

Geli Vision

Generated by Doxygen 1.9.6

1 Geli Vision	1
2	3
2.1 Vue	3
2.2 Element-Plus	3
2.3 LogicFlow	3
2.4 Spring Boot	3
2.5 OpenCV	4
2.6 Socket.IO	4
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Class Documentation	11
6.1 Object< T > Struct Template Reference	11
6.1.1 Detailed Description	11
6.1.2 Constructor & Destructor Documentation	12
6.1.2.1 Object()	12
6.1.3 Member Function Documentation	12
6.1.3.1 inner()	12
6.1.3.2 inner_const_ref()	12
6.1.3.3 inner_ref()	13
6.2 ObjectBase Struct Reference	13
6.2.1 Detailed Description	13
7 File Documentation	15
7.1 general.cpp File Reference	15
7.1.1 Detailed Description	16
7.1.2 Function Documentation	16
7.1.2.1 convert_color()	16
7.1.2.2 draw_contours()	16
7.1.2.3 find_contours()	16
7.1.2.4 median_blur()	17
7.1.2.5 morph_close()	17
7.1.2.6 morph_open()	18
7.1.2.7 read()	18
7.1.2.8 show()	18
7.1.2.9 threshold()	19

7.2 object.h File Reference	19
7.2.1 Detailed Description	20
7.2.2 Function Documentation	20
7.2.2.1 get_inner()	20
7.2.2.2 get_inner_const_ref()	20
7.2.2.3 get_inner_ref()	21
7.2.2.4 make_param()	21
7.3 object.h	21
Index	23

Chapter 1

Geli Vision

An image processing software.

Chapter 2

2.1 Vue

JavaScript Web

<https://vuejs.org>

Vue Mastery: <https://www.vuemastery.com>

Vuetify: <https://vuetifyjs.com>

Vue.js Developers: <https://vuejsdevelopers.com>

2.2 Element-Plus

Vue 3 UI

<https://element-plus.org>

2.3 LogicFlow

LogicFlow

<https://docs.logic-flow.cn/docs/#/zh/guide/start>

2.4 Spring Boot

Spring

<https://spring.io/projects/spring-boot>

Baeldung: <https://www.baeldung.com/spring-boot>

Spring Boot Tutorial: <https://www.javatpoint.com/spring-boot-tutorial>

Spring Framework Guru: <https://springframework.guru>

2.5 OpenCV

<https://opencv.org>

OpenCV C++ Tutorials: https://docs.opencv.org/master/d9/df8/tutorial_root.html

OpenCV-Python Tutorials: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

Learn OpenCV: <https://www.learnopencv.com>

OpenCV Cookbook: <https://www.packtpub.com/product/opencv-cookbook/9781789344912>

2.6 Socket.IO

<https://socket.io/docs/v4>

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ObjectBase	13
Object< T >	11

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Object< T >	A template class for wrapping objects of type T	11
ObjectBase	Base class for all objects	13

Chapter 5

File Index

5.1 File List

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

general.cpp	Contains functions for image processing	15
object.h	Contains container for various types of data	19

Chapter 6

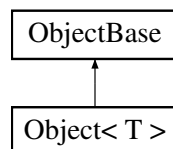
Class Documentation

6.1 Object< T > Struct Template Reference

A template class for wrapping objects of type T.

```
#include <object.h>
```

Inheritance diagram for Object< T >:



Public Member Functions

- `Object (T *ptr)`
Constructs an `Object` from a pointer to an object of type T.
- `~Object ()`
The destructor.
- `T inner ()`
Gets the inner object.
- `T & inner_ref ()`
Gets a reference to the inner object.
- `const T & inner_const_ref ()`
Gets a const reference to the inner object.

6.1.1 Detailed Description

```
template<typename T>  
struct Object< T >
```

A template class for wrapping objects of type T.

Template Parameters

<i>T</i>	The type of object to wrap.
----------	-----------------------------

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Object()

```
template<typename T >
Object< T >::Object (
    T * ptr )
```

Constructs an [Object](#) from a pointer to an object of type T.

