ALICA AC Pack Documentation

Release 0.2.0

Marcel Stefko, Kyle M. Douglass

Jun 06, 2018

Contents

CHAPTER 1

Javadoc

1.1 ch.epfl.leb.alica.acpack.analyzers.autolase

1.1.1 AutoLase

public class **`AutoLase`** implements Analyzer

Wrapper for Thomas Pengo's implementation of AutoLase algorithm.

Author Marcel Stefko

Constructors

AutoLase

public **`AutoLase`** (int *threshold*)

Initializes AutoLase with default threshold (120) and averaging (30) values.

Parameters

- **`threshold`** –

Methods

dispose

public void **`dispose`** ( )

getBatchOutput

public double **`getBatchOutput`** ( )

getIntermittentOutput

public double **getIntermittentOutput**()

getName

public String **getName**()

getShortReturnDescription

public String **getShortReturnDescription**()

getStatusPanel

public AnalyzerStatusPanel **getStatusPanel**()

processImage

public void **processImage**(Object *image*, int *image_width*, int *image_height*, double *pixel_size_um*, long *time_ms*)

setROI

public void **setROI**(Roi *roi*)

1.1.2 AutoLaseAnalyzer

class **AutoLaseAnalyzer**

This class estimates the density of activations. The density at a particular point relates to the maximum time a certain pixel is "on", or above a certain threshold. The density is calculated as a moving average 30 frames. The code only works for 2 bytes per pixel cameras for now.

Author Thomas Pengo

Fields

currentDensity

double **currentDensity**

running

boolean **running**

stopping

boolean **stopping**

Constructors

AutoLaseAnalyzer

public **AutoLaseAnalyzer** (int *threshold*)

Methods

getCurrentValue

public double **getCurrentValue** ()
Returns error signal value from AutoLase

Returns estimated averaged max emitter density

getRawCurrentValue

public double **getRawCurrentValue** ()
Returns raw error signal value from AutoLase

Returns estimated max emitter density for most recent frame

nextImage

public void **nextImage** (ShortProcessor *sp*)
Analyzes next image and adjusts internal state.

Parameters

- **image** – image to be analyzed

setParameters

public void **setParameters** (int *threshold*)

setROI

public void **setROI** (Roi *roi*)

1.1.3 AutoLaseSetupPanel

public class **AutoLaseSetupPanel** extends AnalyzerSetupPanel
Setup panel for AutoLase, allows setup of thresholding and averaging.

Author Marcel Stefko

Constructors

AutoLaseSetupPanel

public **AutoLaseSetupPanel**()
Creates new form SetupPanel

Methods

getName

public String **getName**()

initAnalyzer

public Analyzer **initAnalyzer**()

toString

public String **toString**()

1.2 ch.epfl.leb.alica.acpack.analyzers.defcon

1.2.1 DEFCoNSetupPanel

public class **DEFCoNSetupPanel** extends AnalyzerSetupPanel
The DEFCoN analyzer setup panel.

Author Kyle M. Douglass

Constructors

DEFCoNSetupPanel

public **DEFCoNSetupPanel**()
Creates new form SetupPanel

Methods

getName

public String **getName**()

initAnalyzer

public Analyzer **initAnalyzer**()
Initializes the analyzer using the properties from the setup panel.

1.2.2 Defcon

public class **Defcon** implements Analyzer
A fluorescent spot counter derived from the DEFCoN package.

Author Kyle M. Douglass

See also: DEFCoN-ImageJ

Constructors

Defcon

public **Defcon**(String *pathToModel*)
Initializes the DEFCoN analyzer.

Parameters

- **pathToModel** – The path to the DEFCoN network model.

Methods

dispose

public void **dispose**()
Cleans up the analyzer when it's finished.

finalize

protected void **finalize**()
Failsafe in case the predictor has not been closed at the point of garbage collection.

Throws

- **java.lang.Throwable** –

getBatchOutput

public double **getBatchOutput**()
Returns the averaged DEFCoN count value since the last call. Double.NaN is returned if there is no new count since the previous call.

