

# API Documentation

API Documentation

September 12, 2012

## Contents

<b>Contents</b>	<b>1</b>
<b>1 Package src</b>	<b>3</b>
1.1 Modules . . . . .	3
1.2 Variables . . . . .	3
<b>2 Module src.camera</b>	<b>4</b>
2.1 Variables . . . . .	4
2.2 Class Camera . . . . .	4
2.2.1 Methods . . . . .	4
<b>3 Module src.cameras</b>	<b>5</b>
3.1 Variables . . . . .	5
3.2 Class Cameras . . . . .	5
3.2.1 Methods . . . . .	5
<b>4 Module src.check_goban_moved</b>	<b>6</b>
4.1 Functions . . . . .	6
4.2 Variables . . . . .	6
<b>5 Module src.cte</b>	<b>7</b>
5.1 Variables . . . . .	7
<b>6 Module src.functions</b>	<b>9</b>
6.1 Functions . . . . .	9
6.2 Variables . . . . .	9
<b>7 Module src.goban</b>	<b>10</b>
7.1 Variables . . . . .	10
7.2 Class Goban . . . . .	10
7.2.1 Methods . . . . .	10
<b>8 Module src.igs</b>	<b>11</b>
8.1 Variables . . . . .	11
8.2 Class Igs . . . . .	11
8.2.1 Methods . . . . .	11
<b>9 Module src.kifu</b>	<b>12</b>

9.1	Variables . . . . .	12
9.2	Class Kifu . . . . .	12
9.2.1	Methods . . . . .	12
<b>10</b>	<b>Module src.perspective</b>	<b>13</b>
10.1	Functions . . . . .	13
10.2	Variables . . . . .	13
<b>11</b>	<b>Module src.rocango</b>	<b>14</b>
11.1	Functions . . . . .	14
11.2	Variables . . . . .	14
<b>12</b>	<b>Module src.search_goban</b>	<b>15</b>
12.1	Functions . . . . .	15
12.2	Variables . . . . .	15
<b>13</b>	<b>Module src.search_stones</b>	<b>16</b>
13.1	Functions . . . . .	16
13.2	Variables . . . . .	16
<b>14</b>	<b>Module src.stone</b>	<b>17</b>
14.1	Variables . . . . .	17
14.2	Class Stone . . . . .	17
14.2.1	Methods . . . . .	17
	<b>Index</b>	<b>18</b>

# 1 Package src

## 1.1 Modules

- **camera** (*Section 2, p. 4*)
- **cameras** (*Section 3, p. 5*)
- **check\_goban\_moved** (*Section 4, p. 6*)
- **cte** (*Section 5, p. 7*)
- **functions** (*Section 6, p. 9*)
- **goban** (*Section 7, p. 10*)
- **igs** (*Section 8, p. 11*)
- **kifu** (*Section 9, p. 12*)
- **perspective** (*Section 10, p. 13*)
- **rocango** (*Section 11, p. 14*)
- **search\_goban** (*Section 12, p. 15*)
- **search\_stones** (*Section 13, p. 16*)
- **stone** (*Section 14, p. 17*)

## 1.2 Variables

Name	Description
__package__	<b>Value:</b> None

## 2 Module src.camera

### 2.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

### 2.2 Class Camera

Clase para inicializar la cámara.

#### 2.2.1 Methods

<code>__init__(self)</code>
<code>open_camera(self, index)</code> Abrir cámara con opencv. :param index: índice de la cámara. :type index: int.
<code>get_frame(self)</code> Obtener una imagen desde la cámara.
<code>is_open(self)</code> Comprueba si la cámara está abierta.
<code>close_camera(self)</code> Cierra la cámara.

### 3 Module src.cameras

#### 3.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

#### 3.2 Class Cameras

Clase para abrir las cámaras disponibles en el ordenador.