Parameters

<i>ptr</i>	A pointer to the object to wrap.
------------	----------------------------------

6.1.3 Member Function Documentation

6.1.3.1 inner()

```
template<typename T >
T Object< T >::inner
```

Gets the inner object.

Returns

The inner object.

6.1.3.2 inner_const_ref()

```
template<typename T >
const T & Object< T >::inner_const_ref
```

Gets a const reference to the inner object.

Returns

A const reference to the inner object.

6.1.3.3 inner_ref()

```
template<typename T >
T & Object< T >::inner_ref
```

Gets a reference to the inner object.

Returns

A reference to the inner object.

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

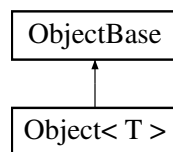
- [object.h](#)

6.2 ObjectBase Struct Reference

base class for all objects

```
#include <object.h>
```

Inheritance diagram for ObjectBase:



6.2.1 Detailed Description

base class for all objects

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

- [object.h](#)

Chapter 7

File Documentation

7.1 general.cpp File Reference

Contains functions for image processing.

```
#include "object.h"
```

Macros

- `#define __EXPORT extern "C" __declspec(dllexport)`

Functions

- `__EXPORT void read (ParamPtrArray ¶ms)`
Reads an image from file.
- `__EXPORT void show (ParamPtrArray ¶ms)`
Shows an image.
- `__EXPORT void morph_open (ParamPtrArray ¶ms)`
Applies morphological opening to an image.
- `__EXPORT void morph_close (ParamPtrArray ¶ms)`
Applies morphological closing to an image.
- `__EXPORT void threshold (ParamPtrArray ¶ms)`
Applies morphological closing to an image.
- `__EXPORT void convert_color (ParamPtrArray ¶ms)`
Convert an input image to a different color space.
- `__EXPORT void median_blur (ParamPtrArray ¶ms)`
Apply median blur to an input image.
- `__EXPORT void find_contours (ParamPtrArray ¶ms)`
Find contours in a binary image.
- `__EXPORT void draw_contours (ParamPtrArray ¶ms)`
Draw contours in a binary image.

7.1.1 Detailed Description

Contains functions for image processing.

7.1.2 Function Documentation

7.1.2.1 `convert_color()`

```
__EXPORT void convert_color (
    ParamPtrArray & params )
```

Convert an input image to a different color space.

Parameters

<i>params</i>	An array of parameters containing: <ol style="list-style-type: none"> 1. input A <code>cv::Mat</code> object representing the image. 2. input A <code>int</code> representing the conversion type. 3. output A <code>cv::Mat</code> object representing the processed image.
---------------	--

7.1.2.2 `draw_contours()`

```
__EXPORT void draw_contours (
    ParamPtrArray & params )
```

Draw contours in a binary image.

Parameters

<i>params</i>	An array of parameters containing: <ol style="list-style-type: none"> 1. input A <code>cv::Mat</code> object representing the input binary image. 2. input A <code>std::vector<std::vector<cv::Point>></code> object representing the contours. 3. output A <code>cv::Mat</code> object representing the image with drawn countours.
---------------	--

7.1.2.3 `find_contours()`

```
__EXPORT void find_contours (
```

```
ParamPtrArray & params )
```

Find contours in a binary image.

Parameters

<i>params</i>	<p>An array of parameters containing:</p> <ol style="list-style-type: none"> 1. input A <code>cv::Mat</code> object representing the input binary image. 2. output A <code>std::vector<std::vector<cv::Point>></code> object representing the found contours.
---------------	---

7.1.2.4 median_blur()

```
__EXPORT void median_blur (
    ParamPtrArray & params )
```

Apply median blur to an input image.

Parameters

<i>params</i>	<p>An array of parameters containing:</p> <ol style="list-style-type: none"> 1. input A <code>cv::Mat</code> object representing the image. 2. input A <code>int</code> representing the kernel size. 3. output A <code>cv::Mat</code> object representing the processed image.
---------------	---

7.1.2.5 morph_close()

```
__EXPORT void morph_close (
    ParamPtrArray & params )
```

Applies morphological closing to an image.