Returns The averaged DEFCoN count.

getBoxSize

public int **getBoxSize**()
Returns the current square kernel size for maximum local counts.

Returns The kernel size for computing the maximum local count.

getIntermittentOutput

public double **getIntermittentOutput**()
Returns the intermittent output of the analyzer.

Returns The analyzer's current output value.

getName

public String **getName**()
Returns the name of the DEFCoN analyzer.

Returns The name of the DEFCoN analyzer.

getShortReturnDescription

public String **getShortReturnDescription**()
Returns a short description of the values returned by the DEFCoN analyzer.

Returns A short description of the values returned by the DEFCoN analyzer.

getStatusPanel

public AnalyzerStatusPanel **getStatusPanel**()
Returns the analyzer's status panel that will be displayed in the GUI. If no panel is implemented, this method should return null. In this case, the corresponding space in the MonitorGUI will appear blank.

Returns The status panel of the DEFCoN analyzer or null.

isLiveModeOn

public boolean **isLiveModeOn**()
True if live mode is on, false otherwise.

Returns True if live mode is on, false otherwise

isMaxLocalCount

public boolean **isMaxLocalCount**()
True if the analyzer is computing the maximum local count.

Returns True if the analyzer is computing the maximum local count.

liveModeOff

public void **liveModeOff**()
Turns off the live view of the density map.

liveModeOn

public void **liveModeOn**()
Turns on the live view of the density map.

maxLocalCountOff

public void **maxLocalCountOff**()
Turns off the live view of the density map.

maxLocalCountOn

public void **maxLocalCountOn**()
Turns on the live view of the density map.

processImage

public void **processImage**(Object *image*, int *width*, int *height*, double *pixelSizeUm*, long *timeMs*)
Processes an image and adjusts the analyzer's internal state to reflect the results of the calculation. This method is called after each new image acquisition by the AnalysisWorker. You can use the synchronized(this) statement within the body of an implementation of an Analyzer to ensure that no output readout happens during code execution.

Parameters

- **image** – The image to be processed as 1D raw pixel data.
- **width** – Image width in pixels.
- **height** – Image height in pixels.
- **pixelSizeUm** – Length of a side of a square pixel in micrometers.
- **timeMs** – Image acquisition time in milliseconds.

setBoxSize

public void **setBoxSize**(int *boxSize*)
Sets the square kernel size for computing the maximum local count.

Parameters

- **boxSize** – The kernel size for computing the maximum local count.

setROI

public void **setROI**(Roi *roi*)

updateLiveView

public void **updateLiveView**()
Updates the live viewer.

1.2.3 DefconStatusPanel

public class **DefconStatusPanel** extends AnalyzerStatusPanel
Status panel of the DEFCoN analyzer.

Author Kyle M. Douglass

Constructors

DefconStatusPanel

public **DefconStatusPanel**(*Defcon defcon*)
Creates new form DefconStatusPanel.

1.2.4 DefconTest

public class **DefconTest**
Unit tests for the Defcon class.

Author Kyle M. Douglass

Methods

setUp

public void **setUp**()

testGetSetBoxSize

public void **testGetSetBoxSize**()
Gets/sets the box size field for the maximum local count.

testMaxLocalCount

public void **testMaxLocalCount**()
Toggles the maximum local count feature.

testNegativeBoxSize

public void **testNegativeBoxSize**()
Ensures that the boxSize value is greater than 1.

testOddBoxSize

public void **testOddBoxSize**()
Ensures that the boxSize value is odd.

testProcessImage

public void **testProcessImage**()
Ensures that the processImage() method is called without errors.

testProcessMaxLocalCount

public void **testProcessMaxLocalCount**()
Analyzer predicts the maximum local count instead of the density map.

1.3 ch.epfl.leb.alica.acpack.analyzers.integrator

1.3.1 Integrator

public class **Integrator** implements Analyzer
Analyzer which outputs the average pixel value per frame. The average is taken over the area of the image (or ROI) in units of squared pixels.

