snappysonic Documentation

Stephen Thompson

May 04, 2023

Contents

Author: Stephen Thompson

SnappySonic can be used as an ultrasound acquisition simulator. The output from a tracking system (NDI or AruCo tags) is to select a frame of pre-recorded video to show. A suitable video of ultrasound data is included in the data directory, however the user can select a video of their choosing. The software and its use is described in the SnappySonic paper.

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

SnappySonic supports Python 3.6.

```
pip install snappysonic
python snappysonic.py --config config.json
```

The config file defines the tracking parameters and image buffer, e.g.

```
{
 "ultrasound buffer": "data/usbuffer.mp4",
       "buffer descriptions": [
              {
               "name": "glove",
               "start frame": 0,
               "end frame": 284,
               "x0": 20, "x1": 200,
               "y0": 200, "y1": 260,
               "scan direction": "x"
              },
  ]
  ....
  "tracker config": {
              "tracker type": "aruco",
              "video source": 2,
              "debug": true,
              "capture properties": {
                      "CAP_PROP_FRAME_WIDTH": 640,
                      "CAP_PROP_FRAME_HEIGHT": 480

 }
}
```

An example configuration file can be downloaded from here and an image buffer from source code repository data directory

CHAPTER 1

Developing

1.1 Cloning

You can clone the repository using the following command:

```
git clone https://github.com/SciKit-Surgery/snappysonic
```

1.2 Running tests

Unit tests are performed in stand alone environments using tox, which also checks coding style.

```
tox
```

CHAPTER 2

Installing

You can pip install from pypi with

```
pip install snappysonic
```

or You can pip install directly from the repository as follows:

```
pip install git+https://github.com/SciKit-Surgery/snappysonic
```

CHAPTER 3

How to Cite

If you use this software in your research or teaching, please cite:

Thompson, S., Dowrick, T., Xiao, G., Ramalhinho, J., Robu, M., Ahmad, M., Taylor, D. and Clarkson, M.J., 2020. SnappySonic: An Ultrasound Acquisition Replay Simulator. Journal of Open Research Software, 8(1), p.8. DOI: http://doi.org/10.5334/jors.289

3.1 Contributing

Please see the contributing guidelines.

3.2 Useful links

- Source code repository
- Documentation

CHAPTER 4

Licensing and copyright

CHAPTER 5

Acknowledgements

Supported by Wellcome and EPSRC.

5.1 Requirements for snappysonic

This is the software requirements file for snappysonic, part of the SciKit-Surgery project. The requirements listed below should define what snappysonic does. Each requirement can be matched to a unit test that checks whether the requirement is met.

5.1.1 Requirements

ID	Description	Test
0000	Module has a help page	pylint, see tests/pylint.rc and tox.ini
0001	Functions are documented	pylint, see tests/pylint.rc and tox.ini
0002	Package has a version number	No test yet, handled by git.

5.2 latest

5.2.1 snappysonic package

Subpackages

snappysonic.algorithms package

Submodules

snappysonic.algorithms.algorithms module

Functions for snappysonic

snappysonic.algorithms.algorithms.**check_us_buffer**(*usbuffer*)
Checks that all ultrasound buffer contains all required key values.

Parameters **usbuffer** – the buffer to check

Raises KeyError

Raises ValueError

snappysonic.algorithms.algorithms.**configure_tracker**(*config*)
Configures a scikit-surgery tracker based on the passed config

Parameters **config** – a tracker configuration dictionary

Returns The tracker

Raises KeyError

snappysonic.algorithms.algorithms.**get_bg_image_size**(*config*)
Reads the geometry from a configuration and returns the extents of the buffer

Parameters **config** – a tracker configuration dictionary

Returns the extents of the buffer

snappysonic.algorithms.algorithms.**lookupimage**(*usbuffer*, *pts*)
determines whether a coordinate (pts) lies with an area defined by a usbuffer, and returns an image from the buffer if appropriate

Parameters **usbuffer** – a dictionary containing bounding box information (x0,y0, x1,y1) and image data

Returns True if point in bounding box. Image.

snappysonic.algorithms.algorithms.**numpy_to_qpixmap**(*np_image*)
Converts the input numpy array to a qpixmap

Parameters **np_image** – a numpy array

Returns a qpixmap

Module contents

snappysonic.overlay_widget package

Submodules

snappysonic.overlay_widget.overlay module

Main loop for tracking visualisation

class snappysonic.overlay_widget.overlay.**OverlayApp**(*config*)
Bases: sksurgeryutils.common_overlay_apps.OverlayBaseWidget

Inherits from OverlayBaseWidget, adding code to read in video buffers, and display a frame of data that depends on the position of an external tracking system, e.g. surgeryarucotracker

staticMetaObject = <PySide2.QtCore.QMetaObject object>

update_view()
Update the background renderer with a new frame, move the model and render

snappysonic.overlay_widget.overlay.**circle**(*img*, *center*, *radius*, *color*[, *thickness*[, *lineType*[, *shift*]]]) → img
. @brief Draws a circle. . . The function cv::circle draws a simple or filled circle with a given center and radius. . @param img Image where the circle is drawn. . @param center Center of the circle. . @param radius Radius of the circle. . @param color Circle color. . @param thickness Thickness of the circle outline, if positive. Negative values, like #FILLED, . mean that a filled circle is to be drawn. . @param lineType Type of the circle boundary. See #LineTypes . @param shift Number of fractional bits in the coordinates of the center and in the radius value.