##### 3.2.1 Methods

<b><code>__init__(self)</code></b>
<b><code>on_mouse(self, event, x, y, flags, camera)</code></b> Capturador de eventos de click de ratón. :param event: Evento capturado. :type event: int :param x: posición x del ratón. :type x: int :param y: posición y del ratón. :type y: int :param camera: objeto Camera. :type camera: Camera
<b><code>check_cameras(self, num=99)</code></b> Comprueba las cámaras disponibles. :param num: máximo número de cámaras a comprobar :keyword num: el valor por defecto es 99, ya que en Linux es lo permitido :param num: int :return: lista de cámaras disponibles :rtype: list of Camera
<b><code>show_and_select_camera(self)</code></b> Muestra las cámaras disponibles en ventanas y da la opción de seleccionar una de ellas pulsando doble click. :return: cámara seleccionada :rtype: Camera

## 4 Module `src.check_goban_moved`

### 4.1 Functions

**`is_same_quadrant(v1, v2)`**

Comprueba si dos vectores pasados por parámetros se encuentran en el mismo cuadrante.  
:param v1: vector :type v1: tuple :param v2: vector :type v2: tuple :return: True si se encuentran los vectores en el mismo cuadrante. :rtype: bool

**`degress_between_two_vectors(v1, v2)`**

Halla los grados que existen entre dos vectores dados. :param v1: vector :type v1: tuple  
:param v2: vector :type v2: tuple :return: grados en radianes :rtype: float

**`check_directions(directions)`**

Comprueba si las direcciones entre los 4 vectores de movimiento de los corners del tablero tienen la misma dirección. :param directions: lista de vectores directores :type directions: list :return: True si todos o ninguno de los vectores tienen la misma dirección. :rtype: bool

**`check_goban_moved(prev_corners, current_corners)`**

Comprobamos si es posible el movimiento de tablero detectado. :param prev\_corners: corners detectados anteriormente :type prev\_corners: list :param current\_corners: corners detectados actualmente :type current\_corners: list :return: True si el tablero se ha movido :rtype: bool

### 4.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

## 5 Module src.cte

### 5.1 Variables

Name	Description
NUM_EDGES	Value: 4
RELATION_WEIGHT_HEIGHT	Value: 0.933333333333
MAX_CAMERAS	Value: 99
GOBAN_SIZE	Value: 19
BLACK	Value: 1
WHITE	Value: 2
current_date	Value: '12 Sep 2012'

*continued on next page*

Name	Description
HEADER_SGF	<p>FEATURES OF FILES .SGF</p> <p>=====</p> <p>* AB: Add Black: locations of Black stones to be placed on the board prior to the first move. * AW: Add White: locations of White stones to be placed on the board prior to the first move. * AN: Annotations: name of the person commenting the game. * AP: Application: application that was used to create the SGF file (e.g. CGOban2,...). * B: a move by Black at the location specified by the property value. * BR: Black Rank: rank of the Black player. * BT: Black Team: name of the Black team. * C: Comment: a comment. * CP: Copyright: copyright information. See Kifu Copyright Discussion. * DT: Date: date of the game. * EV: Event: name of the event (e.g. 58th Honinbo Title Match). * FF: File format: version of SGF specification governing this SGF file. * GM: Game: type of game represented by this SGF file. A property value of 1 refers to Go. * GN: Game Name: name of the game record. * HA: Handicap: the number of handicap stones given to Black. Placement of the handicap stones are set using the AB property. * KM: Komi: komi. * ON: Opening: information about the opening (fuseki), rarely used in any file. * OT: Overtime: overtime system. * PB: Black Name: name of the black player. * PC: Place: place where the game was played (e.g.: Tokyo). * PL: Player: color of player to start. * PW: White Name: name of the white player. * RE: Result: result, usually in the format "B+R" (Black wins by resign) or "B+3.5" (black wins by 3.5 moku). * RO: Round: round (e.g.: 5th game). * RU: Rules: ruleset (e.g.: Japanese). * SO: Source: source of the SGF file. * SZ: Size: size of the board, non square boards are supported. * TM: Time limit: time limit in seconds. * US: User: name of the person who created the SGF file. * W: a move by White at the location specified by the property value. * WR: White Rank: rank of the White player. * WT: White Team: name of the White team.</p> <p><b>Value:</b> ['(;FF[4]GM[1]SZ[19]', '\nAP[Rocamgo]', '\nHA[0]', '\nKM[...</p>
__package__	<b>Value:</b> 'src'