Parameters

<i>params</i>	<p>An array of parameters containing:</p> <ol style="list-style-type: none"> 1. input A <code>cv::Mat</code> object representing the image. 2. input A <code>cv::Size</code> object representing the kernel size for the operation. 3. output A <code>cv::Mat</code> object representing the processed image.
---------------	---

7.1.2.6 morph_open()

```
__EXPORT void morph_open (
    ParamPtrArray & params )
```

Applies morphological opening to an image.

Parameters

<i>params</i>	<p>An array of parameters containing:</p> <ol style="list-style-type: none"> 1. input A <code>cv::Mat</code> object representing the image. 2. input A <code>cv::Size</code> object representing the kernel size for the operation. 3. output A <code>cv::Mat</code> object representing the processed image.
---------------	---

7.1.2.7 read()

```
__EXPORT void read (
    ParamPtrArray & params )
```

Reads an image from file.

Parameters

<i>params</i>	<p>An array of parameters containing:</p> <ol style="list-style-type: none"> 1. input A <code>string</code> representing the file path of the image. 2. input An <code>integer</code> representing the desired color type of the image. 3. output A <code>cv::Mat</code> object representing the read image.
---------------	--

7.1.2.8 show()

```
__EXPORT void show (
    ParamPtrArray & params )
```

Shows an image.

Parameters

<i>params</i>	<p>An array of parameters containing:</p> <ol style="list-style-type: none"> 1. input A <code>cv::Mat</code> object representing the image. 2. output A <code>cv::Mat</code> object representing the image.
---------------	---

7.1.2.9 threshold()

```
__EXPORT void threshold (
    ParamPtrArray & params )
```

Applies morphological closing to an image.

Parameters

<i>params</i>	<p>An array of parameters containing:</p> <ol style="list-style-type: none"> 1. input A <code>cv::Mat</code> object representing the image. 2. input A <code>double</code> representing the threshold. 3. input A <code>double</code> representing the max value. 4. input A <code>int</code> representing the threshold type. 5. output A <code>cv::Mat</code> object representing the processed image.
---------------	--

7.2 object.h File Reference

Contains container for various types of data.

```
#include <string>
#include <vector>
#include <opencv2/opencv.hpp>
```

Classes

- struct [ObjectBase](#)
base class for all objects
- struct [Object< T >](#)
A template class for wrapping objects of type T.

Typedefs

- using **ParamPtr** = `std::shared_ptr< ObjectBase >`
- using **ParamPtrArray** = `std::vector< ParamPtr >`
- using **MatObject** = [Object](#)< `cv::Mat` >
- using **SizeObject** = [Object](#)< `cv::Size` >
- using **IntObject** = [Object](#)< `int` >
- using **DoubleObject** = [Object](#)< `double` >
- using **StringObject** = [Object](#)< `std::string` >

Functions

- `template<typename T >`
`T get_inner (ParamPtr param_ptr)`
Gets a copy of the inner object.
- `template<typename T >`
`T & get_inner_ref (ParamPtr param_ptr)`
Gets a reference of the inner object.
- `template<typename T >`
`const T & get_inner_const_ref (ParamPtr param_ptr)`
Gets a const reference of the inner object.
- `template<typename T >`
`ParamPtr make_param (T *value_ptr)`
Creates a ParamPtr of an inner object.

7.2.1 Detailed Description

Contains container for various types of data.

7.2.2 Function Documentation

7.2.2.1 get_inner()

```
template<typename T >
T get_inner (
    ParamPtr param_ptr )
```

Gets a copy of the inner object.

Parameters

<code>param_ptr</code>	pointer to the object.
------------------------	------------------------

Returns

A copy of the inner object.

7.2.2.2 get_inner_const_ref()

```
template<typename T >
const T & get_inner_const_ref (
    ParamPtr param_ptr )
```

Gets a const reference of the inner object.

Parameters

<i>param_ptr</i>	pointer to the object.
------------------	------------------------

Returns

A const reference of the inner object.

7.2.2.3 get_inner_ref()

```
template<typename T >
T & get_inner_ref (
    ParamPtr param_ptr )
```

Gets a reference of the inner object.

Parameters

<i>param_ptr</i>	pointer to the object.
------------------	------------------------

Returns

A reference of the inner object.

