6.4.2.3 gac_aoi_collection_analyse_finalise()

```
bool gac_aoi_collection_analyse_finalise (
    gac_aoi_collection_t * aoic,
    gac_aoi_collection_analysis_result_t * analysis )
```

Finalise the AOI analysis. This function computes the relative values in each AOI structure based on the collection analysis data.

Parameters

<i>aoic</i>	A pointer to the AOI collection.
<i>analysis</i>	A location to store the analysis result. This structure is only valid if the function returns true.

Returns

True on success, false otherwise.

6.4.2.4 gac_aoi_collection_analyse_fixation()

```
bool gac_aoi_collection_analyse_fixation (
    gac_aoi_collection_t * aoic,
    gac_fixation_t * fixation,
    gac_aoi_collection_analysis_result_t * analysis )
```

Add a fixation to the AOI collection and update the analysis.

Parameters

<i>aoic</i>	A pointer to an AOI collection.
<i>fixation</i>	The fixation point to add.
<i>analysis</i>	A location to store the analysis result. This structure is only valid if the function returns true.

Returns

True on success, false on failure.

6.4.2.5 gac_aoi_collection_analyse_saccade()

```
bool gac_aoi_collection_analyse_saccade (
    gac_aoi_collection_t * aoic,
    gac_saccade_t * saccade )
```

Add a saccade to the AOI collection and update the analysis. Note that this only extends the AOI analysis but no AOI can happen based on saccades only. Always call this function before fixation analysis (see `gac_aoi_collection_analyse_fixation`).

Parameters

<i>aoic</i>	A pointer to an AOI collection.
<i>saccade</i>	The saccade point to add.

Returns

True on success, false on failure.

6.4.2.6 gac_aoi_collection_assign()

```
bool gac_aoi_collection_assign (
    gac_aoi_collection_t * aoic,
    gac_aoi_t * aoi )
```

Assign a new AOI to an AOI collection. This function acts similar to [gac_aoi_collection_add\(\)](#) but only creates a copy if the AOI to assign is allocated on the stack. If a heap allocated AOI is assigned the AOI is not copied and must, therefore, no longer be freed as it will be freed automatically with [gac_aoi_collection_destroy\(\)](#).

Parameters

<i>aoic</i>	A pointer to an AOI collection.
<i>aoi</i>	A pointer to the AOI to be assigned.

Returns

True on success, false otherwise.

6.4.2.7 gac_aoi_collection_create()

```
gac_aoi_collection_t* gac_aoi_collection_create ( )
```

Allocate a new AOI collection on the heap.

Returns

A pointer to the newly allocated AOI collection.

6.4.2.8 gac_aoi_collection_destroy()

```
void gac_aoi_collection_destroy (
    gac_aoi_collection_t * aoic )
```

Destroy an AOI collection.

Parameters

<i>aoic</i>	Destroy an AOI collection.
-------------	----------------------------

6.4.2.9 gac_aoi_collection_init()

```
bool gac_aoi_collection_init (
    gac_aoi_collection_t * aoic )
```

Initialise an AOI collection.

Parameters

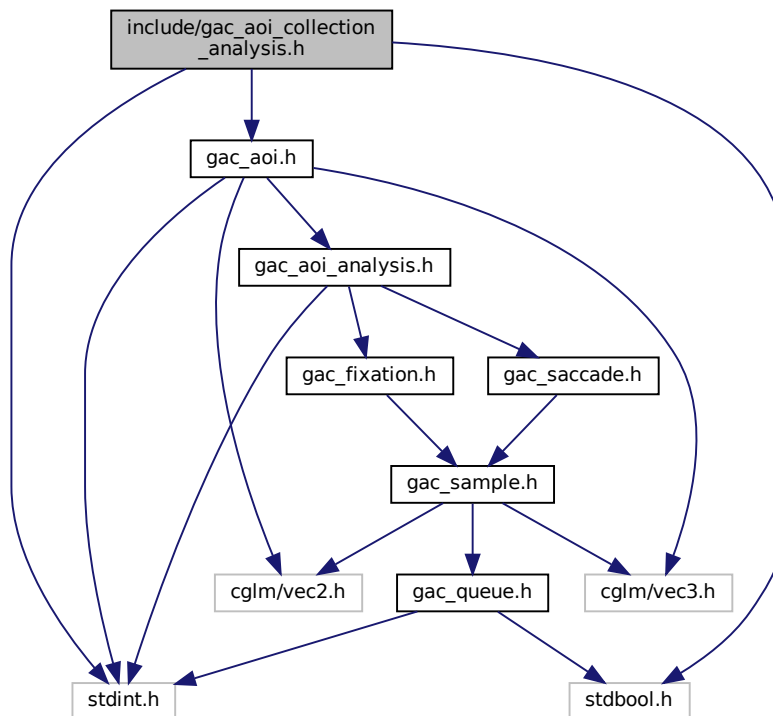
<i>aoic</i>	A pointer to an AOI collection to initialise.
-------------	---

Returns

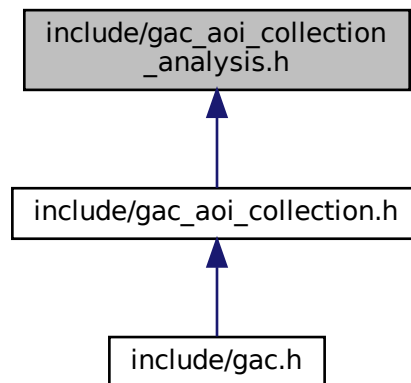
True on success, false otherwise.

6.5 include/gac_aoi_collection_analysis.h File Reference

```
#include "gac_aoi.h"  
#include <stdint.h>  
#include <stdbool.h>  
Include dependency graph for gac_aoi_collection_analysis.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_aoi_collection_analysis_t](#)
- struct [gac_aoi_collection_analysis_result_t](#)

Functions

- bool [gac_aoi_collection_analysis_clear](#) (gac_aoi_collection_analysis_t *analysis)
- gac_aoi_collection_analysis_t * [gac_aoi_collection_analysis_create](#) ()
- void [gac_aoi_collection_analysis_destroy](#) (gac_aoi_collection_analysis_t *analysis)
- bool [gac_aoi_collection_analysis_init](#) (gac_aoi_collection_analysis_t *analysis)

6.5.1 Detailed Description

The analysis structure definition of an AOI collection.

[gac_aoi_collection_analysis.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.5.2 Function Documentation

6.5.2.1 gac_aoi_collection_analysis_clear()

```
bool gac_aoi_collection_analysis_clear (  
    gac_aoi_collection_analysis_t * analysis )
```

Clear the analysis structure.

Parameters

<i>analysis</i>	A pointer to the AOI collection analysis structure to clear.
-----------------	--

Returns

True on success, false otherwise.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.5.2.2 gac_aoi_collection_analysis_create()

```
gac_aoi_collection_analysis_t* gac_aoi_collection_analysis_create ( )
```

Allocate a new AOI collection analysis structure on the heap.

Returns

A pointer to the newly allocated structure.

6.5.2.3 gac_aoi_collection_analysis_destroy()

```
void gac_aoi_collection_analysis_destroy (
    gac_aoi_collection_analysis_t * analysis )
```

Destroy an AOI collection analysis structure.

Parameters

<i>analysis</i>	A pointer to the AOI collection analysis structure to be destroyed.
-----------------	---

6.5.2.4 gac_aoi_collection_analysis_init()

```
bool gac_aoi_collection_analysis_init (
    gac_aoi_collection_analysis_t * analysis )
```

Initialise an AOI collection analysis structure.

Parameters

<i>analysis</i>	A pointer to the AOI collection analysis structure to initialise.
-----------------	---

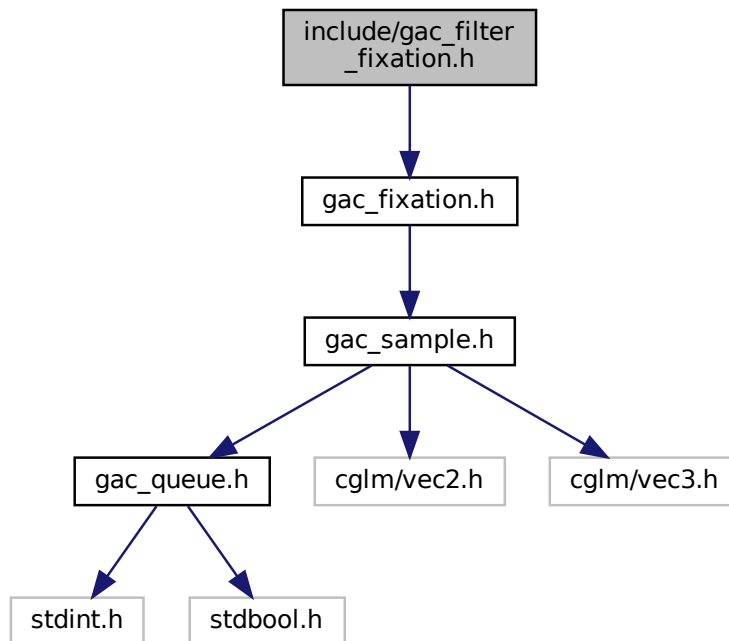
Returns

True on success, false otherwise.

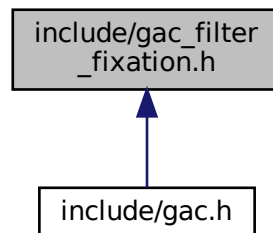
6.6 include/gac_filter_fixation.h File Reference

