BYU AUVSI

Brigham Young University

AUVSI Capstone Team (Team 45)

Imaging GUI Server Interface API

ID	Rev.	Date	Description	Author	Checked By
IM-001	1.0	12-07-2018	Initial release	Brandon McBride & Derek Knowles	Connor Olsen

1 Introduction

The purpose of this artifact is to document the code that interfaces with the GUI and server. This document lists a brief description for each function and the functions input and output parameters.

2 Module client_rest

2.1 Functions

testNextAndPrevRawImage(*interface*)

testNextAndPrevCroppedImage(*interface*)

testCropPost(*interface*, *imgId*)

2.2 Variables

Name	Description
__package__	**Value:** `None`

2.3 Class ImageInfo

An ImageInfo object holds the information concerning an image such as the image id, the image path, the timestamp the image was taken, and whether or not it has been seen by the autonomous or manual system.

2.3.1 Methods

__init__(*self*, *auto_tap*, *imageId*, *path*, *man_tap*, *ts*)

The constructor for the ImageInfo class.

Parameters

`auto_tap:`	True if the autonomous system has seen the image, False otherwise *(type=boolean)*
`imageId:`	the id related to the image *(type=int)*
`path:`	the file path to where the image is located *(type=String)*
`man_tap:`	True if the manual system has seen the image, False otherwise *(type=boolean)*
`ts:`	the timestamp that the image was taken *(type=float)*

Return Value

None

(type=None)

2.4 Class CropInfo

A CropInfo object holds information concerning a cropped image such as the image id to the original image, the top-left and bottom-right coordinates to where the crop took place on the original image, the path of the cropped image, whether or not the cropped image has been seen by the imaging system, and the timestamp of the original image.

2.4.1 Methods

__init__(*self*, *imgId*, *tl*, *br*, *path*, *isTapped*, *ts*)

The constructor for the CropInfo class.

Parameters

`imgId:`	the id of the original image *(type=int)*
`tl:`	the x and y coordinates of the original image of the top left corner to where the image was cropped *(type=int[])*
`br:`	the x and y coordinates of the original image of the bottom right corner to where the image was cropped *(type=int[])*
`path:`	the file path to where the cropped image is located *(type=String)*
`isTapped:`	True if the manual imaging system has seen this image, False otherwise *(type=boolean)*
`ts:`	the timestamp that the image was taken *(type=float)*

Return Value

None

(type=None)

2.5 Class GPSMeasurement

A GPSMeasurement object holds information concerning one measurement taken from the aircraft's GPS. Such information includes the id of the measurement, altitude in meters, longitude, latitude, and the timestamp of when the measurement was taken.

2.5.1 Methods

__init__(*self*, *alt*, *gpsId*, *lat*, *long*, *ts*)

The constructor for the GPSMeasurement class.

Parameters

`alt`: the altitude of the measurement
(type=float)

`gpsId`: the id of the GPS measurement
(type=int)

`lat`: the latitude of the measurement
(type=float)

`long`: the longitude of the measurement
(type=float)

`ts`: the timestamp that the measurement was taken
(type=float)

Return Value

None
(type=None)

2.6 Class StateMeasurement

A StateMeasurement object holds information concerning the state of the aircraft in one point in time. Such information includes the id of the measurement, roll angle of the aircraft in radians, pitch angle of the aircraft in radians, yaw angle of the aircraft in radians, and the timestamp of when the measurement was taken.

2.6.1 Methods

__init__(*self*, *stateId*, *roll*, *pitch*, *yaw*, *ts*)

The constructor for the StateMeasurement class.

Parameters

`stateId:`	the id of the state measurement *(type=int)*
`roll:`	the roll angle in radians of the aircraft at the time the measurement was taken *(type=float)*
`pitch:`	the pitch angle in radians of the aircraft at the time the measurement was taken *(type=float)*
`yaw:`	The yaw angle in radians of the aircraft at the time the measurement was taken *(type=float)*
`ts:`	the timestamp that the measurement was taken *(type=float)*

Return Value

None

(type=None)

2.7 Class ImagingInterface

The ImagingInterface object serves as the bridge between the imaging server and the imaging GUI. It connects to the imaging server given an IP address and a port. It makes certain calls to the server in order to feed the GUI what it needs such as images and measurements taken by the imaging system on the aircraft.

2.7.1 Methods

__init__(*self*, *host*=`'127.0.0.1'`, *port*=`'5000'`, *numIdsStored*=`50`, *isDebug*=`False`)

The constructor for the ImagingInterface class.

Parameters

`host:`	the host of the server that the interface connects to *(type=String)*
`port:`	the port of the server that the interface connects to *(type=String)*
`numIdsStored:`	the number of ids that the interface keeps track of, used for getting previous images *(type=int)*
`isDebug:`	if isDebug is true, the interface will print out status statements *(type=boolean)*

Return Value

None

(type=None)

ping(*self*)

Checks to see if the interface can contact the server.