Author Marcel Stefko

Constructors

Integrator

public **Integrator**()

Methods

dispose

public void **dispose**()

getBatchOutput

public double **getBatchOutput**()

getIntermittentOutput

public double **getIntermittentOutput**()

getName

public String **getName**()

getShortReturnDescription

public String **getShortReturnDescription**()

getStatusPanel

public AnalyzerStatusPanel **getStatusPanel**()

processImage

public void **processImage**(Object *image*, int *image_width*, int *image_height*, double *pixel_size_um*, long *time_ms*)

Computes the average of the pixel values taken over the image (or ROI).

Parameters

- **image** –
- **image_width** –
- **image_height** –
- **pixel_size_um** –
- **time_ms** –

setROI

public void **setROI**(Roi *roi*)

1.3.2 IntegratorSetupPanel

public class **IntegratorSetupPanel** extends AnalyzerSetupPanel

Empty panel

Author stefko

Constructors

IntegratorSetupPanel

public **IntegratorSetupPanel**()

Creates new form IntegratorSetupPanel

Methods

getName

public String **getName**()

initAnalyzer

public Analyzer **initAnalyzer**()

1.3.3 IntegratorTest

public class **IntegratorTest**

Author douglass

Methods

setUp

public void **setUp**()

testGetBatchOutput

public void **testGetBatchOutput**()
Test of getBatchOutput method, of class Integrator.

testProcessImage

public void **testProcessImage**()
Test of processImage method, of class Integrator.

1.4 ch.epfl.leb.alica.acpack.analyzers.quickpalm

1.4.1 MyDialogs

class **MyDialogs**

Fields

attach

boolean **attach**

buffer

int **buffer**

cal_z

double **cal_z**

calfile

java.lang.String **calfile**

fwhm

double **fwhm**

height

int **height**

imagedir

java.lang.String **imagedir**

imp

ImagePlus **imp**

imtitle

String **imtitle**

is3d

boolean **is3d**

magn

double **magn**

maxpart

int **maxpart**

maxsize

double **maxsize**

minsize

double **minsize**

model

java.lang.String **model**

models

java.lang.String[] **models**

nimchars

int **nimchars**

nimstart

int **nimstart**

nrois

int **nrois**

nslices

int **nslices**

part_divergence

boolean **part_divergence**

part_extrainfo

boolean **part_extrainfo**

pattern

java.lang.String **pattern**

pixelsize

double **pixelsize**

prefix

java.lang.String **prefix**

prefs

ij.Prefs **prefs**

ptablefile

java.lang.String **ptablefile**

pthrsh

double **pthrsh**

rmanager

RoiManager **rmanager**

rois

Roi[] **rois**

saturation

double **saturation**

smartsnr

boolean **smartsnr**

snr

int **snr**

sufix

java.lang.String **sufix**

symmetry

double **symmetry**

threads

int **threads**

view

boolean **view**

view_mode

java.lang.String **view_mode**

view_modes

java.lang.String[] **view_modes**

viewer_accumulate

int **viewer_accumulate**

viewer_do3d

boolean **viewer_do3d**

viewer_doConvolve

boolean **viewer_doConvolve**

viewer_doMovie

boolean **viewer_doMovie**

viewer_fwhm

double **viewer_fwhm**

viewer_is8bit

boolean **viewer_is8bit**

viewer_mergeabove

double **viewer_mergeabove**

viewer_mergebellow

double **viewer_mergebellow**

viewer_oheight

int **viewer_oheight**

viewer_owidth

int **viewer_owidth**

viewer_tpixelsize

double **viewer_tpixelsize**

viewer_update

int **viewer_update**

viewer_zstep

double **viewer_zstep**

waittime

int **waittime**

width

int **width**

window

int **window**

Methods

analyseParticles

public boolean **analyseParticles** (*MyFunctions f*)