```
#include "gac_fixation.h"
```

Include dependency graph for gac_filter_fixation.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_filter_fixation_t](#)

Functions

- 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)

6.6.1 Detailed Description

Gaze analysis fixation filter implementation.

[gac_filter_fixation.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.6.2 Function Documentation

6.6.2.1 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.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.6.2.2 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 threshold in degrees.
<i>duration_threshold</i>	The duration threshold in milliseconds.

Returns

The allocated fixation filter structure or NULL on failure.

6.6.2.3 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.
---------------	--

6.6.2.4 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.

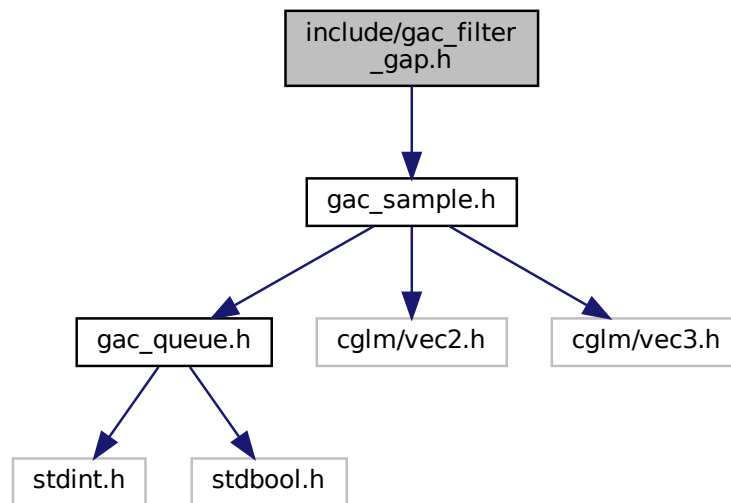
Returns

True on success, false on failure.

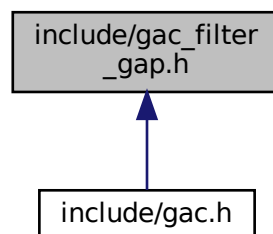
6.7 include/gac_filter_gap.h File Reference

```
#include "gac_sample.h"
```

Include dependency graph for `gac_filter_gap.h`:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_filter_gap_t](#)

Functions

- 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)

6.7.1 Detailed Description

Gaze analysis gap filter implementation.

[gac_filter_gap.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.7.2 Function Documentation**6.7.2.1 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.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.7.2.2 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.

6.7.2.3 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.
---------------	--

6.7.2.4 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).

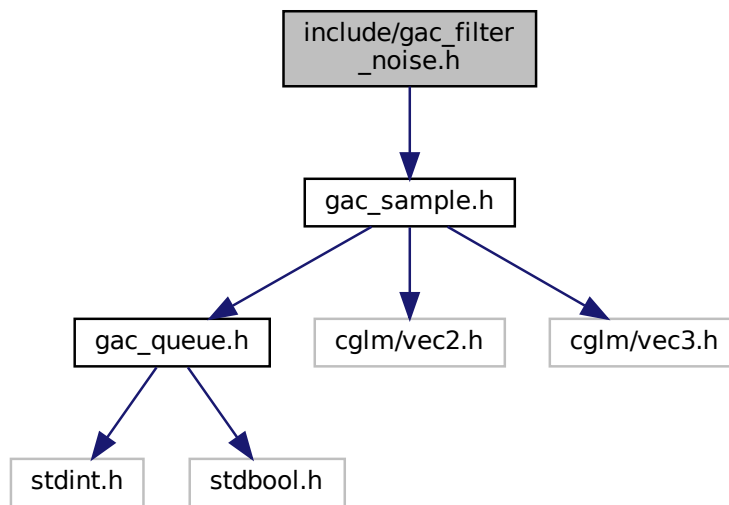
Returns

True on success, false on failure.

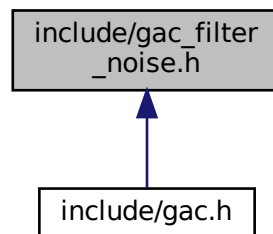
6.8 include/gac_filter_noise.h File Reference

```
#include "gac_sample.h"
```

Include dependency graph for gac_filter_noise.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_filter_noise_t](#)

Typedefs

- typedef enum [gac_filter_noise_type_e](#) [gac_filter_noise_type_t](#)

Enumerations

- enum [gac_filter_noise_type_e](#) { `GAC_FILTER_NOISE_TYPE_AVERAGE`, `GAC_FILTER_NOISE_TYPE_MEDIAN` }

Functions

- `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`)

6.8.1 Detailed Description

Gaze analysis noise filter implementation.

[gac_filter_noise.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.8.2 Typedef Documentation

6.8.2.1 `gac_filter_noise_type_t`

```
typedef enum gac_filter_noise_type_e gac_filter_noise_type_t
gac_filter_noise_type_e
```

6.8.3 Enumeration Type Documentation

6.8.3.1 `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

6.8.4 Function Documentation

6.8.4.1 `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.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.8.4.2 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.

6.8.4.3 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.

6.8.4.4 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.
---------------	--

6.8.4.5 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.

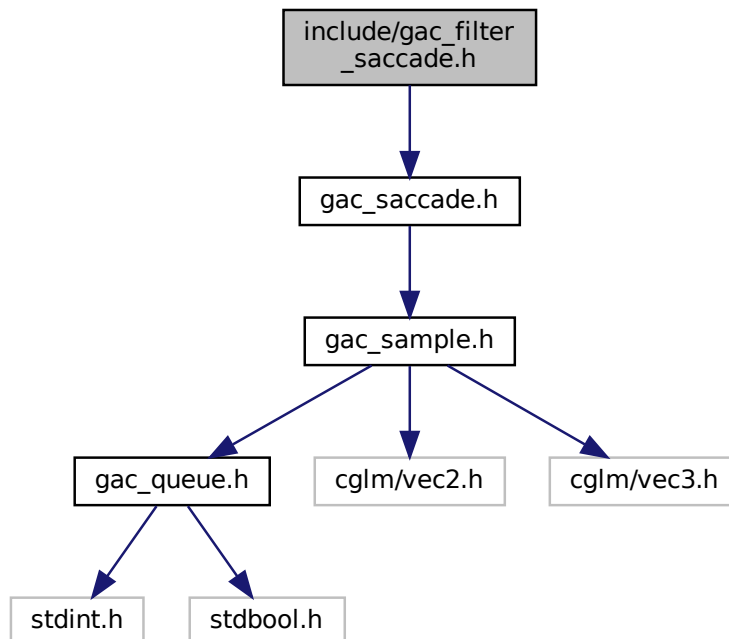
Returns

True on success, false on failure.

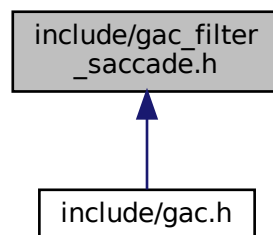
6.9 include/gac_filter_saccade.h File Reference