7.2.2.4 make_param()

```
template<typename T >
ParamPtr make_param (
    T * value_ptr )
```

Creates a ParamPtr of an inner object.

Parameters

<i>value_ptr</i>	a pointer to the inner object.
------------------	--------------------------------

Returns

A ParamPtr of the object.

7.3 object.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 #include<string>
00003 #include<vector>
00004 #include<opencv2/opencv.hpp>
00005
00015 struct ObjectBase {
00016 public:
00017     virtual ~ObjectBase() {}
00018 };
00019
00026 template<typename T>
00027 struct Object : public ObjectBase {
00028 public:
00029     Object(T* ptr);
00030     ~Object();
00031
00032     T         inner();
00033     T&         inner_ref();
00034     const T& inner_const_ref();
00035 private:
00036     T* ptr;
00037 };
00038
00045 template<typename T>
00046 Object<T>::Object(T* ptr) : ptr(ptr) {}
00047
00052 template<typename T>
00053 Object<T>::~~Object() {
00054     delete ptr;
00055 }
00056
00063 template<typename T>
00064 T Object<T>::inner() {
00065     return *ptr;
00066 }
00067
00074 template<typename T>
00075 const T& Object<T>::inner_const_ref() {
00076     return *ptr;
00077 }
00078
00085 template<typename T>
00086 T& Object<T>::inner_ref() {
00087     return *ptr;
00088 }
00089
00090 using ParamPtr = std::shared_ptr<ObjectBase>;
00091 using ParamPtrArray = std::vector<ParamPtr>;
00092
00093 using MatObject = Object<cv::Mat>;
00094 using SizeObject = Object<cv::Size>;
00095 using IntObject = Object<int>;
00096 using DoubleObject = Object<double>;
00097 using StringObject = Object<std::string>;
00098
00104 template<typename T>
00105 T get_inner(ParamPtr param_ptr) {
00106     return dynamic_cast<Object<T>*>(param_ptr.get())->inner();
00107 }
00113 template<typename T>
00114 T& get_inner_ref(ParamPtr param_ptr) {
00115     return dynamic_cast<Object<T>*>(param_ptr.get())->inner_ref();
00116 }
00122 template<typename T>
00123 const T& get_inner_const_ref(ParamPtr param_ptr) {
00124     return dynamic_cast<Object<T>*>(param_ptr.get())->inner_const_ref();
00125 }
00131 template<typename T>
00132 ParamPtr make_param(T* value_ptr) {
00133     return ParamPtr(new Object<T>(value_ptr));
00134 }

```

Index

- convert_color
 - general.cpp, [16](#)
- draw_contours
 - general.cpp, [16](#)
- find_contours
 - general.cpp, [16](#)
- general.cpp, [15](#)
 - convert_color, [16](#)
 - draw_contours, [16](#)
 - find_contours, [16](#)
 - median_blur, [17](#)
 - morph_close, [17](#)
 - morph_open, [17](#)
 - read, [18](#)
 - show, [18](#)
 - threshold, [19](#)
- get_inner
 - object.h, [20](#)
- get_inner_const_ref
 - object.h, [20](#)
- get_inner_ref
 - object.h, [21](#)
- inner
 - Object< T >, [12](#)
- inner_const_ref
 - Object< T >, [12](#)
- inner_ref
 - Object< T >, [12](#)
- make_param
 - object.h, [21](#)
- median_blur
 - general.cpp, [17](#)
- morph_close
 - general.cpp, [17](#)
- morph_open
 - general.cpp, [17](#)
- Object
 - Object< T >, [12](#)
- Object< T >, [11](#)
 - inner, [12](#)
 - inner_const_ref, [12](#)
 - inner_ref, [12](#)
 - Object, [12](#)
- object.h, [19](#)
 - get_inner, [20](#)
 - get_inner_const_ref, [20](#)
 - get_inner_ref, [21](#)
 - make_param, [21](#)
- ObjectBase, [13](#)
- read
 - general.cpp, [18](#)
- show
 - general.cpp, [18](#)
- threshold
 - general.cpp, [19](#)