1.4.2 MyFunctions

class **MyFunctions**

Fields

cal3d_center

int **cal3d_center**

cal3d_wmh

double[] **cal3d_wmh**

cal3d_z

double[] **cal3d_z**

caltable

ResultsTable **caltable**

debug

boolean **debug**

dtable

ResultsTable **dtable**

gblur

GaussianBlur **gblur**

live_view

ImagePlus **live_view**

ptable

ResultsTable **ptable**

ptable_lock

java.util.concurrent.locks.Lock **ptable_lock**

Constructors

MyFunctions

public **MyFunctions**(boolean *live_view*)

Methods

argmax

int **argmax**(double[] *arr*)

argmin

int **argmin**(double[] *arr*)

clearRegion

void **clearRegion**(double *thrsh*, ImageProcessor *ip*, boolean[][] *mask*, int *xstart*, int *xend*, int *ystart*, int *yend*)

detectParticles

int **detectParticles**(ImageProcessor *ip*, *MyDialogs* *dg*, int *nframe*)
Particle finding method, will search the image for particles.

Parameters

- **ip** – image to search for particles on
- **dg** – dialog manager
- **nframe** – the frame index corresponding to this image

dispose

public void **dispose**()

getClosest

int **getClosest**(double *value*, double[] *arr*, int *center*)

getMaxPositions

int[] **getMaxPositions**(ImageProcessor *ip*)

getNextImage

ImagePlus **getNextImage**(*MyDialogs* *dg*, int *frame*)

Grabs a new image on an observed folder on the case of the analysis being attached to the acquisition.

Parameters

- **dg** – dialog manager
- **frame** – frame index to search for on the folder

Returns found image

getParticle

boolean **getParticle**(ImageProcessor *ip*, boolean[][] *mask*, int[] *maxs*, *MyDialogs* *dg*, ResultsTable *ptable*, int *nframe*)

Particle analysis method, called for each particle candidate found by detectParticles.

Parameters

- **ip** – image to search for particles on
- **dg** – dialog manager
- **nframe** – the frame index corresponding to this image

getParticleForCalibration

double[] **getParticleForCalibration**(ImageProcessor *ip*, *MyDialogs* *dg*, int *xstart*, int *xend*, int *ystart*, int *yend*)

getZ

double **getZ**(double *wmh*)

Given a calculated width-minus-height (wmh) converts this value into the corresponding coordinate in Z by comparing against the loaded Z-calibration table. Only used it the particle is disturbed by astigmatism.

Parameters

- **wmh** – the width-minus-height of a particle

Returns corresponding z-position value, will return 9999 if wmh is out of limits

initialize3d

void **initialize3d**()
Loads values from the calibration table into the cal3d_* arrays.

log

void **log**(java.lang.String *txt*)

mean

double **mean**(double[] *array*, int *start*, int *stop*)

movingMean

double[] **movingMean**(double[] *arr*, int *window*)

showTable

void **showTable**()

1.4.3 QuickPalm

public class **QuickPalm** implements Analyzer
Produces a localization count per area using QuickPALM.

Author Marcel Stefko

Constructors

QuickPalm

public **QuickPalm**(boolean *live_view*)
Implementation of the QuickPALM algorithm as an analyzer, which produces particle count as output.

Parameters

- **live_view** – if true, live view of particle positions is shown

Methods

dispose

public void **dispose**()

getBatchOutput

public double **getBatchOutput**()

getIntermittentOutput

public double **getIntermittentOutput**()

getName

public String **getName**()

getShortReturnDescription

public String **getShortReturnDescription**()

getStatusPanel

public AnalyzerStatusPanel **getStatusPanel**()

processImage

public void **processImage**(Object *image*, int *image_width*, int *image_height*, double *pixel_size_um*, long *time_ms*)

setROI

public void **setROI**(Roi *roi*)

1.4.4 QuickPalmCore

public class **QuickPalmCore**

Wrapper for QuickPalm ImageJ plugin functionality developed by Ricardo Henriques @ Instituto de Medicina Molecular (PT) / Institut Pasteur (FR).