```
#include "gac_saccade.h"
```

Include dependency graph for gac_filter_saccade.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_filter_saccade_t](#)

Functions

- 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)

6.9.1 Detailed Description

Gaze analysis saccade filter implementation.

[gac_filter_saccade.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.9.2 Function Documentation

6.9.2.1 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.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.9.2.2 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.

6.9.2.3 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.
---------------	--

6.9.2.4 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.

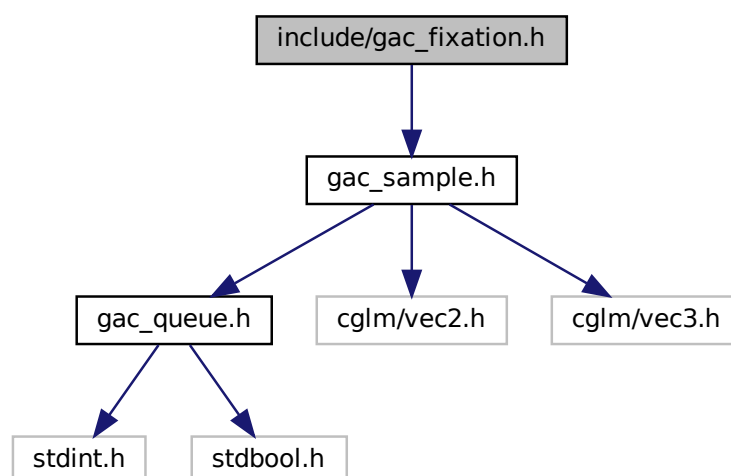
Returns

True on success, false on failure.

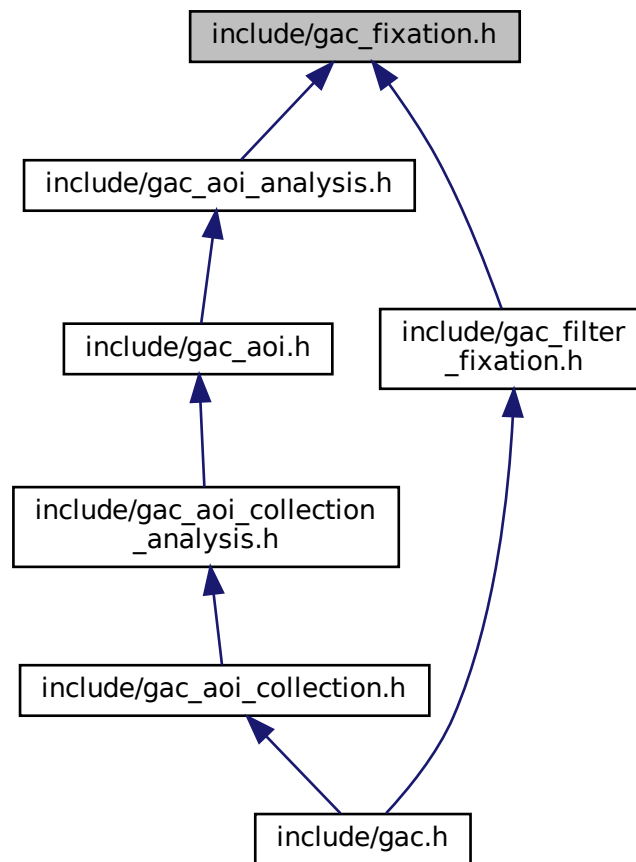
6.10 include/gac_fixation.h File Reference

```
#include "gac_sample.h"
```

Include dependency graph for `gac_fixation.h`:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_fixation_t](#)

Functions

- [gac_fixation_t * gac_fixation_copy](#) ([gac_fixation_t *fixation](#))
- [bool gac_fixation_copy_to](#) ([gac_fixation_t *tgt](#), [gac_fixation_t *src](#))
- [gac_fixation_t * gac_fixation_create](#) ([vec2 *screen_point](#), [vec3 *point](#), [double duration](#), [gac_sample_t *first_sample](#))
- [void gac_fixation_destroy](#) ([gac_fixation_t *fixation](#))
- [bool gac_fixation_init](#) ([gac_fixation_t *fixation](#), [vec2 *screen_point](#), [vec3 *point](#), [double duration](#), [gac_sample_t *first_sample](#))
- [float gac_fixation_normalised_dispersion_threshold](#) ([float angle](#))

6.10.1 Detailed Description

The fixation data structure.

[gac_fixation.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.10.2 Function Documentation**6.10.2.1 gac_fixation_copy()**

```
gac_fixation_t* gac_fixation_copy (
    gac_fixation_t * fixation )
```

Create a new copy of fixation.

Parameters

<i>fixation</i>	A pointer to the fixation to copy.
-----------------	------------------------------------

Returns

A pointer to a newly allocated fixation.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.10.2.2 gac_fixation_copy_to()

```
bool gac_fixation_copy_to (
    gac_fixation_t * tgt,
    gac_fixation_t * src )
```

Copy a fixation structure.

Parameters

<i>tgt</i>	A pointer to a fixation structure to cpy to.
<i>src</i>	A pointer to the fixation structure to be copied.

6.10.2.3 gac_fixation_create()

```
gac_fixation_t* gac_fixation_create (
    vec2 * screen_point,
    vec3 * point,
    double duration,
    gac_sample_t * first_sample )
```

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

Parameters

<i>screen_point</i>	The fixation screen point.
<i>point</i>	The fixation point.
<i>duration</i>	The duration of the fixation.
<i>first_sample</i>	The first sample in the fixation.

Returns

The allocated fixation structure or NULL on failure.

6.10.2.4 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.
-----------------	---

6.10.2.5 gac_fixation_init()

```
bool gac_fixation_init (
    gac_fixation_t * fixation,
    vec2 * screen_point,
    vec3 * point,
    double duration,
    gac_sample_t * first_sample )
```

Initialise a fixation structure.

Parameters

<i>fixation</i>	The fixation structure to initialise.
<i>screen_point</i>	The fixation screen point.
<i>point</i>	The fixation point.
<i>duration</i>	The duration of the fixation.
<i>first_sample</i>	The first sample in the fixation.

Returns

True on success, false on failure.

6.10.2.6 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

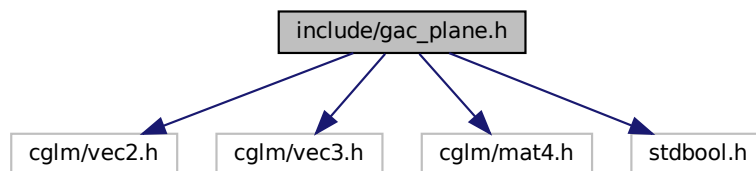
<i>angle</i>	The angel in degrees for which the dispersion threshold is computetd. Usual values range from 0.5 to 1 degree.
--------------	--

Returns

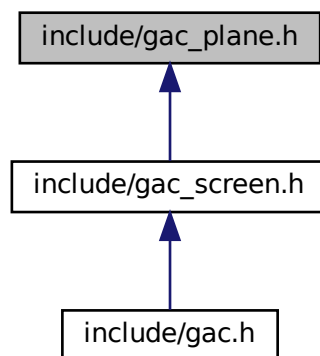
The normalized dispersion threshold.

6.11 include/gac_plane.h File Reference

```
#include <cglm/vec2.h>
#include <cglm/vec3.h>
#include <cglm/mat4.h>
#include <stdbool.h>
Include dependency graph for gac_plane.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_plane_t](#)

Functions

- [gac_plane_t](#) * [gac_plane_create](#) (vec3 *p1, vec3 *p2, vec3 *p3)

- void [gac_plane_destroy](#) (gac_plane_t *plane)
- bool [gac_plane_init](#) (gac_plane_t *plane, vec3 *p1, vec3 *p2, vec3 *p3)
- bool [gac_plane_intersection](#) (gac_plane_t *plane, vec3 *origin, vec3 *dir, vec3 *intersection)
- bool [gac_plane_point](#) (gac_plane_t *plane, vec3 *point3d, vec2 *point2d)

6.11.1 Detailed Description

Plane definitions to work with 3d to 2d conversions. A plane is defined through three arbitrary points.
[gac_plane.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.11.2 Function Documentation

6.11.2.1 gac_plane_create()

```
gac_plane_t* gac_plane_create (  
    vec3 * p1,  
    vec3 * p2,  
    vec3 * p3 )
```

Allocate a plane in 3d space. This need to be freed with [gac_plane_destroy\(\)](#).

Parameters

<i>p1</i>	The 3d coordinates of a point in 3d space.
<i>p2</i>	The 3d coordinates of a point in 3d space.
<i>p3</i>	The 3d coordinates of a point in 3d space.

Returns

A pointer to the allocated plane or NULL on failure.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.11.2.2 gac_plane_destroy()

```
void gac_plane_destroy (  
    gac_plane_t * plane )
```

Destroy a plane structure.

Parameters

<i>plane</i>	A pointer to the plane structure to destroy.
--------------	--

6.11.2.3 gac_plane_init()

```
bool gac_plane_init (
    gac_plane_t * plane,
    vec3 * p1,
    vec3 * p2,
    vec3 * p3 )
```

Initialise a plane in 3d space.

Parameters

<i>plane</i>	A pointer to the plane structure to initialise.
<i>p1</i>	The 3d coordinates of a point in 3d space.
<i>p2</i>	The 3d coordinates of a point in 3d space.
<i>p3</i>	The 3d coordinates of a point in 3d space.

Returns

True on succes and false on failure.

6.11.2.4 gac_plane_intersection()

```
bool gac_plane_intersection (
    gac_plane_t * plane,
    vec3 * origin,
    vec3 * dir,
    vec3 * intersection )
```

Compute the 3d intersection point with a plane.

Parameters

<i>plane</i>	A pointer to the plane structure.
<i>origin</i>	The origin of the gaze.
<i>dir</i>	The gaze direction.
<i>intersection</i>	A location to store the intersection point. This is only valid if the function returns true.

Returns

True if an intersection was found, false otherwise.