Return Value

True if the interface contacted the server, False otherwise

(type=boolean)

debug(*self*, *printStr*)

If interface is in debug mode, it will print the string given, else it does nothing.

Parameters

`printStr:`	the string that will be printed if in debug mode *(type=String)*

Return Value

None

(type=None)

getRawImage(*self*, *imageId*)

Retrieves an image with the given imageId from the server.

Parameters

`imageId`: the id of the image that is going to be returned

(type=int)

Return Value

a tuple of the pillow Image associated with the given image id and the image id if there are any images available for processing, otherwise None

(type=(Image, int))

getNextRawImage(*self*, *isManual*)

Retrieves the next available raw image from the server.

Parameters

`isManual`: specify whether this is a manual imaging request (True) or an autonomous one (False)

(type=boolean)

Return Value

a tuple of a pillow Image and the image id if there are any images available for processing, otherwise None

(type=(Image, int))

getPrevRawImage(*self*)

Re-retrieves a raw image that was previously viewed. The interface maintains an ordered list (of up to numIdsStored) of ids that has previously been viewed and traverses the list backwards.

Return Value

a tuple of a pillow Image and the image id if there are any previous images to process, and the server is able to find the given id, otherwise None.

(type=(Image, int))

getImageInfo(*self*, *imageId*)

Retrieves information about an image from the server given the image id.

Parameters

`imageId`: the id of the image of interest

(type=int)

Return Value

an object that contains the information about the given image if it exists and connects to the server, otherwise None

(type=ImageInfo)

getCroppedImage(*self*, *imageId*)

Retrieves a cropped image of the image from the server given the imageId.

Parameters

`imageId`: the id of the image

(type=int)

Return Value

a tuple of a pillow Image and the image id if the image with that id is cropped, otherwise None

(type=(Image, int))

getNextCroppedImage(*self*)

Retrieves the next available cropped image from the server.

Return Value

a tuple of a pillow Image and the image id if the image with that id is cropped, otherwise None

(type=(Image, int))

getPrevCroppedImage(*self*)

Re-retrieves a cropped image that was previously viewed. The interface maintains an ordered list (of up to numIdsStored) of ids that has previously been viewed and traverses the list backwards.

Return Value

a pillow Image if there are any previous images to process, and the server is able to find the given id, otherwise None.

(type=(Image, int))

getCroppedImageInfo(*self*, *imageId*)

Retrieves information about a cropped image from the server given the image id.

Parameters

`imageId`: the id of the image of interest

(type=int)

Return Value

an object that contains the information about the given cropped image if it exists and connects to the server, otherwise None

(type=CropInfo)

getAllCroppedInfo(*self*)

Retrieves the information pertaining to all of the cropped images in the server.

Return Value

a list of CropInfo objects of all of the cropped images if it connects to the server, otherwise None

(type=CropInfo[])

imageToBytes(*self*, *img*)

Takes an Image object and returns the bytes of the given image.

Parameters

`img:` the image to convert into bytes

(type=Image)

Return Value

bytes of the given image

(type=bytes)

postCroppedImage(*self*, *imageId*, *crop*, *tl*, *br*)

Posts a cropped image to the server.

Parameters

`imageId:` the id to the original image being cropped

(type=integer)

`crop:` the image file of the cropped image

(type=PIL Image)

`tl:` the x and y coordinate of the location of the cropped image in the top left corner relative to the original image

(type=integer array of length 2)

`br:` the x and y coordinate of the location of the cropped image in the bottom right corner relative to the original image

(type=integer array of length 2)

Return Value

The response of the http request if it successfully posts, otherwise None

(type=Response)

getGPSByTs(*self*, *ts*)

Retrieves from the server the GPS measurement that is closest to the given timestamp.

Parameters

`ts:` the timestamp of interest

(type=float)

Return Value

GPSMeasurement object closest to the given timestamp if it connects to the server, otherwise None

(type=GPSMeasurement)

getGPSById(*self*, *gpsId*)

Retrieves from the server a GPS measurement given an id.

Parameters

`gpsId`: the id of the gps measurement of interest

(type=int)

Return Value

GPSMeasurement object of the given Id if it exists and connects to the server, otherwise None

(type=GPSMeasurement)

getStateByTs(*self*, *ts*)

Retrieves from the server the state measurement that is closest to the given timestamp.

Parameters

`ts`: the timestamp of interest

(type=float)

Return Value

StateMeasurement object closest to the given timestamp if it connects to the server, otherwise None

(type=StateMeasurement)

getStateById(*self*, *stateId*)

Retrieves from the server a state measurement given an id.

Parameters

`stateId`: the id of the state measurement of interest

(type=int)

Return Value

StateMeasurement object of the given Id if it exists and connects to the server, otherwise None

(type=StateMeasurement)