Pose-Trigger

Release v1.0.0

Keisuke Sehara, Paul Zimmer-Harwood

Dec 27, 2020

CONTENTS:

Pose-Trigger is a python application for real-time, closed-loop application of TTL trigger generation based on the pose of the subject.

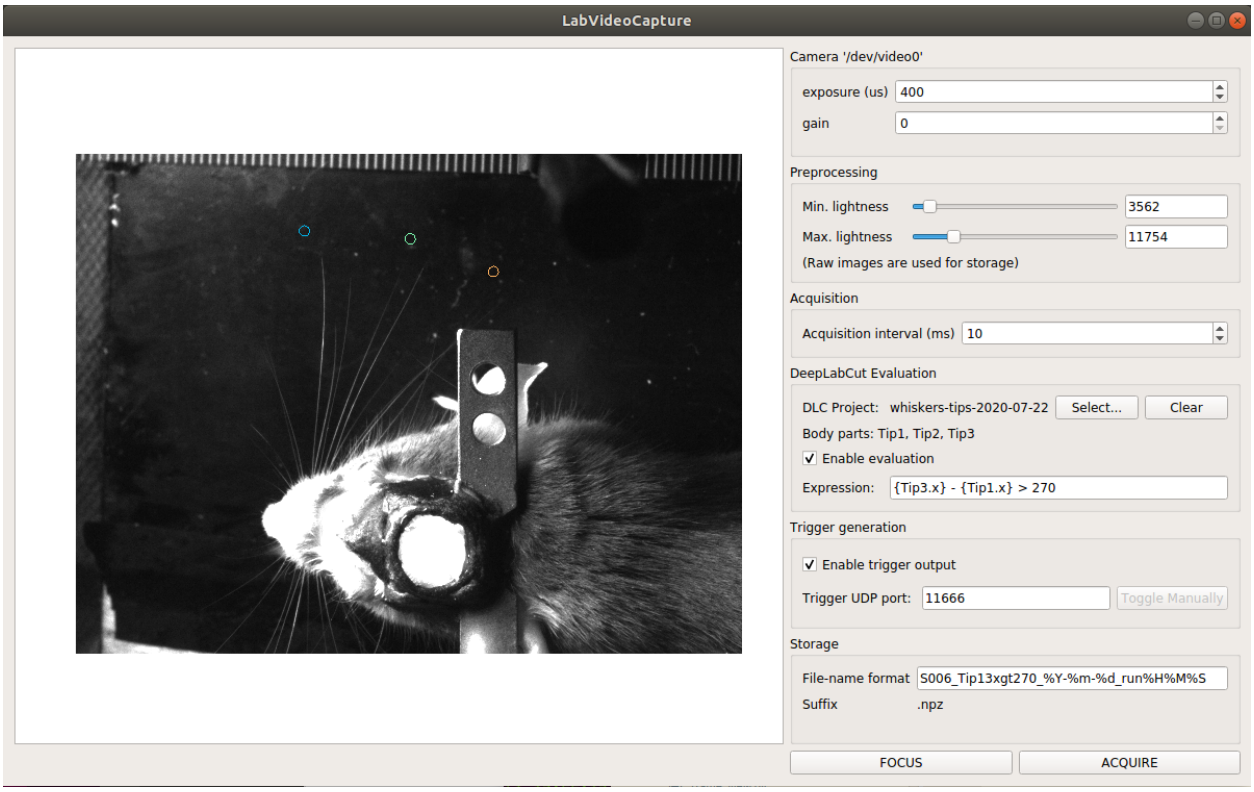

Fig. 1: A screenshot of a working Pose-Trigger app (development version)

CHAPTER

ONE

INTRODUCTION

Contents

1.1 What Pose-Trigger can do

Pose-Trigger is designed to work on a linux computer equipped with a high-speed video camera.

The current version of the software features:

- Acquisition of **high-speed videos** (up to 100-200 fps without on-line pose estimation).
- On-line exposure/gain adjustment.
- Adjustment of acquisition intervals.
- **On-line estimation of body-part positions** using DeepLabCut.
  - On-line evaluation of **arbitrary posture conditions** based on the estimated body-part positions.
  - **Fast output-trigger generation** (<1 ms) using the FastEventServer program.
- **Brightness/contrast adjustment** for on-line display.
- **Storage of frames** into the NumPy-style zip archive.

