API Documentation

API Documentation

September 12, 2012

Contents

1 Package src

1.1 Modules

1.2 Variables

Name	Description
___package___	**Value:** `None`

2 Module src.camera

2.1 Variables

Name	Description
___package___	**Value:** `'src'`
capture	valor de la captura actual *(type=Capture)*
index	índice de cámara *(type=int)*

2.2 Class Camera

Clase para inicializar la cámara.

2.2.1 Methods

___init___(*self*)

open_camera(*self*, *index*)

Abrir cámara con opencv. :param index: índice de la cámara. :type index: int.

get_frame(*self*)

Obtener una imagen desde la cámara.

is_open(*self*)

Comprueba si la cámara está abierta.

close_camera(*self*)

Cierra la cámara.

3 Module src.cameras

3.1 Variables

Name	Description
___package___	**Value:** `'src'`
camera	cámara seleccionada *(type=Capture)*
cameras	lista de cámaras *(type=list)*

3.2 Class Cameras

Clase para abrir las cámaras disponibles en el ordenador.

3.2.1 Methods

___init___(*self*)

on_mouse(*self*, *event*, *x*, *y*, *flags*, *camera*)

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

check_cameras(*self*, *num*=99)

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

show_and_select_camera(*self*)

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 vecores de movimiento de los corners del tablego 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
___package___	**Value:** `'src'`

5 Module src.cte

5.1 Variables

Name	Description
NUM_EDGES	número de esquinas que existen en un tablero **Value:** `4` *(type=int)*
RELATION_WEIGHT_HEIGHT	relación anchura/altura que existe en el tablero **Value:** `0.933333333333` *(type=float)*
MAX_CAMERAS	número máximo de cámaras a buscar **Value:** `99` *(type=int)*
GOBAN_SIZE	tamaño del tablero **Value:** `19` *(type=int)*
BLACK	constante para decir que una piedra es negra **Value:** `1` *(type=int)*
WHITE	constante para decir que una piedra es blanca **Value:** `2` *(type=int)*
current_date	**Value:** `'12 Sep 2012'`

continued on next page

Name	Description
HEADER_SGF	• 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. Placment 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. **Value:** `['(;FF[4]GM[1]SZ[19]', '\nAP[Rocamgo]', '\nHA[0]', '\nKM[...` *(type=str)*

continued on next page

Name	Description
___package___	**Value:** `'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
___package___	**Value:** `'src'`

7 Module src.goban

7.1 Variables

Name	Description
___package___	**Value:** `'src'`
goban	matriz de piedras puestas *(type=list)*
igs	Objeto Igs *(type=Igs)*
kifu	Objeto Kifu *(type=Kifu)*
statistical	matriz de estadísticas para comprobar piedras buenas o malas *(type=list)*
stones	piedras a comprobar para añadir a estadísticas *(type=list)*

7.2 Class Goban

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

7.2.1 Methods

___init___(*self*, *size*)

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

add__stones__to__statistical(*self*, *stones*)

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

print__st(*self*)

___str___(*self*)

8 Module src.igs

8.1 Variables

Name	Description
___package___	**Value:** `'src'`
pwd	password correspondiente al usuario de Igs *(type=str)*
s	socket para la conexión con el servidor *(type=socket)*
user	usuario del servidor Igs *(type=str)*

8.2 Class Igs

Clase que se comunica con el servidor de IGS.

8.2.1 Methods

___init___(*self*, *user*=`'rocamgo'`, *pwd*=`'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

add_stone(*self*, *pos*)

Añadimos piedra al servidor. :param pos: posición de la piedra a añadir :type pos: tuple

close(*self*)

Cerramos la conexión con el servidor.

9 Module src.kifu

9.1 Variables

Name	Description
___package___	**Value:** `'src'`
dir	dirección del directorio donde guardaremos la partida *(type=str)*
handicap	numero de piedras de ventaja *(type=int)*
num_jug	número de jugada actual *(type=int)*
path	directorio donde guardaremos las partidas *(type=str)*
player1	nombre del jugador 1 *(type=str)*
player2	nombre del jugador 2 *(type=str)*
rank_player1	nivel del jugador 1 *(type=str)*
rank_player2	nivel del jugador 2 *(type=str)*

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
___package___	**Value:** `'src'`

11 Module src.rocamgo

11.1 Functions

main()

11.2 Variables

Name	Description
___package___	**Value:** `'src'`
cam	Objeto Cameras *(type=Cameras)*
camera	cámara que estamos usando *(type=Camera)*
cams_found	número de cámaras encontradas en el ordenador *(type=int)*
circles	circulos encontrado en la imagen *(type=CvMat)*
color	color de la piedra *(type=int)*
current_corners	esquinas actuales del tablero encontradas *(type=list)*
false_stones	contador para piedras falsas, no son negras o blancas *(type=int)*
goban	Objeto tablero *(type=Goban)*
good_corners	últimas esquinas buenas encontradas *(type=list)*
ideal_img	tablero en formato ideal *(type=IplImage)*
img	imagen actual sacada de la cámara o video *(type=IplImage)*
key	tecla pulsada *(type=int)*
prev_corners	esquinas del tablero anteriores encontradas *(type=list)*
pt	centro de la piedra *(type=tuple)*
radious	radio de la piedra *(type=float)*
stones	piedras detectadas como negras o blancas *(type=list)*

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
___package___	**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*, *radious*, *img*, *threshold*=`190`)

Devuelve el color de la piedra dado el centro y el radio de la piedra y una imagen. También desechamos las piedras que no sean negras o blancas. :param pt: centro de la piedra :type pt: tuple :param radious: radio de la piedra :type radious: 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
___package___	**Value:** `'src'`

14 Module src.stone

14.1 Variables

Name	Description
___package___	**Value:** `'src'`
color	color de la piedra *(type=int)*
img	imagen donde se encuentra la piedra *(type=IplImage)*
pix	pixel donde se encuentra la piedra dentro de la imagen *(type=tuple)*
pt	coordenada del tablero donde se encuentra la piedra *(type=tuple)*
x	coordenada x del tablero donde se encuentra la piedra *(type=int)*
y	coordenada y del tablero donde se encuentra la piedra *(type=int)*

14.2 Class Stone

Clase piedra.

14.2.1 Methods

___init___(*self*, *color*, *img*=`None`, *pix*=`None`, *pt*=`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 ningun 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.

___str___(*self*)

___eq___(*self*, *st*)

___cmp___(*self*, *st*)

___hash___(*self*)

Index