Author Marcel Stefko

Constructors

QuickPalmCore

public **QuickPalmCore**(boolean *live_view*)

Initializes the core and launches a dialog for parameter setup.

Parameters

- **live_view** – if true, live view of particle positions will be shown

Methods

dispose

public void **dispose**()
Close preview window if opened.

processImage

public int **processImage**(ImageProcessor *ip*, int *frame*)
Counts particles in the image.

Parameters

- **ip** – image to be processed
- **frame** – id of the image

Returns no. of detected particles

1.4.5 QuickPalmSetupPanel

public class **QuickPalmSetupPanel** extends AnalyzerSetupPanel

Author stefko

Constructors

QuickPalmSetupPanel

public **QuickPalmSetupPanel**()
Creates new form QuickPalmSetupPanel

Methods

getName

public String **getName**()

initAnalyzer

public Analyzer **initAnalyzer**()

1.5 ch.epfl.leb.alica.acpack.analyzers.spotcounter

1.5.1 FindLocalMaxima

public class **FindLocalMaxima**
Find local maxima in an Image (or ROI) using the algorithm described in Neubeck and Van Gool. Efficient non-

maximum suppression. Pattern Recognition (2006) vol. 3 pp. 850-855 Jonas Ries brought this to my attention and send me C code implementing one of the described algorithms

Methods

FindMax

public static Polygon **FindMax**(ImageProcessor *iProc*, Roi *roi*, int *n*, int *threshold*, *FilterType filterType*)
Static utility function to find local maxima in an Image

Parameters

- **iProc** –
  - ImageProcessor object in which to look for local maxima
- **roi** –
  - region of interest to which the analysis is constrained
- **n** –
  - minimum distance to other local maximum
- **threshold** –
  - value below which a maximum will be rejected
- **filterType** –
  - Prefilter the image. Either none or Gaussian1_5

Returns Polygon with maxima

noiseFilter

public static Polygon **noiseFilter**(ImageProcessor *iProc*, Polygon *inputPoints*, int *threshold*)

Parameters

- **iProc** –
- **inputPoints** –
- **threshold** –

1.5.2 FindLocalMaxima.FilterType

public enum **FilterType**
Different filters for image preprocessing.

Enum Constants

GAUSSIAN1_5

public static final *FindLocalMaxima.FilterType* **GAUSSIAN1_5**

NONE

public static final *FindLocalMaxima.FilterType* **NONE**
No preprocessing.

1.5.3 SpotCounter

public class **SpotCounter** implements Analyzer

Author stefko

Constructors

SpotCounter

public **SpotCounter**(int *noise_tolerance*, int *box_size*, boolean *live_view*)
Initialize the analyzer

Parameters

- **noise_tolerance** – required height of peak around surroundings
- **box_size** – size of the scanning box in pixels
- **live_view** – if true, live preview is shown

Methods

dispose

public void **dispose**()

getBatchOutput

public double **getBatchOutput**()

getIntermittentOutput

public double **getIntermittentOutput**()

getName

public String **getName**()

getShortReturnDescription

public String **getShortReturnDescription**()

getStatusPanel

public AnalyzerStatusPanel **getStatusPanel**()

processImage

public void **processImage**(Object *image*, int *image_width*, int *image_height*, double *pixel_size_um*, long *time_ms*)

setROI

public void **setROI**(Roi *roi*)

1.5.4 SpotCounterCore

public class **SpotCounterCore**
Core of the SpotCounter algorithm

Author Nico Stuurman

Constructors

SpotCounterCore

public **SpotCounterCore**(int *noiseTolerance*, int *boxSize*, boolean *live_mode*)

Parameters

- **noiseTolerance** – minimum peak value
- **boxSize** – size of scanning box
- **live_mode** – if true, live preview is shown

Methods

analyze

public HashMap<String, Double> **analyze**(ImageProcessor *ip*)
Analyzes the image and returns information about current state.