## 6 Module *src.functions*

### 6.1 Functions

**distance\_between\_two\_points**(*p1*, *p2*)

Halla la distancia entre dos puntos dados. :param p1: punto 1 :type p1: tuple :param p2: punto 2 :type p2: tuple :return: distancia entre dos puntos :rtype: float

**direction\_between\_two\_points**(*p1*, *p2*)

Halla la direccion entre dos puntos dados. :param p1: punto 1 :type p1: tuple :param p2: punto 2 :type p2: tuple :return: dirección del punto 1 al punto 2 :rtype: tuple

**get\_max\_edge**(*corners*)

Halla la arista más larga dado 4 puntos. :param corners: lista de 4 puntos :type corners: list :return: máxima distancia entre 4 puntos :rtype: int

**get\_external\_corners**(*corners*)

Halla los corners externos en el caso de que haber capturando los internos. :param corners: lista de 4 puntos :type corners: list :return: lista con los 4 corners exteriores :rtype: list

### 6.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

## 7 Module src.goban

### 7.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

### 7.2 Class Goban

Clase tablero, contiene la matriz de estadísticas y funciones para rellenar el tablero.

#### 7.2.1 Methods

<b><code>__init__</code></b> ( <i>self</i> , <i>size</i> )
Crea dos matrices de tamaño pasado por parámetro, una para estadísticas y otra para guardar el estado de las piedras. Creamos un set de piedras para ir guardando las piedras que estemos comprobando. También inicializa un kifu para guardar la partida y un el objetos igs que se encargará de conectarse con el servidor que subirá la partida. :param size: tamaño del tablero :type size: int

<b><code>add_stones_to_statistical</code></b> ( <i>self</i> , <i>stones</i> )
Recorremos la lista de piedras pasadas por parámetros para buscar hacer comprobaciones estadísticas en esas piedras, luego recorremos la lista de piedras guardada y la actualizamos. Actualiza kifu, igs y el tablero donde guardamos el estado de las piedras cuando detecta estadísticamente que una piedra se ha puesto. :param stones: lista de piedras :type stones: list

<b><code>print_st</code></b> ( <i>self</i> )
--

<b><code>__str__</code></b> ( <i>self</i> )
---

## 8 Module src.igs

### 8.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

### 8.2 Class Igs

Clase que se comunica con el servidor de IGS.

#### 8.2.1 Methods

<b><code>__init__</code></b> ( <i>self</i> , <i>user</i> ='rocamgo', <i>pwd</i> ='qwe')
Inicializamos la conexión con el servidor y creamos un tablero de aprendizaje dentro del servidor para comenzar a subir la partida. :param user: usuario que se conectará al servidor :type user: str :param password: contraseña del usuario para conetarse al servidor :type password: str
<b><code>add_stone</code></b> ( <i>self</i> , <i>pos</i> )
Añadimos piedra al servidor. :param pos: posición de la piedra a añadir :type pos: tuple
<b><code>close</code></b> ( <i>self</i> )
Cerramos la conexión con el servidor.

## 9 Module src.kifu

### 9.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

### 9.2 Class Kifu

Clase para crear un fichero .sgf y guardar la partida.

