scikit-surgerydavinci Documentation

Thomas Dowrick

Oct 08, 2020

Contents

Augmented reality for image-guided laproscopic surgery.

scikit-surgerydavinci is part of the SNAPPY software project, developed at the Wellcome EPSRC Centre for Interventional and Surgical Sciences, part of University College London (UCL).

Author: Thomas Dowrick

scikit-surgerydavinci supports Python 3.6.

CHAPTER 1

Installing

You can install using pip:

```
pip install scikit-surgerydavinci
```

CHAPTER 2

Example Usage

Single video feed:

```
python sksurgerydavinci -s 0 -m MODEL_DIR
```

-m specifies the location of a directory containing vtl/stl/vtp/ply models.

Stereo video feed:

```
python sksurgerydavinci -s 0 1 -m MODEL_DIR
```

Mock stereo feed using only one input:

```
python sksurgerydavinci -s 0 -1 -m MODEL_DIR
```

If 3 or more screens are available, each video feed will run full screen on a separate display. Video feeds can be assigned to specific displays using the -o arugment:

```
python sksurgerydavinci -s 0 1 -o 2 3 1
```

More details on command line arguments can be viewed using:

```
python sksurgerydavinci -h
```

2.1 Contributing

Please see the contributing guidelines.

2.2 Useful links

- Source code repository

- Documentation

CHAPTER 3

Licensing and copyright

CHAPTER 4

Acknowledgements

Supported by Wellcome and EPSRC.

4.1 stable

4.1.1 sksurgerydavinci package

Subpackages

sksurgerydavinci.ui package

Submodules

sksurgerydavinci.ui.gui module

GUI for AR Davinci

class sksurgerydavinci.ui.gui.**UI**(*overlay_window*)
Bases: PySide2.QtWidgets.QWidget

User interface class, to combine vtk render view with buttons/sliders etc

add_crop_buttons()
Add buttons to crop the video stream.

add_exit_button()
Add a button to exit the program

add_opacity_slider()
Create a QSlider that controls VTK model opactity

add_record_buttons()
Add buttons to take screenshot and record video. Button callbacks should be implmenented in the Viewer class.

add_toggle_stereo_button()
Add button to switch between L and R views

add_vtk_models(*vtk_models*)
Add VTK models to the UI and vtk_overlay_window

Parameters **vtk_models** (*List of VTKSurfaceModels*) – List of vtk models to add

add_vtk_overlay_window(*overlay_window*)
Add the vtk_overlay_window to the UI layout.

Parameters **overlay_window** – vtk_overlay_window

create_toggle_checkbox_for_model(*model*)
Create a checkbox that will toggle on/off model visiblity.

Parameters **model** – VTKSurfaceModel

Returns QCheckBox

exit_signal = <PySide2.QtCore.Signal object>

on_exit_clicked()
Close the application when the exit button has been clicked

set_all_opacities(*opacity_percent*)
Set the opacity for all VTK models

Parameters **opacity_percent** (*int*) – Target opacity value, in %

set_screen(*screen*)
Link the widget to a particular screen

Parameters **screen** – QGuiQpplication.screen() object

staticMetaObject = <PySide2.QtCore.QMetaObject object>

sksurgerydavinci.ui.sksurgerydavinci_command_line module

Command line processing

sksurgerydavinci.ui.sksurgerydavinci_command_line.**main**(*args=None*)
Entry point for ardavin application

sksurgerydavinci.ui.sksurgerydavinci_demo module

Hello world demo module

sksurgerydavinci.ui.sksurgerydavinci_demo.**create_sample_model**()
Create an empty VTKSurfaceModel object, and set the actor to a vtkConeSource

sksurgerydavinci.ui.sksurgerydavinci_demo.**run_demo**(*args*)
Show message

Module contents

scikit-surgerydavinci

sksurgerydavinci.widgets package

Submodules

sksurgerydavinci.widgets.Viewers module

Mono/Stereo viewing windows for Ardavinci

class sksurgerydavinci.widgets.Viewers.**FakeOverlayOnVideoFeed**
Bases: object

Implement empty methods to replicate OverlayOnVideoFeed.

on_record_start()
Intentionally Blank.

on_record_stop()
Intentionally Blank.

set_roi()
Intentionally Blank.

start()
Intentionally Blank.

stop()
Intentionally Blank.

class sksurgerydavinci.widgets.Viewers.**FakeVTKOverlayWindow**
Bases: object

Implement empty methods to replicate VTKOverlayWindow.

add_vtk_models(*models*)
Intentionally Blank.

get_foreground_camera()
Intentionally Blank.

save_scene_to_file(*fname*)
Intentionally Blank.

set_foreground_camera(*camera*)
Intentionally Blank.

set_geometry()
Intentionally Blank.

set_screen()
Intentionally Blank.

class sksurgerydavinci.widgets.Viewers.**MockStereoViewer**(*video_source*)
Bases: *sksurgerydavinci.widgets.Viewers.StereoViewerBase*

Mock stereo viewer, duplicating a single camera input to multiple screens.