Parameters

- **ip** –
  - image to be analyzed

Returns ResultsTable which contains information about analysis results.

dispose

public void **dispose**()
Hide live view window if it exists.

getBoxSize

public int **getBoxSize**()

getNoiseTolerance

public int **getNoiseTolerance**()

isLiveModeOn

public boolean **isLiveModeOn**()

Returns true if live mode is on, false otherwise

liveModeOff

public void **liveModeOff**()
Turns off live viewing of SpotCounter analysis.

liveModeOn

public void **liveModeOn**()
Turns on live viewing of SpotCounter analysis.

setParams

public void **setParams**(int *noiseTolerance*, int *boxSize*)
Set new parameters for the analysis

Parameters

- **noiseTolerance** – minimum peak value
- **boxSize** – size of scanning box

setROI

public void **setROI**(Roi *roi*)
Constrain analysis to given ROI.

Parameters

- **roi** – ROI to constrain analysis to

1.5.5 SpotCounterSetupPanel

public class **SpotCounterSetupPanel** extends AnalyzerSetupPanel
Setup panel to initialize the SpotCounter

Author Marcel Stefko

Constructors

SpotCounterSetupPanel

public **SpotCounterSetupPanel**()
Creates new form SetupPanel

Methods

getName

public String **getName**()

initAnalyzer

public Analyzer **initAnalyzer**()

1.5.6 SpotCounterStatusPanel

public class **SpotCounterStatusPanel** extends AnalyzerStatusPanel
Status panel of SpotCounter, enables modification of threshold in real time.

Author Marcel Stefko

Constructors

SpotCounterStatusPanel

public **SpotCounterStatusPanel**(*SpotCounterCore core*)
Creates new form SpotCounterStatusPanel

Parameters

- **core** – SpotCounterCore

1.6 ch.epfl.leb.alica.acpack.controllers.inverter

1.6.1 InvertController

public class **InvertController** implements Controller
Controller which inverts and scales the input using 1/x function. (high input -> low output, low input -> high output)

Author Marcel Stefko

Fields

maximum

protected double **maximum**
Maximal possible output value.

Constructors

InvertController

public **InvertController**(double *maximum*, double *value_at_1_mw*)
Initializes the InvertController

Parameters

- **maximum** – max output value
- **value_at_1_mw** – what is the value of input that you want to cause an output value of 1.0 (scaling constant)

Methods

getCurrentOutput

public double **getCurrentOutput**()

getName

public String **getName**()

getSetpoint

public double **getSetpoint**()

getStatusPanel

public ControllerStatusPanel **getStatusPanel**()

nextValue

public double **nextValue**(double *value*)

setSetpoint

public void **setSetpoint**(double *value*)
Sets the scaling constant, since it basically fulfills the role of setpoint for this controller.

Parameters

- **value** – new scaling constant

1.6.2 InverterSetupPanel

public class **InverterSetupPanel** extends ControllerSetupPanel
Setup panel for the InvertController

Author Marcel Stefko

Constructors

InverterSetupPanel

public **InverterSetupPanel**()
Creates new form InverterSetupPanel

Methods

getName

public String **getName**()

initController

public Controller **initController**(double *max_controller_output*, double *tick_rate_ms*)

1.7 ch.epfl.leb.alica.acpack.controllers.manual

1.7.1 ManualController

public class **ManualController** implements Controller
Manual controller. Output is equal to setpoint value, any input is ignored.

