Brigham Young University
AUVSI Capstone Team (Team 45)

GUI API

ID	Rev.	Date	Description	Author	Checked By
IM-002	1.0	12-07-2018	Initial release	Derek Knowles & Brandon McBride	Connor Olsen

API Documentation

API Documentation

December 7, 2018

Contents

1 Module client_gui

Authors: D. Knowles, B. McBride, T. Miller

Prereqs: python 3 sudo apt install python3-tk pip3 install Pillow, opencv-python, ttkthemes

1.1 Class GuiClass

tkinter.Frame ———┐

              client_gui.GuiClass

Graphical User Interface for 2019 AUVSI competition Tab 0: Setting for setting up the server_error Tab 1: Pull raw images and submit cropped images Tab 2: Pull cropped iamges and submit classification for images Tab 3: Display results for manual and autonomous classification

1.1.1 Methods

__init__(*self, master=*`None`)

initialization for Gui Class

Parameters
    `tk.Frame:` nothing

               (type=nothing)

Return Value
    None

    (type=None)

---
get_image(*self, path*)

Reads in an image from folder on computer

Parameters
    `path:` the file path to where the image is located

               (type=file path)

Return Value
    Numpy array of selected image

    (type=Numpy image array)

---

np2im(*self, image*)

Converts from numpy array to PIL image

Parameters
    `image`: Numpy array of selected image

        (type=Numpy image array)

Return Value
    PIL image of numpy array

    (type=PIL image)

---

im2tk(*self, image*)

Converts from PIL image to TK image

Parameters
    `image`: PIL image of numpy array

        (type=PIL image)

Return Value
    TK image of PIL image

    (type=TK image)

---

mouse_click(*self, event*)

Saves pixel location of where on the image the mouse clicks

Parameters
    `event`: mouse event

        (type=event)

Return Value
    None

    (type=None)

---

mouse_move(*self, event*)

Gets pixel location of where the mouse is moving and show rectangle for crop preview

Parameters
    `event`: mouse event

        (type=event)

Return Value
    None

    (type=None)

---

mouse_release(*self, event*)

Saves pixel location of where the mouse clicks and creates crop preview

Parameters
    `event:` mouse event

            (type=event)

Return Value
    None

    (type=None)

---

close_window(*self, event*)

Closes gui safely

Parameters
    `event:` ESC event

            (type=event)

Return Value
    None

    (type=None)

---

resizeEventTab0(*self, event=*`None`)

Resizes picture on Tab0

Parameters
    `event:` resize window event

            (type=event)

Return Value
    None

    (type=None)

---

resizeEventTab1(*self, event=*`None`)

Resizes pictures on Tab1

Parameters
    `event:` resize window event

            (type=event)

Return Value
    None

    (type=None)

resizeEventTab2(*self, event=*None)

Resizes picture on Tab2

Parameters

    event: resize window event

        (type=event)

Return Value

    None

    (type=None)

---

resizeIm(*self, image, image_width, image_height, width_restrict, height_restrict*)

Resizes PIL image according to given bounds

Parameters

    image: PIL image that you want to crop

                        (type=PIL image)

    image_width: the original image width in pixels

                        (type=integer)

    image_height: the original image height in pixels

                        (type=integer)

    width_restrict: the width in pixels of restricted area

                        (type=integer)

    height_restrict: the height in pixels of restricted area

                        (type=integer)

Return Value

    Resized PIL image

    (type=PIL image)

---

cropImage(*self, x0, y0, x1, y1*)

Crops raw image

Parameters

    x0: pixel x location of first click

        (type=integer)

    y0: pixel y location of first click

        (type=integer)

    x1: pixel x location of second click

        (type=integer)

    y1: pixel y location of second click

        (type=integer)

Return Value

    None

    (type=None)

undoCrop(*self*, *event*=`None`)

Undoes crop and resets the raw image

Parameters
> `event`: Ctrl + Z event
>> *(type=event)*

Return Value
> None
>
> *(type=None)*

---

nextRaw(*self*, *event*)

Requests and displays next raw image

Parameters
> `event`: Right arrow event
>> *(type=event)*

Return Value
> None
>
> *(type=None)*

---

previousRaw(*self*, *event*)

Requests and displays previous raw image

Parameters
> `event`: Left arrow event
>> *(type=event)*

Return Value
> None
>
> *(type=None)*

---

submitCropped(*self*, *event*=`None`)

Submits cropped image to server

Parameters
> `event`: Enter press or button press event
>> *(type=event)*

Return Value
> None
>
> *(type=None)*

nextCropped(*self, event*)

Requests and displays next cropped image

Parameters
    event: Right arrow event

            (type=event)

Return Value
    None

    (type=None)

---

prevCropped(*self, event*)

Requests and displays previous cropped image

Parameters
    event: Left arrow event

            (type=event)

Return Value
    None

    (type=None)

---

submitClassification(*self, event=*None)

Submits classification of image to server

Parameters
    event: Enter press event

            (type=event)

Return Value
    None

    (type=None)

---

tabChanged(*self, event*)

Performs the correct keybindings when you move to a new tab of the gui

Parameters
    event: Tab changed event

            (type=event)

Return Value
    None

    (type=None)

updateSettings(*self, event*=None)

Attempts to connect to server when settings are changed

Parameters
> event: Enter press or button press event
>> *(type=event)*

Return Value
> None
>
> *(type=None)*

---

pingServer(*self*)

Checks if server is correctly connected

Return Value
> None
>
> *(type=None)*

---

disableEmergentDescription(*self, *args*)

Disables emergent discription unless emergent target selected

Return Value
> None
>
> *(type=None)*

Index