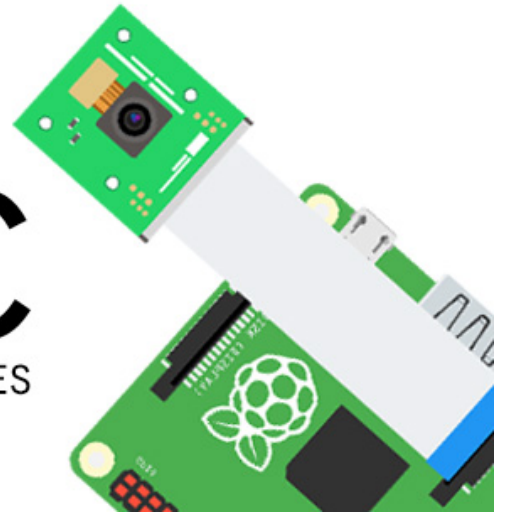


# AnimRec

A python module for automated image and video recording on the RaspberryPi.

ANIMREC  
2015-2018 © JWJOLLES



## Installation

---

To install, open a terminal window and enter:

```
bash pip install git+https://github.com/jolleslab/animrec.git
```

When AnimRec is already installed, make sure to update to the latest version:

```
bash pip install --update git+https://github.com/jolleslab/animrec.git
```

## Dependencies

---

AnimRec depends on [Python 2.7](#) and the [picamera](#) package and makes use of various utility functions of the associated [AnimLab](#) package. AnimRec is created specifically for automated recording with the RaspberryPi, but is adaptable to broader possible instances.

## Overview

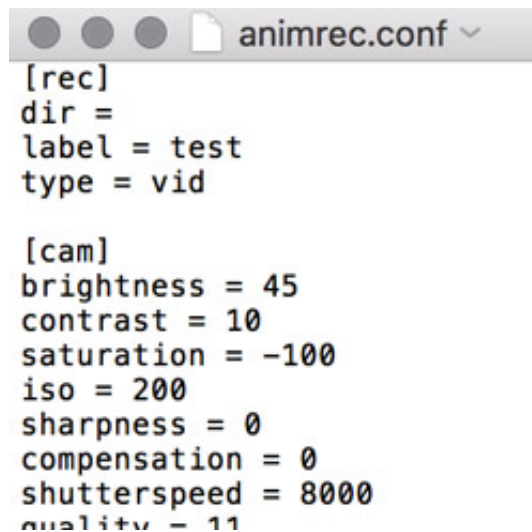
---

*AnimRec* is a python package designed to help facilitate automated recording using the RPi, specifically with easy, customized, repeated image and video recording for scientists in mind. *AnimRec* is (still) currently a private package on [GitHub](#) that can be easily installed from github with the right credentials ([see above](#)).

## Recorder class

The main functionality of *AnimRec* is the `Recorder` class in the `animrec` module. This class initiates a Recorder instance that sets up the pi to record either a single image, a sequence of images, or a loop of videos. AnimRec creates a `[setup]` directory in the users' home directory to store all relevant setup files. In addition AnimRec automatically creates a log file `[animrec.log]` file that stores all output of the terminal while using the module.

AnimRec has a lot of custom settings to facilitate controlled and automated recording. When AnimRec is initiated for the first time a specific configuration file `[animrec.conf]` is created and stored in the setup folder. The settings that can be stored are divided into 1) general user recording parameters, 2) camera settings, specific 3) video and 4) image recording settings, and 5) custom settings that are specific to the rpi. For a detailed overview and description of these settings ([see below](#)).

A screenshot of a text editor window titled 'animrec.conf'. The window shows the following configuration: [rec] dir = label = test type = vid [cam] brightness = 45 contrast = 10 saturation = -100 iso = 200 sharpness = 0 compensation = 0 shutterspeed = 8000 quality = 11. The text is in a monospaced font, and the window has standard macOS-style window controls (red, yellow, green buttons) at the top left.

```
[rec]
dir =
label = test
type = vid

[cam]
brightness = 45
contrast = 10
saturation = -100
iso = 200
sharpness = 0
compensation = 0
shutterspeed = 8000
quality = 11
```

AnimRec is set up in such a way that it is very easy to set and save custom settings that are then automatically used without further user input. The `setup/animrec.conf` file is directly editable (see screenshot above) or alternatively settings can be stored when running the `animrec.set_config()` function.