Author Marcel Stefko

Fields

maximum

protected double **maximum**
Maximal possible output value.

setpoint

protected double **setpoint**
Output value

Constructors

ManualController

public **ManualController**(double *maximum*, double *initial_output*)
Initialize with maximal output value

Parameters

- **maximum** – max output value
- **initial_output** – starting value of output

Methods

getCurrentOutput

public double **getCurrentOutput**()

getName

public String **getName**()

getSetpoint

public double **getSetpoint**()

getStatusPanel

public ControllerStatusPanel **getStatusPanel**()

nextValue

public double **nextValue**(double *value*)

setSetpoint

public void **setSetpoint**(double *new_setpoint*)

1.7.2 ManualSetupPanel

public class **ManualSetupPanel** extends ControllerSetupPanel

Author stefko

Constructors

ManualSetupPanel

public **ManualSetupPanel**()
Creates new form ManualSetupPanel

Methods

getName

public String **getName**()

initController

public Controller **initController**(double *max_controller_output*, double *tick_rate_ms*)

1.8 ch.epfl.leb.alica.acpack.controllers.pi

1.8.1 PI_SetupPanel

public class **PI_SetupPanel** extends ControllerSetupPanel

Author stefko

Constructors

PI_SetupPanel

public **PI_SetupPanel**()
Creates new form PI_SetupPanel

Methods

getName

public String **getName**()

initController

public Controller **initController**(double *max_controller_output*, double *tick_rate_ms*)

1.8.2 PI_StatusPanel

public class **PI_StatusPanel** extends ControllerStatusPanel

Author stefko

Constructors

PI_StatusPanel

public **PI_StatusPanel** (*PI_controller controller*)
Creates new form PI_StatusPanel

Methods

setValuesDisplay

public void **setValuesDisplay** (double *P*, double *I*)

1.8.3 PI_controller

public class **PI_controller** implements Controller
Simple implementation of PI controller with output constraining and windup prevention.

Author Marcel Stefko

Fields

I

protected double **I**
Integral component

P

protected double **P**
Proportional component

current_output

protected double **current_output**
Last calculated output of the controller

is_blocked

protected boolean **is_blocked**

Constructors

PI_controller

public **PI_controller**(double *P*, double *I_per_second*, double *max_output*, double *sampling_period_s*)
Initialize the PI controller.

Parameters

- **P** – proportional component
- **I_per_second** – integral component (per second)
- **max_output** – maximal output value
- **sampling_period_s** – controller tick rate in seconds

Methods

block

public void **block**()
Temporarily stops the controller from taking in input, and forces output to be 0.

getCurrentOutput

public double **getCurrentOutput**()

getName

public String **getName**()

getSetpoint

public double **getSetpoint**()

getStatusPanel

public ControllerStatusPanel **getStatusPanel**()

nextValue

public double **nextValue**(double *value*)

setSetpoint

public void **setSetpoint**(double *new_setpoint*)

unblock

public void **unblock**()
Resets integral before unblocking the output

1.9 ch.epfl.leb.alica.acpack.controllers.selftuningpi

1.9.1 SelfTuningController

public class **SelfTuningController** extends *PI_controller*
A self-tuning implementation of the PI controller. It waits for 10 cycles, and over next 20 cycles generates a step pulse and measures response. From this response it estimates the P and I components of the PID controller.

Author Marcel Stefko

Constructors

SelfTuningController

public **SelfTuningController**(double *max_output*, double *sampling_period_s*)
Initialize a new SelfTuningController

Parameters

- **max_output** – max output
- **sampling_period_s** – tick rate of controller in seconds
- **step_height** – how big step pulse should be generated
- **p_factor** – scaling factor for P in tuning
- **i_factor** – scaling factor for I in tuning

Methods

getName

public String **getName**()

getStatusPanel

public ControllerStatusPanel **getStatusPanel**()

nextValue

public double **nextValue**(double *value*)

recalibrate

public final void **recalibrate**(double *step_height*, double *p_factor*, double *i_factor*)

1.9.2 SelfTuningSetupPanel

public class **SelfTuningSetupPanel** extends ControllerSetupPanel

Author stefko

Constructors

SelfTuningSetupPanel

public **SelfTuningSetupPanel**()
Creates new form SelfTuningSetupPanel

Methods

getName

public String **getName**()

initController

public Controller **initController**(double *max_controller_output*, double *tick_rate_ms*)