Parameters **video_source** – OpenCV compatible video source (int or filename)

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class sksurgerydavinci.widgets.Viewers.**MonoViewer**(*video_source*)
Bases: *sksurgerydavinci.widgets.Viewers.StereoViewerBase*

Generates a VTK interactor UI with a single video stream as background. :param video_source: OpenCV compatible video source (int or filename)

Only use the left_view of StereoViewerBase. Set the other views to non-existent views.

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class sksurgerydavinci.widgets.Viewers.**StereoViewer**(*left_source*, *right_source*)
Bases: *sksurgerydavinci.widgets.Viewers.StereoViewerBase*

Stereo viewer, creates an overlay window for each video input.

Parameters

- **left_source** – OpenCV compatible video source (int or filename)
- **right_source** – OpenCV compatible video source (int or filename)

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class sksurgerydavinci.widgets.Viewers.**StereoViewerBase**
Bases: PySide2.QtWidgets.QWidget

Base class for StereoViewers.

Child classes implment the left/right/ui video feeds as appropriate.

add_vtk_models(*models*)
Add VTK models to all widgets.

Parameters **models** (*VTKSurfaceModel*) – List of models to add.

are_widgets_on_different_screens(*screens*)
Check if each widget has been placed on it's own screen. :param screens: List of screen numbers corresponding to the screen

each widget is displayed on.

maximize_left_and_right_widgets()
Fullscreen view for each widget. QWidget.setFullScreen() isn't working properly in all cases, so resizing maunally instead.

on_autocrop_started()
Start auto cropping.

on_autocrop_stopped()
Stop auto cropping.

on_crop_clicked()
Set the ROI on the left view, and copy it to the right.

on_record_start_clicked()
Start recording a video. In the MockStereoViewer we only need to record the 'left' input, as the right view is a duplicate of the left. The proper StereoViewer class extends this method to also record the right view.

on_record_stop_clicked()
Stop recording data to file and restore button settings.

on_screenshot_clicked()
Save a screenshot to disk, using date and time as filename

run_before_quit()
Clean up the VTK interactor instances before quitting

set_all_roi(*roi*)
Set the roi for left/right views.

set_external_screens(*output_screens*)

Set the display screens for each widget, if

external monitors are available.

Parameters **output_screens** (*List of ints*) – List of screens on which to put widgets

set_widget_screens(*screens_to_use*)
Move each of the widgets to a particular screen. If each widget is on it's own screen, run fullscreen.

Parameters **screens_to_use** – List of QScreen objects.

start()
Start all widgets.

staticMetaObject = <PySide2.QtCore.QMetaObject object>

sufficent_displays_for_stereo_view()
Check if there are enough additional displays to give each window it's own screen.

sync_camera_view_between_windows()
Set all the foreground cameras to the same vtkCamera, so that camera changes are synchronised.

update_autocrop()
Automatically crop the incoming video stream using AutoCropper.

Module contents

Submodules

sksurgerydavinci.davinci_xi_auto_cropping module

AutoCropper Module.

class sksurgerydavinci.davinci_xi_auto_cropping.**AutoCropBlackBorder**(*threshold=1*, *min_size=0*)

Bases: object

Class to generate a ROI for an image based on a threshold value. Originally developed for auto cropping video feed from the DaVinci Xi and Tile Pro, where there is a central image (which we want to keep), surrounded by a black region, and then other stuff that we're not interested in.

By searching for the start/end of the black region, we can auto crop the bit we do want.

For speed, only the red channel of the image is scanned when detecting black areas (See comment in set_roi for more details).

Parameters **threshold** – Threshold value

: param min_size: Minimum size (applies to both width and length) of the expected output.

get_bounds(*vector*)
Given a vector, skip any leading values that are > threshold, then find the first contiguous range of values that are below the threshold value. If the whole vector is above the threshold, return the start/end indexes. e.g. [0 0 0 50 50 50 0 0 0] return 3, 5 - [50 50 50]

[25 25 0 0 25 25 25 0 0] returns 4, 7 [25 25 25] as the leading 25s are skipped.

[100 100 100 100 100 100] returns 0, 5.

Parameters **vector** – Input vector, in which to find the start/end bounds

Return start First element that is above the threshold.

Return end First element following start, that is below threshold.

get_roi(*img*)
Calculate the ROI. Find the x/y extent of the ROI by averaging row and column values, and then finding the area of interest.

Module contents

scikit-surgerydavinci

- modindex
- genindex
- search

Python Module Index

Index

R

S

U