## Recording modes

AnimRec has three recording modes (with the addition of the `imgtask` function, [explained below](#)): `img`, `imgseq`, and `vid`. Files are automatically stored in the directory set in the custom configuration (`recdir`), by default this is the a NAS drive, and automatically named according to the provided label, the computer name, the date and time, and the session number or image sequence nr (see examples below).

1. `img` mode: This mode simply records a single image with the custom settings and then quits. Example of filename: "pilot\_180312\_PI13\_101300.jpg".
2. `imgseq` mode: This mode is to create a controlled sequence of images based on either a set duration (setting `imgtime`) or total number of images to be recorded (setting `imgnr`) with a certain delay between images (setting `imgwait`). The minimum of `imgnr` and the calculated number of images based on `imgwait` and `imgtime` will be selected. For example, if one wishes to

specifically record 100 images 10.0s after one another, one would use the settings: `imgwait=10` `imgnr=100` and `imgtime=9999`, or if one wishes to record images every 0.5s for 10 hours irrespective of their total number one would use: `imgwait=0.5` `imgnr=999999` `imgtime=36000`. Example of filename: "pilot\_180312\_PI13\_img00231\_101300.jpg".

3. `vid` mode: This mode records a loop of standardized videos based on the custom settings. After each recording has finished, the user is asked if a new recording should be started or the recorder should exit. Specific settings that can be set for this mode are `vidfps`, the framerate of the video, `vidduration`: the duration of the video, and `viddelay`: Extra recording time in seconds that will be added to `vidduration`, used for filming acclimatisation time that is automatically ignored in potential subsequent tracking. Example of filename: "pilot\_180312\_PI13\_S01\_101300.h264".

## Other modules

In addition to the main recording module, AnimRec contains a couple of other modules to aid in setting-up the rpi to have the best standardized recording parameters:

1. `getedge()`: a dynamic function that lets the user draw a rectangle on a live stream of the rpi camera to create the region of interest to be used for recording.
2. `getgains()`: an automatic function that tries to determine the optimal white balance for the current camera position and lighting conditions. This function stores a tuple of rg and bg values that can be further adjusted by the user.
3. `imgtask()`: an add-on module for the `Recorder` class that enables the scheduling of automated image recording tasks, such as to record a sequence of images from 7am > 7pm at 1 image/min every day of the week.

## Workflow

---

The workflow for which AnimRec was designed is as follows:

1. Install the latest version of [Raspbian](#) and make sure it is fully up to date with python installed:  
`sudo apt-get update && sudo apt-get upgrade`.
2. Set up the rpi with an (IR) camera and position it in such a way that it records the zone of interest (using the `raspistill -t 0 -k` command).
3. Install AnimRec [see above](#)
4. Run the `getedge()` function to get the right region of interest to be used for recording.
5. Run the `getgains()` function to get the right, standardized white balance.
6. Run `AnimRec` for the first time to determine the right brightness settings for the camera. Camera brightness depends on the following parameters: `brightness`, `iso`, `contrast`, and `compensation`. Easiest is to record a single image (use `rectype=img`) and adjust these parameters until satisfied, which are then automatically stored.
7. Now the rpi and AnimRec configuration are fully set up, simply use AnimRec with the required custom configuration file (for some examples, [see below](#)).

# Using AnimRec

---

## Python script

The most straight forward way is to use write a simple python script (e.g. `rpirec.py`) to run AnimRec, containing the following code:

```
import animrec

# Initiate the recorder instance
AR = animrec.Recorder()

# Further store some new settings
AR.set_config(label = "test", rectype = "vid", saturation = -100)

# Run record function
while True:
    AR.record()
```

To now run this script, you simply type in terminal: `python rpirec.py`

**Note:** Make sure that the provided parameters conform to the datatype and are within the possible range, see [settings](#) below, as otherwise the script will result in an error.

## Alias

To make running AnimRec even easier we can create an alias for our recording script with a custom command. For this we need to open the `.bashrc` file in our root directory:

```
sudo nano ~/.bashrc
```

and add the following to the bottom of the file:

```
alias rec='sudo rpirec.py'
```

Now all you need to enter in terminal to start Animrec is `rec`, and AnimRec automatically starts with your custom settings.