snappysonic.overlay_widget.overlay.**imread**(*filename*[, *flags*]) → retval
. @brief Loads an image from a file. . . @anchor imread . . The function imread loads an image from the specified file and returns it. If the image cannot be . read (because of missing file, improper permissions, unsupported or invalid format), the function . returns an empty matrix (Mat::data==NULL). . . Currently, the following file formats are supported: . . - Windows bitmaps - *.bmp, *.dib (always supported) . - JPEG files - *.jpeg, *.jpg, *.jpe (see the *Note* section) . - JPEG 2000 files - *.jp2 (see the *Note* section) . - Portable Network Graphics - *.png (see the *Note* section) . - WebP - *.webp (see the *Note* section) . - Portable image format - *.pbm, *.pgm, *.ppm *.pxm, *.pnm (always supported) . - PFM files - *.pfm (see the *Note* section) . - Sun rasters - *.sr, *.ras (always supported) . - TIFF files - *.tiff, *.tif (see the *Note* section) . - OpenEXR Image files - *.exr (see the *Note* section) . - Radiance HDR - *.hdr, *.pic (always supported) . - Raster and Vector geospatial data supported by GDAL (see the *Note* section) . . @note . - The function determines the type of an image by the content, not by the file extension. . - In the case of color images, the decoded images will have the channels stored in **B G R** order. . - When using IMREAD_GRAYSCALE, the codec's internal grayscale conversion will be used, if available. . Results may differ to the output of cvtColor() . - On Microsoft Windows* OS and MacOSX*, the codecs shipped with an OpenCV image (libjpeg, . libpng, libtiff, and libjasper) are used by default. So, OpenCV can always read JPEGs, PNGs, . and TIFFs. On MacOSX, there is also an option to use native MacOSX image readers. But beware . that currently these native image loaders give images with different pixel values because of . the color management embedded into MacOSX. . - On Linux*, BSD flavors and other Unix-like open-source operating systems, OpenCV looks for . codecs supplied with an OS image. Install the relevant packages (do not forget the development . files, for example, "libjpeg-dev", in Debian* and Ubuntu*) to get the codec support or turn . on the OPENCV_BUILD_3RDPARTY_LIBS flag in CMake. . - In the case you set *WITH_GDAL* flag to true in CMake and @ref IMREAD_LOAD_GDAL to load the image, . then the [GDAL](http://www.gdal.org) driver will be used in order to decode the image, supporting . the following formats: [Raster](http://www.gdal.org/formats_list.html), . [Vector](http://www.gdal.org/ogr_formats.html). . - If EXIF information is embedded in the image file, the EXIF orientation will be taken into account . and thus the image will be rotated accordingly except if the flags @ref IMREAD_IGNORE_ORIENTATION . or @ref IMREAD_UNCHANGED are passed. . - Use the IMREAD_UNCHANGED flag to keep the floating point values from PFM image. . - By default number of pixels must be less than 2^30. Limit can be set using system . variable OPENCV_IO_MAX_IMAGE_PIXELS . . @param filename Name of file to be loaded. . @param flags Flag that can take values of cv::ImreadModes

snappysonic.overlay_widget.overlay.**putText**(*img*, *text*, *org*, *fontFace*, *fontScale*, *color*[, *thickness*[, *lineType*[, *bottomLeftOrigin*]]]) → img
. @brief Draws a text string. . . The function cv::putText renders the specified text string in the image. Symbols that cannot be rendered . using the specified font are replaced by question marks. See #getTextSize for a text rendering code . example. . . @param img Image. . @param text Text string to be drawn. . @param org Bottom-left corner of the text string in the image. . @param fontFace Font type, see #HersheyFonts. . @param fontScale Font scale factor that is multiplied by the font-specific base size. . @param color Text color. . @param thickness Thickness of the lines used to draw a text. . @param lineType Line type. See #LineTypes . @param bottomLeftOrigin When true, the image data origin is at the bottom-left corner. Otherwise, . it is at the top-left corner.

snappysonic.overlay_widget.overlay.**rectangle**(*img*, *pt1*, *pt2*, *color*[, *thickness*[, *lineType*[, *shift*]]]) → img

. @brief Draws a simple, thick, or filled up-right rectangle. . . The function cv::rectangle draws a rectangle outline or a filled rectangle whose two opposite corners . are pt1 and pt2. . . @param img Image. . @param pt1 Vertex of the rectangle. . @param pt2 Vertex of the rectangle opposite to pt1 . . @param color Rectangle color or brightness (grayscale image). . @param thickness Thickness of lines that make up the rectangle. Negative values, like #FILLED, . mean that the function has to draw a filled rectangle. . @param lineType Type of the line. See #LineTypes . @param shift Number of fractional bits in the point coordinates.

rectangle(img, rec, color[, thickness[, lineType[, shift]]]) -> img . @overload . . use *rec* parameter as alternative specification of the drawn rectangle: *r.tl() and . r.br()-Point(1,1)* are opposite corners

Module contents

snappysonic.ui package

Submodules

snappysonic.ui.snappysonic_command_line module

Command line processing

snappysonic.ui.snappysonic_command_line.**main**(*args=None*)
Entry point for snappysonic application

snappysonic.ui.snappysonic_demo module

SnappySonicdemo module

snappysonic.ui.snappysonic_demo.**run_demo**(*configfile*)
Run the application

Module contents

snappysonic

Module contents

snappysonic

- modindex
- genindex
- search

Python Module Index

Index