1.2 How Pose-Trigger works

Pose-Trigger is essentially a Python application. You can install Pose-Trigger on a Linux computer, and run from Terminal by typing:

```
$ pose-trigger
```

(The `$` character represents a prompt. You are not supposed to type it)

1.2.1 The model setup

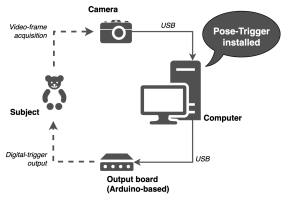

Fig. 1: The model setup

Above is the model setup that uses Pose-Trigger. Pose-Trigger is designed to work in a closed-loop experiment setup, where a single PC acquires video frames from the camera and generates trigger output based on the behavior of the subject.

For more detailed system requirements, refer to the *System requirements* section.

1.2.2 The main acquisition loop

Below is the schematics for the main acquisition loop:

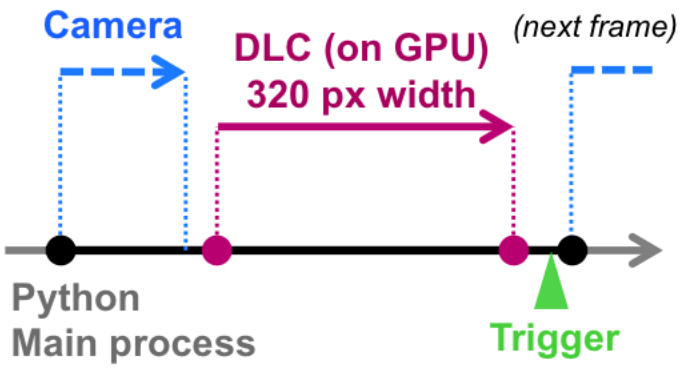

Fig. 2: The main acquisition loop

1. The **timer** generates timings for the acquisition of the next frame (black filled circles).
2. Pose-Trigger commands the camera to acquire a video frame, and receive it (blue dashed arrow).
3. Pose-Trigger delegates body-part estimation to the underlying DeepLabCut library (in case it exists; magenta arrow).
4. Pose-Trigger updates the status of trigger output by sending information to FastEventServer (in case it is serving; green arrowhead).

CHAPTER

TWO

INSTALLATION

Contents

2.1 Installation

We recommend installing everything through Anaconda3. Make sure your PC satisfies the *requirements*.

Find the environment file `posetrigger.yaml` from the repository, and run the following command in the terminal:

```
$ conda env create -f posetrigger.yaml; conda activate posetrigger
```

You can change the name of the environment by giving the alternative as `conda env create -n <name> -f posetrigger.yaml`.

Note: Upon the public release of Pose-Trigger in the future, `timedcapture`, `dlclib` and `pose-trigger` packages will be made available in PyPI.

2.2 System requirements

Before installing Pose-Trigger, make sure you have set up the following hardware:

- **A linux computer** (tested on Ubuntu 18.04 LTS)
- **a 16-bit monochrome video camera** from ImagingSource (e.g. refer to the *Reference setup specifications*).

> **Note:** Other Video4Linux2-compliant cameras should also work with a few adjustments in the code, but will require some efforts.

- For a faster working of DeepLabCut, **NVIDIA graphics board with a large amount of RAM** is required.

> **Note:** For example, running DeepLabCut on ResNet-50 requires ~10.6 GB of RAM, so we use GeForce RTX 2080 Ti that has 11 GB on-board RAM (refer to the *Reference setup specifications*).

2.2.1 Requirements for trigger-output generation

In addition to the pose-estimation feature, the trigger-output feature requires the followings:

- The FastEventServer **server program**.
- An Arduino UNO or its clone, being flashed with the arduino-fasteventtrigger program.

For installation of the softwares, refer to the README file in the "libraries" directory of the repository.

> **Caution:** `arduino-fasteventtrigger`, in reality, will **only make use of the serial-to-USB conversion tip on the UNO (i.e. `ATmega16U2` _)**. This means:
>
> - Make sure that your UNO clone has the ATmega16U2 as its converter chip.
> - Other USB-based boards that uses the ATmega16U2 chip *may* work (not recommended nor supported).