6.11.2.5 gac_plane_point()

```
bool gac_plane_point (
    gac_plane_t * plane,
    vec3 * point3d,
    vec2 * point2d )
```

Transform a 3d gaze point into a 2d point on a plane. This only works for 3d points which coincide with the plane. To compute an intersection use the function [gac_plane_intersection\(\)](#).

Parameters

<i>plane</i>	A pointer to the plane structure.
<i>point3d</i>	The 3d point to transform.
<i>point2d</i>	A location where the 2d point will be stored. This is only valid if the function returns true.

Returns

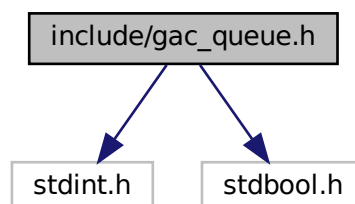
True on success, false otherwise.

6.12 include/gac_queue.h File Reference

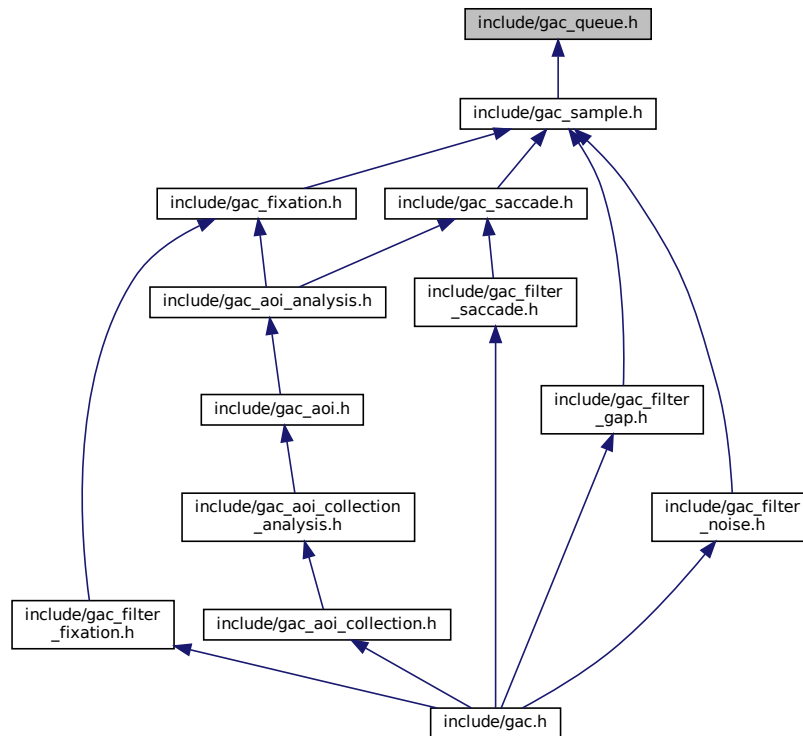
```
#include <stdint.h>
```

```
#include <stdbool.h>
```

Include dependency graph for gac_queue.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_queue_item_t](#)
- struct [gac_queue_t](#)

Functions

- 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 *))

6.12.1 Detailed Description

A queue structure which grows dynamically with added items.

[gac_queue.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.12.2 Function Documentation

6.12.2.1 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.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.12.2.2 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.

6.12.2.3 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
--------------	-----------------------------------

6.12.2.4 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.

6.12.2.5 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.

6.12.2.6 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.

6.12.2.7 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.

6.12.2.8 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.

6.12.2.9 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.

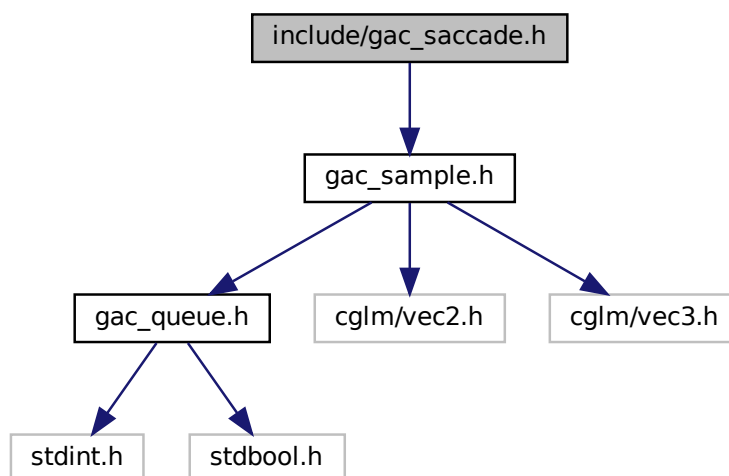
Returns

True on success, false on failure.

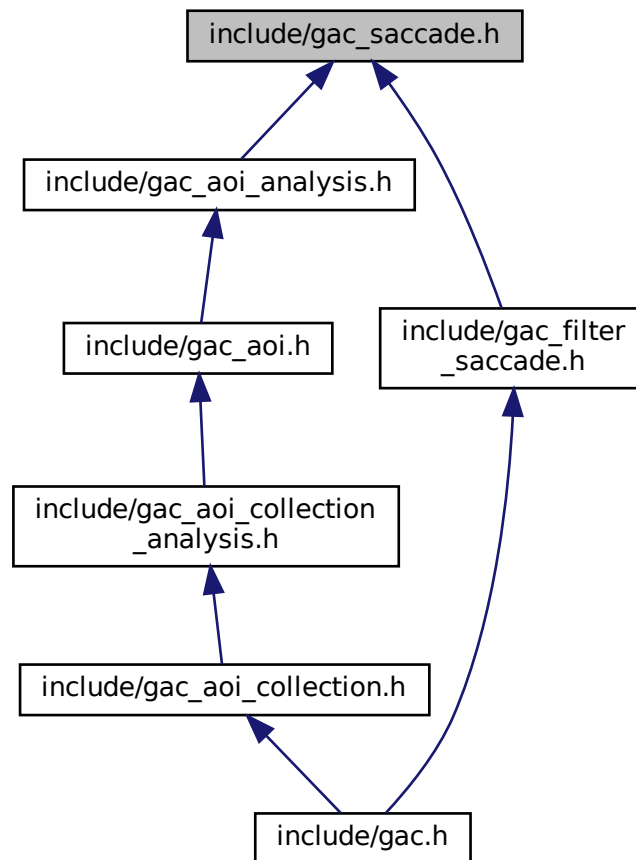
6.13 include/gac_saccade.h File Reference

```
#include "gac_sample.h"
```

Include dependency graph for gac_saccade.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_saccade_t](#)

Functions

- [gac_saccade_t](#) * [gac_saccade_copy](#) ([gac_saccade_t](#) *saccade)
- bool [gac_saccade_copy_to](#) ([gac_saccade_t](#) *tgt, [gac_saccade_t](#) *src)
- [gac_saccade_t](#) * [gac_saccade_create](#) ([gac_sample_t](#) *first_sample, [gac_sample_t](#) *last_sample)
- void [gac_saccade_destroy](#) ([gac_saccade_t](#) *saccade)
- bool [gac_saccade_init](#) ([gac_saccade_t](#) *saccade, [gac_sample_t](#) *first_sample, [gac_sample_t](#) *last_sample)

6.13.1 Detailed Description

The saccade data structure.

[gac_saccade.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.13.2 Function Documentation**6.13.2.1 gac_saccade_copy()**

```
gac_saccade_t* gac_saccade_copy (
    gac_saccade_t * saccade )
```

Create a new copy of saccade.

Parameters

<i>saccade</i>	A pointer to the saccade to copy.
----------------	-----------------------------------

Returns

A pointer to a newly allocated saccade.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.13.2.2 gac_saccade_copy_to()

```
bool gac_saccade_copy_to (
    gac_saccade_t * tgt,
    gac_saccade_t * src )
```

Copy a saccade structure.

Parameters

<i>tgt</i>	A pointer to a saccade structure to copy to.
<i>src</i>	A pointer to the saccade structure to be copied.

6.13.2.3 gac_saccade_create()

```
gac_saccade_t* gac_saccade_create (
    gac_sample_t * first_sample,
    gac_sample_t * last_sample )
```

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

Parameters

<i>first_sample</i>	The first sample of the saccade, holding the source point.
<i>last_sample</i>	The last sample of the saccade, holding the target point.

Returns

The allocated saccade structure on success or NULL on failure.

6.13.2.4 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.
----------------	--

6.13.2.5 gac_saccade_init()

```
bool gac_saccade_init (
    gac_saccade_t * saccade,
    gac_sample_t * first_sample,
    gac_sample_t * last_sample )
```

Initialise a saccade structure.

Parameters

<i>saccade</i>	A pointer to the saccade structure to initialise.
<i>first_sample</i>	The first sample of the saccade, holding the source point.
<i>last_sample</i>	The last sample of the saccade, holding the target point.

