# Geli Vision

Generated by Doxygen 1.9.6

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

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

# **Geli Vision**

An image processing software.

2 Geli Vision

# 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

# **Hierarchical Index**

# 3.1 Class Hierarchy

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

ObjectBase	 				 					 												13
Object < T >									 										 			11

6 Hierarchical Index

# **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

8 Class Index

# File Index

# 5.1 File List

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

general.	эрр	
	Contains functions for image processing	
object.h		
	Contains container for various types of data	10

10 File Index

# **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.

•  $\sim$ Object ()

The destructor.

• Tinner ()

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

```
\label{eq:typename} \begin{array}{l} \text{template} \! < \! \text{typename T} \! > \\ \text{struct Object} \! < \! \text{T} \! > \end{array}
```

A template class for wrapping objects of type T.

12 Class Documentation

# **Template Parameters**

```
T 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**

ptr 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

14 Class Documentation

# **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 &params)
   Reads an image from file.
- \_\_EXPORT void show (ParamPtrArray &params)
   Shows an image.
- \_\_EXPORT void morph\_open (ParamPtrArray &params)

  Applies morphological opening to an image.
- \_\_EXPORT void morph\_close (ParamPtrArray &params)
- Applies morphological closing to an image.

   \_\_EXPORT void threshold (ParamPtrArray &params)
- \_\_EXPORT void threshold (ParamPtrArray &params)
   Applies morphological closing to an image.
- \_\_EXPORT void convert\_color (ParamPtrArray &params)

  Convert an input image to a different color space.
- \_\_EXPORT void median\_blur (ParamPtrArray &params)

Apply median blur to an input image.

- \_\_EXPORT void find\_contours (ParamPtrArray &params) Find contours in a binary image.
- \_\_EXPORT void draw\_contours (ParamPtrArray &params)

Draw contours in a binary image.

16 File Documentation

# 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**

### params

An array of parameters containing:

- 1. *input* A cv::Mat object representing the image.
- 2. *input* A int representing the conversion type.
- 3. **output** A cv::Mat object representing the processed image.

# 7.1.2.2 draw\_contours()

```
__EXPORT void draw_contours (

ParamPtrArray & params)
```

Draw contours in a binary image.

### **Parameters**

# params

An array of parameters containing:

- 1. *input* A cv::Mat object representing the input binary image.
- 2. input A std::vector<std::vector<cv::Point>> object representing the
   contours.
- 3. **output** A cv::Mat 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**

params	An array of parameters containing:
	1. <i>input</i> A cv::Mat object representing the input binary image.
	<ol> <li>output A std::vector<std::vector<cv::point>&gt; object representing the found contours.</std::vector<cv::point></li> </ol>

# 7.1.2.4 median\_blur()

```
__EXPORT void median_blur (

ParamPtrArray & params )
```

Apply median blur to an input image.

### **Parameters**

params	An array of parameters containing:			
	1. input A cv::Mat object representing the image.			
	2. <i>input</i> A int representing the kernel size.			
	3. output A cv::Mat object representing the processed image.			

# 7.1.2.5 morph\_close()

```
__EXPORT void morph_close (

ParamPtrArray & params )
```

Applies morphological closing to an image.

# **Parameters**

params	An array of parameters containing:				
	1. <i>input</i> A cv::Mat object representing the image.				
	2. <i>input</i> A cv::Size object representing the kernel size for the operation.				
	3. output A cv:: Mat object representing the processed image.				

18 File Documentation

# 7.1.2.6 morph\_open()

```
__EXPORT void morph_open (

ParamPtrArray & params)
```

Applies morphological opening to an image.

# **Parameters**

An array of parameters containing:			
1. <i>input</i> A cv::Mat object representing the image.			
2. <i>input</i> A cv::Size object representing the kernel size for the operation.			
3. output A cv::Mat object representing the processed image.			

# 7.1.2.7 read()

```
__EXPORT void read (

ParamPtrArray & params )
```

Reads an image from file.

### **Parameters**

# params An array of parameters containing: 1. input A string representing the file path of the image. 2. input An integer representing the desired color type of the image. 3. output A cv::Mat object representing the read image.

# 7.1.2.8 show()

```
__EXPORT void show (

ParamPtrArray & params )
```

# Shows an image.

### **Parameters**

params	An array of parameters containing:
	1. <i>input</i> A cv::Mat object representing the image.
	2. output A cv::Mat object representing the image.

### 7.1.2.9 threshold()

```
__EXPORT void threshold (

ParamPtrArray & params)
```

Applies morphological closing to an image.

### **Parameters**

# params An array of parameters containing: 1. input A cv::Mat object representing the image. 2. input A double representing the threshold. 3. input A double representing the max value. 4. input A int representing the threshold type. 5. output A cv::Mat 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 >

20 File Documentation

# **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()

Gets a copy of the inner object.

### **Parameters**

```
param_ptr | pointer to the object.
```

### Returns

A copy of the inner object.

### 7.2.2.2 get inner const ref()

Gets a const reference of the inner object.

7.3 object.h

### **Parameters**

param_ptr	pointer to the object.
-----------	------------------------

# Returns

A const reference of the inner object.

# 7.2.2.3 get\_inner\_ref()

Gets a reference of the inner object.

# **Parameters**

param ptr	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**

value_ptr	a pointer to the inner object.
-----------	--------------------------------

### Returns

A ParamPtr of the object.

# 7.3 object.h

Go to the documentation of this file.

22 File Documentation

```
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
        T
T&
00032
                    inner();
00033
          T& inner_ref();
const T& inner_const_ref();
00035 private:
00036
         T* ptr;
00037 };
00038
00045 template<typename T>
00046 Object<T>::Object(T* ptr) : ptr(ptr) {}
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) {
          return ParamPtr(new Object<T>(value_ptr));
00134 }
```

# Index

```
convert_color
                                                            get_inner_const_ref, 20
    general.cpp, 16
                                                            get_inner_ref, 21
                                                            make_param, 21
draw_contours
                                                       ObjectBase, 13
    general.cpp, 16
                                                       read
find_contours
                                                            general.cpp, 18
    general.cpp, 16
                                                       show
general.cpp, 15
                                                            general.cpp, 18
    convert_color, 16
    draw_contours, 16
                                                       threshold
    find_contours, 16
                                                            general.cpp, 19
    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
```

get\_inner, 20

object.h, 19