## Jupyter

A nice alternative is to make use of [jupyter notebook](#). This is an open-source web application that allows you to create python scripts (among many other coding languages) that contain live code, equations, and visualizations that can be executed on a cell-by-cell basis. Jupyter is a great way to sequentially run parts of your code and problem solve it. To install, type in: `python -m pip install jupyter`. To start

jupyter, type in: `jupyter notebook` .

## Settings

---

### Parameters

-----

`recdir : str, default = "NAS"`

The directory where media will be stored. Default is "NAS", which is the automatically mounted NAS drive. If different, a folder with name corresponding to location will be created inside the home directory.

If no name is provided (""), the files are stored in the home directory.

`setupdir : str, default = "setup"`

The directory where setup files are stored relative to home directory. Best to keep this except for very rare instances.

`Label : str, default = "test"`

Label for associating with the recording and stored in the filenames.

`rectype : ["img", "imgseq", "vid"], default = "img"`

Recording type, either a single image, a sequence of images, or a video.

### Config settings

-----

`rotation : int, default = 0`

Custom rotation specific to the RPi, should be either 0 or 180.

`brighttune : int, default = 0`

A rpi specific brightness compensation factor to standardize light levels across multiple rpi's, an integer between -10 and 10.

`gains : tuple, default = (1.0, 2.5)`

Custom gains specific to the RPi to have a 'normal' colorspace.

`brightness : int, default = 45`

The brightness level of the camera, an integer value between 0 and 100.

`contrast : int, default = 20`

The image contrast, an integer value between 0 and 100.

`saturation : int, default -100`

The color saturation level of the image, an integer value between -100 and 100.

`iso : int, default = 200`

The camera ISO value, an integer value in sequence [200,400,800,1600].

Higher values are more light sensitive but have higher gain.

`sharpness : int, default = 50`

The sharpness of the camera, an integer value between -100 and 100.

`compensation : int, default = 0`

Camera lighting compensation. Ranges between 0 and 20. Compensation artificially adds extra light to the image.

`shutterspeed : int, default = 10000`

Shutter speed of the camera in microseconds, i.e. the default of 10000 is equivalent to 1/100th of a second. A longer shutterspeed will result

in a brighter image but more motion blur. Important: the framerate of the camera will be adjusted based on the shutterspeed. At shutter-speeds above ~ 0.2s this results in increasingly longer waiting times between images so a standard imgwait time should be chosen that is 6+ times more than the shutterspeed. For example, for a shutterspeed of 300000 imgwait should be > 1.8s.

quality : int, default = 11

Specifies the quality that the encoder should attempt to maintain. Valid values are between 10 and 40, where 10 is extremely high quality, and 40 is extremely low.

imgdims : tuple, default = (3280,2464)

The resolution of the images to be taken in pixels. The default is the max resolution that does not return an error for this mode.

viddims : tuple, default = (1640,1232)

The resolution of the videos to be taken in pixels. The default is the max resolution that does not return an error for this mode.

imgfps : int, default = 1

The framerate for recording images. Will be set automatically based on the imgwait setting so should not be set by user.

vidfps : int, default = 24

The framerate for recording video.

imgwait : float, default = 1.0

The delay between subsequent images in seconds. When a delay is provided that is less than ~0.5s (shutterspeed + processingtime) it will be automatically set to 0 and images thus taken immediately one after the other.

imgnr : int, default = 60

The number of images that should be taken. When this number is reached, the script will automatically terminate.

imgtime : integer, default = 60

The time in seconds during which images should be taken. The minimum of a) imgnr and b) nr of images based on imgwait and imgtime will be selected.

vidduration : int, default = 10

Duration of video recording in seconds.

viddelay : int, default = 0

Extra recording time in seconds that will be added to vidduration. Its use is for filming acclimatisation time that can then easily be cropped for tracking.

## Output

-----

Either one or multiple .h264 or .jpg files depending on the filetype and single input. All files are automatically named according to the label, the host name, date, time and potentially session number or count nr, e.g.

- single image: 'pilot\_180312\_PI13\_101300.jpg

- multiple images: 'pilot\_180312\_PI13\_img00231\_101300.jpg

- video: 'pilot\_180312\_PI13\_S01\_101300.h264

## Returns

-----

self : class

Recorder class instance