#### 9.2.1 Methods

**`__init__`**(*self*, *player1*='j1', *player2*='j2', *handicap*=0, *path*='sgf',  
*rank\_player1*='20k', *rank\_player2*='20k')

Inicializamos configuración del archivo sgf. :param *player1*: nombre del jugador 1 :type *player1*: str :keyword *player1*: j1 por defecto :param *player2*: nombre del jugador 2 :type *player2*: str :keyword *player2*: j2 por defecto :param *handicap*: handicap dado en la partida :type *handicap*: int :keyword *handicap*: ninguno por defecto (0) :param *path*: ruta relativa donde guardamos el fichero :type *path*: str :keyword *path*: carpeta sgf por defecto :param *rank\_player1*: rango del jugador 1 :type *rank\_player1*: str :keyword *rank\_player1*: 20k por defecto, nivel de inicio en el go :param *rank\_player2*: rango del jugador 2 :type *rank\_player2*: str :keyword *rank\_player2*: 20k por defecto, nivel de inicio en el go

**`add_stone`**(*self*, *pos*, *color*)

Añadir piedra al sgf. :param *pos*: posición de la piedra :type *pos*: tuple :param *color*: color de la piedra :type *color*: int

**`end_file`**(*self*)

Cerrar el fichero y dejarlo listo para poder abrirlo.

## 10 Module *src.perspective*

### 10.1 Functions

**perspective**(*img*, *corners*)

Crea una imagen en modelo ideal del tablero dado en perspectiva. :param img: imagen con el tablero en perspectiva :todo comprobar de que tipo es la imagen TODO :type img: IplImage or CvMat :param corners: lista de las esquinas del tablero :type corners: list :return: imagen en modelo ideal :rtype: IplImage

### 10.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

## 11 Module *src.rocamgo*

### 11.1 Functions

<code>main()</code>
---------------------

### 11.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'src'</code>
<code>cam</code>	Objeto Cameras ( <i>type=Cameras</i> )
<code>camera</code>	cámara que estamos usando ( <i>type=Camera</i> )
<code>cams_found</code>	número de cámaras encontradas en el ordenador ( <i>type=int</i> )
<code>current_corners</code>	esquinas actuales del tablero encontradas ( <i>type=list</i> )
<code>goban</code>	Objeto tablero ( <i>type=Goban</i> )
<code>good_corners</code>	últimas esquinas buenas encontradas ( <i>type=list</i> )
<code>ideal_img</code>	tablero en formato ideal ( <i>type=IpImage</i> )
<code>img</code>	imagen actual sacada de la cámara o video ( <i>type=IpImage</i> )
<code>prev_corners</code>	esquinas del tablero anteriores encontradas ( <i>type=list</i> )

## 12 Module `src.search_goban`

### 12.1 Functions

#### `count_perimeter(seq)`

Contamos el perímetro de una secuencia dada. :param seq: secuencia de puntos :type seq: CvSeq :return: distancia del perímetro :rtype: float

#### `get_corners(contour)`

Hallamos las esquinas a partir de un contorno y las ordenamos de la siguiente manera: ul, dl, ur, dr. u = up, l = left, d = down, r = right. :param contour: contorno del tablero obtenido :type contour: CvSeq :return: lista de esquinas :rtype: list

#### `filter_image(img)`

Aplicamos unos filtros a las imágenes para facilitar su tratamiento. Buscamos contornos y suavizamos. :param img: imagen sin filtrar :type img: CvMat :return: imagen filtrada :rtype: CvMat

#### `detect_contour(img)`

Buscamos contornos con unas características determinadas para encontrar un tablero de go en una imagen. :param img: imagen filtrada para buscar contornos en ella :type img: CvMat :return: Contorno si no lo encuentra, sino None :rtype: CvSeq

#### `search_goban(img)`

Busca el tablero en una imagen. :param img: imagen del tablero :type img: IplImage # TODO comprobar tipo imagen :return: lista de esquinas si las encuentra, sino None :rtype: list or None

### 12.2 Variables

Name	Description
<code>__package__</code>	Value: 'src'