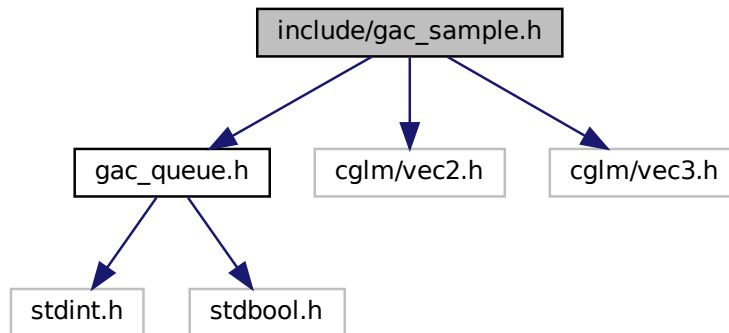
Returns

True on success, false on failure.

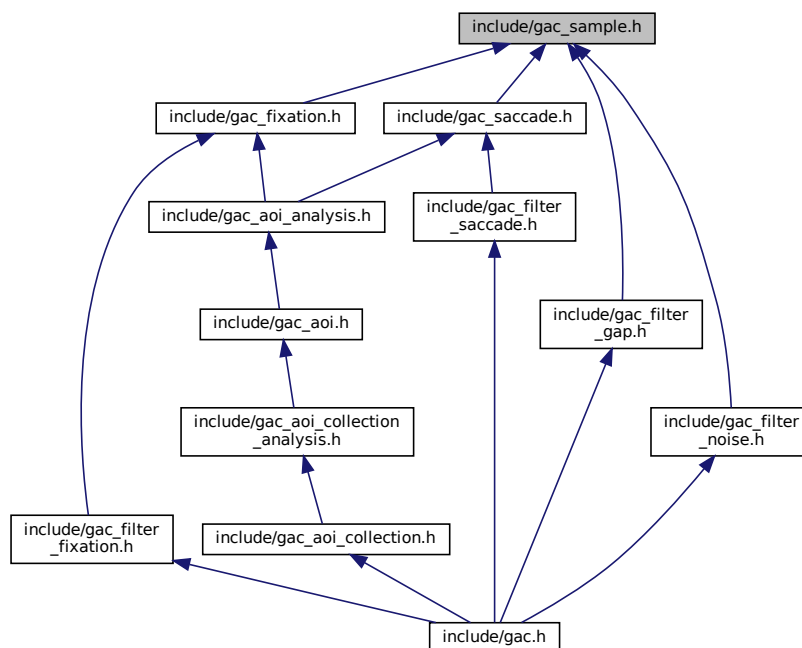
6.14 include/gac_sample.h File Reference

```
#include "gac_queue.h"
#include <cglm/vec2.h>
#include <cglm/vec3.h>
```

Include dependency graph for `gac_sample.h`:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `gac_sample_t`

Macros

- `#define GAC_SAMPLE_MAX_LABEL_LEN 100`

Functions

- `gac_sample_t * gac_sample_create` (`vec2 *screen_point`, `vec3 *origin`, `vec3 *point`, `double timestamp`, `uint32_t trial_id`, `const char *label`)
- `gac_sample_t * gac_sample_copy` (`gac_sample_t *sample`)
- `bool gac_sample_copy_to` (`gac_sample_t *dest`, `gac_sample_t *sample`)
- `void gac_sample_destroy` (`void *sample`)
- `double gac_sample_get_label_timestamp` (`gac_sample_t *sample`)
- `double gac_sample_get_onset` (`gac_sample_t *sample`, `double ref`)
- `double gac_sample_get_trial_timestamp` (`gac_sample_t *sample`)
- `bool gac_sample_init` (`gac_sample_t *sample`, `vec2 *screen_point`, `vec3 *origin`, `vec3 *point`, `double timestamp`, `uint32_t trial_id`, `const char *label`)
- `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_average_screen_point` (`gac_queue_t *samples`, `vec2 *avg`, `uint32_t count`)
- `bool gac_samples_dispersion` (`gac_queue_t *samples`, `float *dispersion`, `uint32_t count`)

6.14.1 Detailed Description

Gaze analysis sample definitions and helper functions.

[gac_sample.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.14.2 Macro Definition Documentation

6.14.2.1 GAC_SAMPLE_MAX_LABEL_LEN

```
#define GAC_SAMPLE_MAX_LABEL_LEN 100
```

The maximal label length

6.14.3 Function Documentation

6.14.3.1 gac_sample_copy()

```
gac_sample_t* gac_sample_copy (
    gac_sample_t * sample )
```

Create a deep copy of a sample. This needs to be freed with [gac_sample_destroy\(\)](#).

Parameters

<i>sample</i>	The sample to copy
---------------	--------------------

Returns

A pointer to the new sample or NULL.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.14.3.2 gac_sample_copy_to()

```
bool gac_sample_copy_to (
    gac_sample_t * dest,
    gac_sample_t * sample )
```

Deep copy of a sample to a target. This needs to be freed with [gac_sample_destroy\(\)](#).

Parameters

<i>dest</i>	The location where the sample will be copied to.
<i>sample</i>	The sample to copy

Returns

A pointer to the new sample or NULL.

6.14.3.3 gac_sample_create()

```
gac_sample_t* gac_sample_create (
    vec2 * screen_point,
    vec3 * origin,
    vec3 * point,
    double timestamp,
    uint32_t trial_id,
    const char * label )
```

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

Parameters

<i>screen_point</i>	The 2d screen gaze point vector.
<i>origin</i>	The gaze origin vector.
<i>point</i>	The gaze point vector.
<i>timestamp</i>	The timestamp of the sample.
<i>trial_id</i>	The ID of the ongoing trial.
<i>label</i>	An optional arbitrary label annotating the sample.

Returns

The allocated sample structure or NULL on failure.

6.14.3.4 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.
---------------	---

6.14.3.5 gac_sample_get_label_timestamp()

```
double gac_sample_get_label_timestamp (
    gac_sample_t * sample )
```

Compute the label timestamp of a sample.

Parameters

<i>sample</i>	A pointer to a sample.
---------------	------------------------

Returns

The label timestamp in milliseconds.

6.14.3.6 gac_sample_get_onset()

```
double gac_sample_get_onset (
    gac_sample_t * sample,
    double ref )
```

Compute the sample onset in milliseconds given a reference timestamp.

Parameters

<i>sample</i>	A pointer to a sample.
<i>ref</i>	A reference timestamp.

Returns

The sample onset in milliseconds.

6.14.3.7 gac_sample_get_trial_timestamp()

```
double gac_sample_get_trial_timestamp (
    gac_sample_t * sample )
```

Compute the trial timestamp of a sample.

Parameters

<i>sample</i>	A pointer to a sample.
---------------	------------------------

Returns

The trial timestamp in milliseconds.

6.14.3.8 gac_sample_init()

```
bool gac_sample_init (
    gac_sample_t * sample,
    vec2 * screen_point,
    vec3 * origin,
    vec3 * point,
    double timestamp,
    uint32_t trial_id,
    const char * label )
```

Initialise a sample structure.

Parameters

<i>sample</i>	The sample structure to initialise.
<i>screen_point</i>	The 2d screen gaze point vector.
<i>origin</i>	The gaze origin vector.
<i>point</i>	The gaze point vector.
<i>timestamp</i>	The timestamp of the sample.
<i>trial_id</i>	The ID of the ongoing trial.
<i>label</i>	An optional arbitrary label annotating the sample.

Returns

True on success, false on failure.

6.14.3.9 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.

6.14.3.10 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.

Parameters

<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.
--------------	---

6.14.3.11 gac_samples_average_screen_point()

```
bool gac_samples_average_screen_point (
    gac_queue_t * samples,
    vec2 * avg,
    uint32_t count )
```

Compute the average screen 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.

6.14.3.12 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.

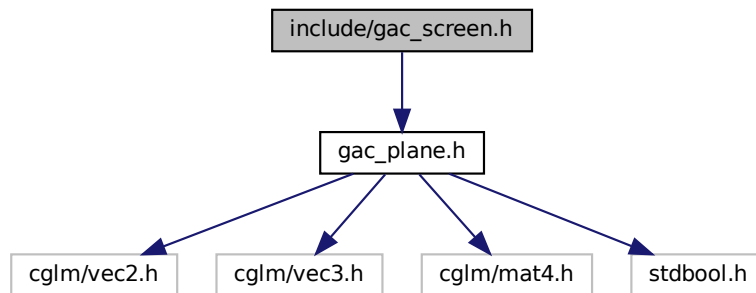
Parameters

<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.