2.3 Reference setup specifications

We develop and test Pose-Trigger in the following environment:

2.3.1 Hardware

Table 1: Reference setup hardware specifications

Part name	Model type
CPU	3.7 GHz Core i7-9700K
RAM	64 GB DDR4-3200
GPU	NVIDIA GeForce RTX 2080 Ti (11 GB RAM)
Camera	ImagingSource DMK 37BUX287
Output board	Arduino UNO, rev. 2 (clone), with arduino-fasteventtrigger

2.3.2 Software

Table 2: Reference setup software environment

Software	Specification
Operating system	Ubuntu 18.04 LTS
Python environment	Anaconda3, Python 3.7.7
CUDA Toolkit	version 10.1 (through *conda*)
Tensorflow	version 1.13.1 (*tensorflow-gpu* package of *conda*)
DeepLabCut	version 2.1.3
NumPy	version 1.19.1 (through *conda*)

CHAPTER

THREE

QUICK USAGE GUIDE

Contents

3.1 Launching Pose-Trigger

1. Open Terminal.
2. Run the following command on Terminal:

```
$ pose-trigger
```

Note: When being run without a parameter, Pose-Trigger will use the device at `/dev/video0` by default. In case you want to use e.g. `/dev/video1`, specify the device as the parameter, i.e. run `pose-trigger /dev/video1`.

3.2 Organization of the main window

The Pose-Trigger main window can be divided into three groups:

- The **Capture** buttons (yellow) are for starting/stopping acquisition.
- The **Preview** panel (green) provides an on-line preview of the acquired video frames. If estimation of body-part positions is activated (refer to *DeepLabCut evaluation*), estimated positions will be shown as colored circles, too.
- In the **Settings** panel (blue), you can configure how acquisition is performed (refer to the *Panel-by-panel guide*).