## 13 Module `src.search_stones`

### 13.1 Functions

**`search_stones`**(*img*, *corners*, *dp=1.7*)

Devuelve las circunferencias encontradas en una imagen. :param *img*: imagen donde buscaremos las circunferencias :type *img*: `IplImage` :param *corners*: lista de esquinas :type *corners*: list :param *dp*: profundidad de búsqueda de círculos :type *dp*: int :keyword *dp*: 1.7 era el valor que mejor funcionaba. Prueba y error

**`check_color_stone`**(*pt*, *radius*, *img*, *threshold=190*)

Devuelve el color de la piedra dado el centro y el radio de la piedra y una imagen. También deseamos las piedras que no sean negras o blancas. :param *pt*: centro de la piedra :type *pt*: tuple :param *radius*: radio de la piedra :type *radius*: int :param *img*: imagen donde comprobaremos el color de ciertos pixeles :type *img*: `IplImage` :param *threshold*: umbral de blanco :type *threshold*: int :keyword *threshold*: 190 cuando hay buena luminosidad

### 13.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'src'</code>



## 14 Module src.stone

### 14.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'src'

### 14.2 Class Stone

Clase piedra.

#### 14.2.1 Methods

<b><code>__init__</code></b> ( <i>self</i> , <i>color</i> , <i>img</i> =None, <i>pix</i> =None, <i>pt</i> =None)
Inicializamos una piedra, si no tenemos la posición, buscamos cual es esa posición dado una imagen ideal y un pixel. :param color: color de la piedra, BLACK or WHITE :type color: int :param img: imagen en formato ideal :type img: IplImage :keyword img: None si no le pasamos ninguna imagen por parámetro :param pix: pixel donde se encuentra la piedra en la imagen :type pix: tuple :keyword pix: None si no le pasamos ningún pixel por parámetro :param pt: punto donde se encuentra la piedra en el tablero :type pt: tuple :keyword pt: None si no le pasamos ningún punto parámetro.
<b><code>__str__</code></b> ( <i>self</i> )
<b><code>__eq__</code></b> ( <i>self</i> , <i>st</i> )
<b><code>__cmp__</code></b> ( <i>self</i> , <i>st</i> )
<b><code>__hash__</code></b> ( <i>self</i> )

# Index

- src (*package*), 3
  - src.camera (*module*), 4
    - src.camera.Camera (*class*), 4
  - src.cameras (*module*), 5
    - src.cameras.Cameras (*class*), 5
  - src.check\_goban\_moved (*module*), 6
    - src.check\_goban\_moved.check\_directions (*function*), 6
    - src.check\_goban\_moved.check\_goban\_moved (*function*), 6
    - src.check\_goban\_moved.degress\_between\_two\_vectors (*function*), 6
    - src.check\_goban\_moved.is\_same\_quadrant (*function*), 6
  - src.cte (*module*), 7–8
  - src.functions (*module*), 9
    - src.functions.direction\_between\_two\_points (*function*), 9
    - src.functions.distance\_between\_two\_points (*function*), 9
    - src.functions.get\_external\_corners (*function*), 9
    - src.functions.get\_max\_edge (*function*), 9
  - src.goban (*module*), 10
    - src.goban.Goban (*class*), 10
  - src.igs (*module*), 11
    - src.igs.Igs (*class*), 11
  - src.kifu (*module*), 12
    - src.kifu.Kifu (*class*), 12
  - src.perspective (*module*), 13
    - src.perspective.perspective (*function*), 13
  - src.rocango (*module*), 14
    - src.rocango.main (*function*), 14
  - src.search\_goban (*module*), 15
    - src.search\_goban.count\_perimeter (*function*), 15
    - src.search\_goban.detect\_contour (*function*), 15
    - src.search\_goban.filter\_image (*function*), 15
    - src.search\_goban.get\_corners (*function*), 15
    - src.search\_goban.search\_goban (*function*), 15
  - src.search\_stones (*module*), 16
    - src.search\_stones.check\_color\_stone (*function*), 16
    - src.search\_stones.search\_stones (*function*), 16
  - src.stone (*module*), 17
    - src.stone.Stone (*class*), 17