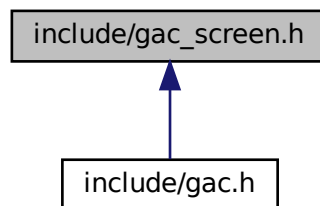
6.15 include/gac_screen.h File Reference

```
#include "gac_plane.h"
```

Include dependency graph for `gac_screen.h`:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gac_screen_t](#)

Functions

- `gac_screen_t * gac_screen_create (vec3 *top_left, vec3 *top_right, vec3 *bottom_left)`
- `void gac_screen_destroy (gac_screen_t *screen)`
- `bool gac_screen_init (gac_screen_t *screen, vec3 *top_left, vec3 *top_right, vec3 *bottom_left)`
- `bool gac_screen_point (gac_screen_t *screen, vec3 *point3d, vec2 *point2d)`
- `bool gac_screen_point_res (gac_screen_t *screen, vec3 *point3d, vec2 *point2d)`
- `bool gac_screen_set_resolution (gac_screen_t *screen, float resolution_x, float resolution_y)`

6.15.1 Detailed Description

Screen definitions to work with 3d to 2d conversions. A screen is defined through three points building the top left, top right and bottom left points of a rectangle. The width of a screen is defined by the length of the vector top left -> top right and the height of the screen is defined by the length of the vector top left -> bottom left.

[gac_screen.h](#)

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.15.2 Function Documentation

6.15.2.1 gac_screen_create()

```
gac_screen_t* gac_screen_create (
    vec3 * top_left,
    vec3 * top_right,
    vec3 * bottom_left )
```

Allocate the screen structure. This needs to be freed with `gac_screen_destroy()`. The screen is defined through the top left, the top right and the bottom left point of the screen in 3d space. The width, the height, and the bottom right point of the screen are computed based on these three points. Make sure to provide points that describe a rectangle for this to make sense.

Parameters

<i>top_left</i>	The 3d coordinates of the top left screen point.
<i>top_right</i>	The 3d coordinates of the top right screen point.
<i>bottom_left</i>	The 3d coordinates of the bottom left screen point.

Returns

A pointer to the allocated screen structure or NULL on failure.

Author

Simon Maurer

License:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

6.15.2.2 gac_screen_destroy()

```
void gac_screen_destroy (
    gac_screen_t * screen )
```

Destroy a screen structure.

Parameters

<i>screen</i>	A pointer to the screen structure to destroy
---------------	--

6.15.2.3 gac_screen_init()

```
bool gac_screen_init (
    gac_screen_t * screen,
    vec3 * top_left,
    vec3 * top_right,
```

```
vec3 * bottom_left )
```

Initialise a screen structure through the top left, the top right and the bottom left point of the screen in 3d space. The width, the height, and the bottom right point of the screen are computed based on these three points. Make sure to provide points that describe a rectangle for this to make sense.

Parameters

<i>screen</i>	A pointer to the screen structure to initialise.
<i>top_left</i>	The 3d coordinates of the top left screen point.
<i>top_right</i>	The 3d coordinates of the top right screen point.
<i>bottom_left</i>	The 3d coordinates of the bottom left screen point.

Returns

True on succes and false on failure.

6.15.2.4 gac_screen_point()

```
bool gac_screen_point (
    gac_screen_t * screen,
    vec3 * point3d,
    vec2 * point2d )
```

Transform a 3d gaze point into a normalized 2d point on the screen. (0, 0) represents the top left corner of the screen and (1, 1) represents the bottom right corner.

Parameters

<i>screen</i>	A pointer to the screen structure.
<i>point3d</i>	The 3d point to transform.
<i>point2d</i>	A location where the 2d point will be stored. This is only valid if the function returns true.

Returns

True on success, false otherwise.

6.15.2.5 gac_screen_point_res()

```
bool gac_screen_point_res (
    gac_screen_t * screen,
    vec3 * point3d,
    vec2 * point2d )
```

The same as [gac_screen_point\(\)](#) but storing the resulting 2d point in terms of screen resolution with (0, 0) being the top left corner of the screen. Note that this function will always return false if [gac_screen_set_resolution\(\)](#) was never called.

6.15.2.6 gac_screen_set_resolution()

```
bool gac_screen_set_resolution (
    gac_screen_t * screen,
    float resolution_x,
    float resolution_y )
```

Set the screen resolution. This allows to use all functions with an `res` suffix. These functions will act exactly like their counter part function without the `res` suffix but use 2d points expressed in the screen resolution.

Parameters

<i>screen</i>	A pointer to a screen structure.
<i>resolution</i> ↔ _x	The width of the screen resolution.
<i>resolution</i> ↔ _y	The height of the screen resolution.

Returns

True on success, false on failure.

Index

`_me`

- `gac_aoi_analysis_t`, [11](#)
- `gac_aoi_collection_analysis_result_t`, [12](#)
- `gac_aoi_collection_analysis_t`, [13](#)
- `gac_aoi_collection_t`, [14](#)
- `gac_aoi_t`, [15](#)
- `gac_filter_fixation_t`, [17](#)
- `gac_filter_gap_t`, [18](#)
- `gac_filter_noise_t`, [19](#)
- `gac_filter_parameter_t`, [20](#)
- `gac_filter_saccade_t`, [21](#)
- `gac_fixation_t`, [22](#)
- `gac_plane_t`, [23](#)
- `gac_queue_t`, [25](#)
- `gac_saccade_t`, [25](#)
- `gac_sample_t`, [26](#)
- `gac_screen_t`, [27](#)
- `gac_t`, [29](#)

`analysis`

- `gac_aoi_collection_t`, [14](#)
- `gac_aoi_t`, [15](#)

`aoi_visited_before_count`

- `gac_aoi_analysis_t`, [11](#)

`aoi_visited_count`

- `gac_aoi_collection_analysis_t`, [13](#)

`aoic`

- `gac_t`, [29](#)

`aois`

- `gac_aoi_collection_analysis_result_t`, [13](#)
- `gac_aoi_collection_t`, [14](#)

`avg_edge_len`

- `gac_aoi_t`, [15](#)

`bounding_box`

- `gac_aoi_t`, [16](#)

`count`

- `gac_aoi_collection_analysis_result_t`, [13](#)
- `gac_aoi_collection_t`, [14](#)
- `gac_aoi_t`, [16](#)
- `gac_queue_t`, [25](#)

`data`

- `gac_queue_item_t`, [24](#)

`dispersion_threshold`

- `gac_filter_parameter_t`, [20](#)

`duration`

- `gac_filter_fixation_t`, [17](#)
- `gac_fixation_t`, [22](#)

`duration_threshold`

- `gac_filter_fixation_t`, [17](#)
- `gac_filter_parameter_t`, [20](#)

`dwel_time`

- `gac_aoi_analysis_t`, [11](#)
- `gac_aoi_collection_analysis_t`, [13](#)

`dwel_time_relative`

- `gac_aoi_analysis_t`, [11](#)

`e1`

- `gac_plane_t`, [23](#)

`e2`

- `gac_plane_t`, [23](#)

`enter_saccade_count`

- `gac_aoi_analysis_t`, [12](#)

`first_fixation`

- `gac_aoi_analysis_t`, [12](#)

`first_saccade`

- `gac_aoi_analysis_t`, [12](#)

`first_sample`

- `gac_fixation_t`, [22](#)
- `gac_saccade_t`, [26](#)

`fixation`

- `gac_filter_parameter_t`, [20](#)
- `gac_t`, [29](#)

`fixation_count`

- `gac_aoi_analysis_t`, [12](#)
- `gac_aoi_collection_analysis_t`, [14](#)

`fixation_count_relative`

- `gac_aoi_analysis_t`, [12](#)

`gac.h`

- `gac_add_aoi`, [32](#)
- `gac_create`, [32](#)
- `gac_destroy`, [33](#)
- `gac_finalise`, [33](#)
- `gac_get_filter_parameter`, [33](#)
- `gac_get_filter_parameter_default`, [33](#)
- `gac_init`, [34](#)
- `gac_sample_window_cleanup`, [34](#)
- `gac_sample_window_fixation_filter`, [35](#)
- `gac_sample_window_saccade_filter`, [35](#)
- `gac_sample_window_update`, [35](#)
- `gac_sample_window_update_screen`, [36](#)
- `gac_sample_window_update_vec`, [37](#)
- `gac_set_screen`, [37](#)
- `gac_version`, [38](#)

`gac_add_aoi`

- `gac.h`, [32](#)