1.9.3 SelfTuningStatusPanel

public class **SelfTuningStatusPanel** extends ControllerStatusPanel

Author stefko

Constructors

SelfTuningStatusPanel

public **SelfTuningStatusPanel**(*SelfTuningController controller*)
Creates new form SeltTuningStatusPanel

Methods

setCalibrationStatusDisplay

public void **setCalibrationStatusDisplay**(String *text*)

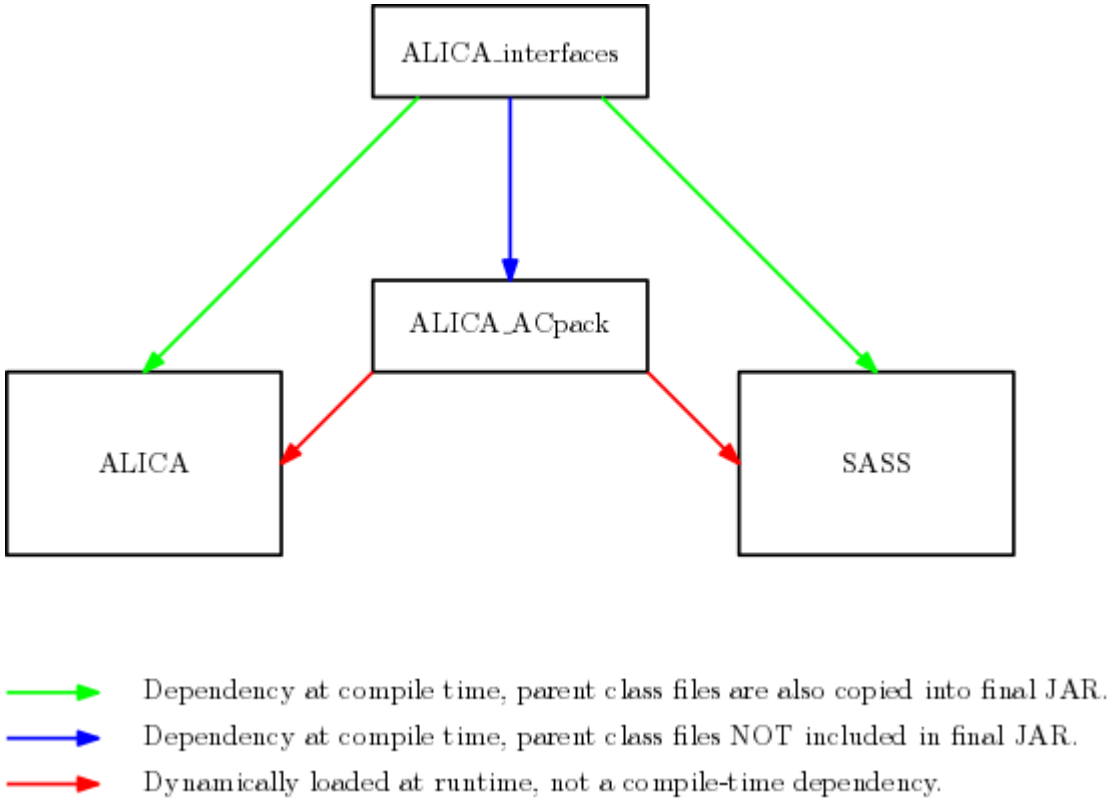
setValuesDisplay

public void **setValuesDisplay**(double *P*, double *I*)

This package contains the implementations for the analyzers and controllers in ALICA, the automatic illumination control package for light microscopy. It is a run-time dependency of ALICA. These implementations are maintained separately of ALICA because they are used in SASS as well.

CHAPTER 2

Relationship with ALICA and SASS



CHAPTER 3

Authors

- Marcel Štefko
- Kyle M. Douglass

CHAPTER 4

See Also

- ALICA - Automated Laser Illumination Control Algorithm
- SASS - SMLM Acquistion Simulation Software

CHAPTER 5

Indices and tables

- genindex

Index

H

I

L

M

N

P

Q

R

S

T

U

V

W