- gac_aoi.h
 - gac_aoi_add_point, 41
 - gac_aoi_add_point_res, 41
 - gac_aoi_add_rect, 41
 - gac_aoi_add_rect_res, 42
 - gac_aoi_copy, 42
 - gac_aoi_copy_to, 42
 - gac_aoi_create, 42
 - gac_aoi_destroy, 43
 - gac_aoi_includes_point, 43
 - gac_aoi_includes_point_res, 43
 - gac_aoi_init, 43
 - gac_aoi_intersect, 44
 - GAC_AOI_MAX, 40
 - GAC_AOI_MAX_LABEL_LEN, 40
 - GAC_AOI_MAX_POINTS, 40
 - GAC_AOI_ORIENTATION_CLOCKWISE, 41
 - GAC_AOI_ORIENTATION_COLINEAR, 41
 - GAC_AOI_ORIENTATION_COUNTER_CLOCKWISE, 41
 - gac_aoi_orientation_e, 40
 - gac_aoi_orientation_t, 40
 - gac_aoi_orientation_triplet, 44
 - gac_aoi_point_on_segment, 45
 - gac_aoi_set_resolution, 45
- gac_aoi_add_point
 - gac_aoi.h, 41
- gac_aoi_add_point_res
 - gac_aoi.h, 41
- gac_aoi_add_rect
 - gac_aoi.h, 41
- gac_aoi_add_rect_res
 - gac_aoi.h, 42
- gac_aoi_analysis.h
 - gac_aoi_analysis_clear, 48
 - gac_aoi_analysis_copy, 48
 - gac_aoi_analysis_copy_to, 48
 - gac_aoi_analysis_create, 49
 - gac_aoi_analysis_destroy, 49
 - gac_aoi_analysis_init, 49
- gac_aoi_analysis_clear
 - gac_aoi_analysis.h, 48
- gac_aoi_analysis_copy
 - gac_aoi_analysis.h, 48
- gac_aoi_analysis_copy_to
 - gac_aoi_analysis.h, 48
- gac_aoi_analysis_create
 - gac_aoi_analysis.h, 49
- gac_aoi_analysis_destroy
 - gac_aoi_analysis.h, 49
- gac_aoi_analysis_init
 - gac_aoi_analysis.h, 49
- gac_aoi_analysis_t, 11
 - _me, 11
 - aoi_visited_before_count, 11
 - dwelt_time, 11
 - dwelt_time_relative, 11
 - enter_saccade_count, 12
 - first_fixation, 12
 - first_saccade, 12
 - fixation_count, 12
 - fixation_count_relative, 12
- gac_aoi_collection.h
 - gac_aoi_collection_add, 51
 - gac_aoi_collection_analyse_clear, 51
 - gac_aoi_collection_analyse_finalise, 52
 - gac_aoi_collection_analyse_fixation, 52
 - gac_aoi_collection_analyse_saccade, 52
 - gac_aoi_collection_assign, 53
 - gac_aoi_collection_create, 53
 - gac_aoi_collection_destroy, 53
 - gac_aoi_collection_init, 53
- gac_aoi_collection_add
 - gac_aoi_collection.h, 51
- gac_aoi_collection_analyse_clear
 - gac_aoi_collection.h, 51
- gac_aoi_collection_analyse_finalise
 - gac_aoi_collection.h, 52
- gac_aoi_collection_analyse_fixation
 - gac_aoi_collection.h, 52
- gac_aoi_collection_analyse_saccade
 - gac_aoi_collection.h, 52
- gac_aoi_collection_analysis.h
 - gac_aoi_collection_analysis_clear, 55
 - gac_aoi_collection_analysis_create, 56
 - gac_aoi_collection_analysis_destroy, 56
 - gac_aoi_collection_analysis_init, 56
- gac_aoi_collection_analysis_clear
 - gac_aoi_collection_analysis.h, 55
- gac_aoi_collection_analysis_create
 - gac_aoi_collection_analysis.h, 56
- gac_aoi_collection_analysis_destroy
 - gac_aoi_collection_analysis.h, 56
- gac_aoi_collection_analysis_init
 - gac_aoi_collection_analysis.h, 56
- gac_aoi_collection_analysis_result_t, 12
 - _me, 12
 - aois, 13
 - count, 13
 - items, 13
 - trial_id, 13
- gac_aoi_collection_analysis_t, 13
 - _me, 13
 - aoi_visited_count, 13
 - dwelt_time, 13
 - fixation_count, 14
 - trial_id, 14
- gac_aoi_collection_assign
 - gac_aoi_collection.h, 53
- gac_aoi_collection_create
 - gac_aoi_collection.h, 53
- gac_aoi_collection_destroy
 - gac_aoi_collection.h, 53
- gac_aoi_collection_init
 - gac_aoi_collection.h, 53
- gac_aoi_collection_t, 14

- [_me](#), [14](#)
 - [analysis](#), [14](#)
 - [aois](#), [14](#)
 - [count](#), [14](#)
 - [items](#), [15](#)
- [gac_aoi_copy](#)
 - [gac_aoi.h](#), [42](#)
- [gac_aoi_copy_to](#)
 - [gac_aoi.h](#), [42](#)
- [gac_aoi_create](#)
 - [gac_aoi.h](#), [42](#)
- [gac_aoi_destroy](#)
 - [gac_aoi.h](#), [43](#)
- [gac_aoi_includes_point](#)
 - [gac_aoi.h](#), [43](#)
- [gac_aoi_includes_point_res](#)
 - [gac_aoi.h](#), [43](#)
- [gac_aoi_init](#)
 - [gac_aoi.h](#), [43](#)
- [gac_aoi_intersect](#)
 - [gac_aoi.h](#), [44](#)
- [GAC_AOI_MAX](#)
 - [gac_aoi.h](#), [40](#)
- [GAC_AOI_MAX_LABEL_LEN](#)
 - [gac_aoi.h](#), [40](#)
- [GAC_AOI_MAX_POINTS](#)
 - [gac_aoi.h](#), [40](#)
- [GAC_AOI_ORIENTATION_CLOCKWISE](#)
 - [gac_aoi.h](#), [41](#)
- [GAC_AOI_ORIENTATION_COLINEAR](#)
 - [gac_aoi.h](#), [41](#)
- [GAC_AOI_ORIENTATION_COUNTER_CLOCKWISE](#)
 - [gac_aoi.h](#), [41](#)
- [gac_aoi_orientation_e](#)
 - [gac_aoi.h](#), [40](#)
- [gac_aoi_orientation_t](#)
 - [gac_aoi.h](#), [40](#)
- [gac_aoi_orientation_triplet](#)
 - [gac_aoi.h](#), [44](#)
- [gac_aoi_point_on_segment](#)
 - [gac_aoi.h](#), [45](#)
- [gac_aoi_set_resolution](#)
 - [gac_aoi.h](#), [45](#)
- [gac_aoi_t](#), [15](#)
 - [_me](#), [15](#)
 - [analysis](#), [15](#)
 - [avg_edge_len](#), [15](#)
 - [bounding_box](#), [16](#)
 - [count](#), [16](#)
 - [items](#), [16](#)
 - [label](#), [16](#)
 - [points](#), [16](#)
 - [ray_origin](#), [16](#)
 - [resolution_x](#), [16](#)
 - [resolution_y](#), [16](#)
- [gac_create](#)
 - [gac.h](#), [32](#)
- [gac_destroy](#)
 - [gac.h](#), [33](#)
- [gac_filter_fixation](#)
 - [gac_filter_fixation.h](#), [58](#)
- [gac_filter_fixation.h](#)
 - [gac_filter_fixation](#), [58](#)
 - [gac_filter_fixation_create](#), [58](#)
 - [gac_filter_fixation_destroy](#), [59](#)
 - [gac_filter_fixation_init](#), [59](#)
- [gac_filter_fixation_create](#)
 - [gac_filter_fixation.h](#), [58](#)
- [gac_filter_fixation_destroy](#)
 - [gac_filter_fixation.h](#), [59](#)
- [gac_filter_fixation_init](#)
 - [gac_filter_fixation.h](#), [59](#)
- [gac_filter_fixation_t](#), [16](#)
 - [_me](#), [17](#)
 - [duration](#), [17](#)
 - [duration_threshold](#), [17](#)
 - [is_collecting](#), [17](#)
 - [new_samples](#), [17](#)
 - [normalized_dispersion_threshold](#), [17](#)
 - [point](#), [17](#)
 - [screen_point](#), [17](#)
 - [window](#), [17](#)
- [gac_filter_gap](#)
 - [gac_filter_gap.h](#), [61](#)
- [gac_filter_gap.h](#)
 - [gac_filter_gap](#), [61](#)
 - [gac_filter_gap_create](#), [61](#)
 - [gac_filter_gap_destroy](#), [62](#)
 - [gac_filter_gap_init](#), [62](#)
- [gac_filter_gap_create](#)
 - [gac_filter_gap.h](#), [61](#)
- [gac_filter_gap_destroy](#)
 - [gac_filter_gap.h](#), [62](#)
- [gac_filter_gap_init](#)
 - [gac_filter_gap.h](#), [62](#)
- [gac_filter_gap_t](#), [18](#)
 - [_me](#), [18](#)
 - [is_enabled](#), [18](#)
 - [max_gap_length](#), [18](#)
 - [sample_period](#), [18](#)
- [gac_filter_noise](#)
 - [gac_filter_noise.h](#), [64](#)
- [gac_filter_noise.h](#)
 - [gac_filter_noise](#), [64](#)
 - [gac_filter_noise_average](#), [65](#)
 - [gac_filter_noise_create](#), [65](#)
 - [gac_filter_noise_destroy](#), [65](#)
 - [gac_filter_noise_init](#), [66](#)
 - [GAC_FILTER_NOISE_TYPE_AVERAGE](#), [64](#)
 - [gac_filter_noise_type_e](#), [64](#)
 - [GAC_FILTER_NOISE_TYPE_MEDIAN](#), [64](#)
 - [gac_filter_noise_type_t](#), [64](#)
- [gac_filter_noise_average](#)
 - [gac_filter_noise.h](#), [65](#)
- [gac_filter_noise_create](#)
 - [gac_filter_noise.h](#), [65](#)

- gac_filter_noise_destroy
 - gac_filter_noise.h, 65
- gac_filter_noise_init
 - gac_filter_noise.h, 66
- gac_filter_noise_t, 18
 - _me, 19
 - is_enabled, 19
 - mid, 19
 - type, 19
 - window, 19
- GAC_FILTER_NOISE_TYPE_AVERAGE
 - gac_filter_noise.h, 64
- gac_filter_noise_type_e
 - gac_filter_noise.h, 64
- GAC_FILTER_NOISE_TYPE_MEDIAN
 - gac_filter_noise.h, 64
- gac_filter_noise_type_t
 - gac_filter_noise.h, 64
- gac_filter_parameter_t, 19
 - _me, 20
 - dispersion_threshold, 20
 - duration_threshold, 20
 - fixation, 20
 - gap, 20
 - max_gap_length, 20
 - mid_idx, 20
 - noise, 20
 - saccade, 20
 - sample_period, 20
 - type, 20
 - velocity_threshold, 21
- gac_filter_saccade
 - gac_filter_saccade.h, 68
- gac_filter_saccade.h
 - gac_filter_saccade, 68
 - gac_filter_saccade_create, 68
 - gac_filter_saccade_destroy, 69
 - gac_filter_saccade_init, 69
- gac_filter_saccade_create
 - gac_filter_saccade.h, 68
- gac_filter_saccade_destroy
 - gac_filter_saccade.h, 69
- gac_filter_saccade_init
 - gac_filter_saccade.h, 69
- gac_filter_saccade_t, 21
 - _me, 21
 - is_collecting, 21
 - new_samples, 21
 - velocity_threshold, 21
 - window, 21
- gac_finalise
 - gac.h, 33
- gac_fixation.h
 - gac_fixation_copy, 72
 - gac_fixation_copy_to, 72
 - gac_fixation_create, 72
 - gac_fixation_destroy, 73
 - gac_fixation_init, 73
 - gac_fixation_normalised_dispersion_threshold, 73
 - gac_fixation_copy
 - gac_fixation.h, 72
 - gac_fixation_copy_to
 - gac_fixation.h, 72
 - gac_fixation_create
 - gac_fixation.h, 72
 - gac_fixation_destroy
 - gac_fixation.h, 73
 - gac_fixation_init
 - gac_fixation.h, 73
 - gac_fixation_normalised_dispersion_threshold
 - gac_fixation.h, 73
 - gac_fixation_t, 22
 - _me, 22
 - duration, 22
 - first_sample, 22
 - point, 22
 - screen_point, 22
 - gac_get_filter_parameter
 - gac.h, 33
 - gac_get_filter_parameter_default
 - gac.h, 33
 - gac_init
 - gac.h, 34
 - gac_plane.h
 - gac_plane_create, 75
 - gac_plane_destroy, 75
 - gac_plane_init, 76
 - gac_plane_intersection, 76
 - gac_plane_point, 76
 - gac_plane_create
 - gac_plane.h, 75
 - gac_plane_destroy
 - gac_plane.h, 75
 - gac_plane_init
 - gac_plane.h, 76
 - gac_plane_intersection
 - gac_plane.h, 76
 - gac_plane_point
 - gac_plane.h, 76
 - gac_plane_t, 22
 - _me, 23
 - e1, 23
 - e2, 23
 - m, 23
 - norm, 23
 - p1, 23
 - p2, 23
 - p3, 23
 - gac_queue.h
 - gac_queue_clear, 79
 - gac_queue_create, 79
 - gac_queue_destroy, 79
 - gac_queue_grow, 80
 - gac_queue_init, 80
 - gac_queue_pop, 80
 - gac_queue_push, 80

[illegible]

- gac_screen_init, 93
 - gac_screen_point, 94
 - gac_screen_point_res, 94
 - gac_screen_set_resolution, 94
- gac_screen_create
 - gac_screen.h, 93
- gac_screen_destroy
 - gac_screen.h, 93
- gac_screen_init
 - gac_screen.h, 93
- gac_screen_point
 - gac_screen.h, 94
- gac_screen_point_res
 - gac_screen.h, 94
- gac_screen_set_resolution
 - gac_screen.h, 94
- gac_screen_t, 27
 - _me, 27
 - height, 27
 - origin, 28
 - plane, 28
 - resolution_x, 28
 - resolution_y, 28
 - width, 28
- gac_set_screen
 - gac.h, 37
- gac_t, 28
 - _me, 29
 - aoic, 29
 - fixation, 29
 - gap, 29
 - label_timestamp, 29
 - last_sample, 29
 - noise, 29
 - parameter, 29
 - saccade, 29
 - samples, 29
 - screen, 29
 - trial_timestamp, 29
- gac_version
 - gac.h, 38
- gap
 - gac_filter_parameter_t, 20
 - gac_t, 29
- head
 - gac_queue_t, 25
- height
 - gac_screen_t, 27
- include/gac.h, 31
- include/gac_aoi.h, 38
- include/gac_aoi_analysis.h, 46
- include/gac_aoi_collection.h, 49
- include/gac_aoi_collection_analysis.h, 54
- include/gac_filter_fixation.h, 56
- include/gac_filter_gap.h, 59
- include/gac_filter_noise.h, 62
- include/gac_filter_saccade.h, 66
- include/gac_fixation.h, 69
- include/gac_plane.h, 74
- include/gac_queue.h, 77
- include/gac_saccade.h, 82
- include/gac_sample.h, 85
- include/gac_screen.h, 91
- is_collecting
 - gac_filter_fixation_t, 17
 - gac_filter_saccade_t, 21
- is_enabled
 - gac_filter_gap_t, 18
 - gac_filter_noise_t, 19
- items
 - gac_aoi_collection_analysis_result_t, 13
 - gac_aoi_collection_t, 15
 - gac_aoi_t, 16
- label
 - gac_aoi_t, 16
 - gac_sample_t, 26
- label_onset
 - gac_sample_t, 26
- label_timestamp
 - gac_t, 29
- last_sample
 - gac_saccade_t, 26
 - gac_t, 29
- length
 - gac_queue_t, 25
- m
 - gac_plane_t, 23
- max_gap_length
 - gac_filter_gap_t, 18
 - gac_filter_parameter_t, 20
- mid
 - gac_filter_noise_t, 19
- mid_idx
 - gac_filter_parameter_t, 20
- new_samples
 - gac_filter_fixation_t, 17
 - gac_filter_saccade_t, 21
- next
 - gac_queue_item_t, 24
- noise
 - gac_filter_parameter_t, 20
 - gac_t, 29
- norm
 - gac_plane_t, 23
- normalized_dispersion_threshold
 - gac_filter_fixation_t, 17
- origin
 - gac_sample_t, 26
 - gac_screen_t, 28
- p1
 - gac_plane_t, 23

p2
 gac_plane_t, 23
p3
 gac_plane_t, 23
parameter
 gac_t, 29
plane
 gac_screen_t, 28
point
 gac_filter_fixation_t, 17
 gac_fixation_t, 22
 gac_sample_t, 27
points
 gac_aoi_t, 16
prev
 gac_queue_item_t, 24

ray_origin
 gac_aoi_t, 16
resolution_x
 gac_aoi_t, 16
 gac_screen_t, 28
resolution_y
 gac_aoi_t, 16
 gac_screen_t, 28
rm
 gac_queue_t, 25

saccade
 gac_filter_parameter_t, 20
 gac_t, 29
sample_period
 gac_filter_gap_t, 18
 gac_filter_parameter_t, 20
samples
 gac_t, 29
screen
 gac_t, 29
screen_point
 gac_filter_fixation_t, 17
 gac_fixation_t, 22
 gac_sample_t, 27

tail
 gac_queue_t, 25
timestamp
 gac_sample_t, 27
trial_id
 gac_aoi_collection_analysis_result_t, 13
 gac_aoi_collection_analysis_t, 14
 gac_sample_t, 27
trial_onset
 gac_sample_t, 27
trial_timestamp
 gac_t, 29
type
 gac_filter_noise_t, 19
 gac_filter_parameter_t, 20

velocity_threshold
 gac_filter_parameter_t, 21
 gac_filter_saccade_t, 21

width
 gac_screen_t, 28
window
 gac_filter_fixation_t, 17
 gac_filter_noise_t, 19
 gac_filter_saccade_t, 21