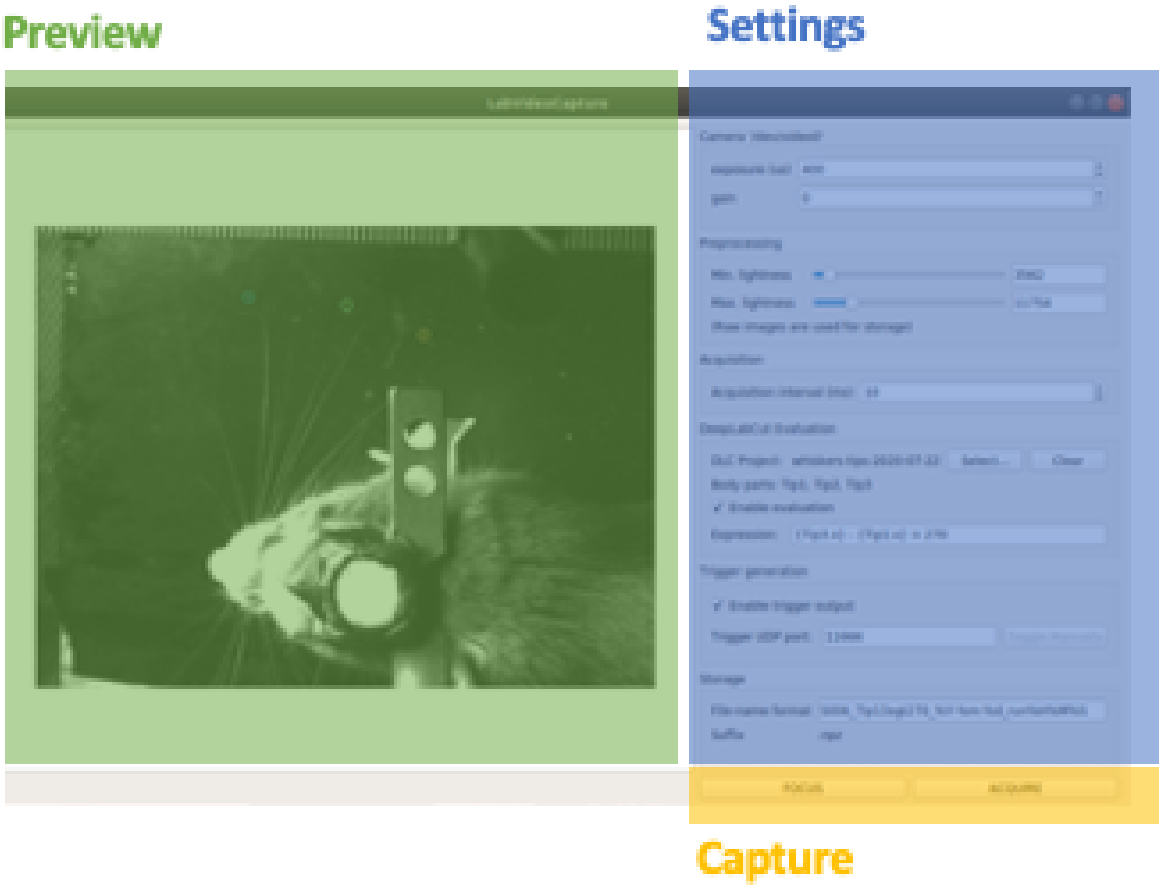

Fig. 1: Overview of the main window

3.3 Capturing videos

3.3.1 Capture modes

There are two modes of running for Pose-Trigger:

- **FOCUS mode**: capturing video frames without storing them
- **ACQUIRE mode**: captures video frames *and* stores acquired data

You can start/stop either of the capturing modes by clicking on the button at the bottom of the main window.

Caution: Pose-Trigger does !not! stream data into storage during acquisition! During acquisition, it keeps all the data in-memory. The data will be written out to a file only *after* acquisition. The duration of acquisition will be thus limited to the order of 1–2 minutes.

Note: Currently, the following parameters are "hard-coded" and used as default:

- Image format: 640x480 pixels, 16-bit grayscale
- Timing generation: a busy-wait algorithm
- Storage format: the NumPy zip-file format (.npz)

3.3.2 Format of the saved files

The data are saved in the NumPy zip-file format (i.e. ".npz" file). Each file includes the following entries:

Table 1: Entries in saved files

Name	Always there?	Description
`frames`	Yes	the 3-D frame data, (frame-index, height, width)
`timestamps`	Yes	1-D array containing unix timestamps in seconds
`metadata`	Yes	a JSON-serialized text object containing information on acquisition configuration
`estimation`	No (Optional)	when a DeepLabCut project is selected; 3-D array with the (frame-index, parameter) shape
`trigger_status`	No (Optional)	when pose-evaluation is enabled; 1-D boolean array of evaluation results

TODO

add some examples for metadata (and probably for other entries, too)

CHAPTER

FOUR

PANEL-BY-PANEL GUIDE

Contents

4.1 "Camera" panel

Fig. 1: "Camera" panel for capture-parameter settings

Here, you can set the exposure and the gain of each video frame acquisition.

Note: For the time being, **the image format is restricted to 16-bit grayscale, with the 640x480 frame size** (otherwise there will be an unexpected behavior).

4.2 "Preprocessing" panel

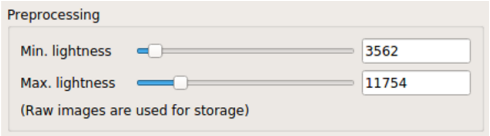

Fig. 2: "Preprocessing" settings

This controls the brightness/contrast settings for "live" video frames. It controls signal conditioning parameters for:

- Video-frame preview
- The images being fed to DeepLabCut (i.e. body-part position estimation)

On the other hand, **the raw, unconditioned images are used** for data storage.

4.3 "Acquisition" panel

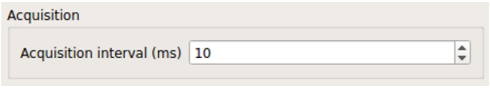

Fig. 3: "Acquisition" timing control

Here you can set the (targeted) acquisition intervals. For example, if you want to have Pose-Trigger running at 50 Hz, set this interval to 20 ms.

Note: For the time being, you can only choose to use the busy-wait timing-generation mechanism. This means that the *minimum* inter-frame interval is set to the value specified here.

4.4 "DeepLabCut evaluation" panel

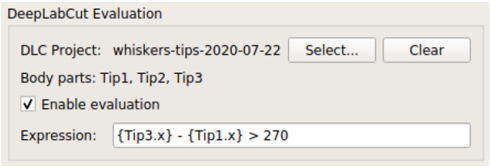

Fig. 4: "Evaluation" mode control

Here, you can configure how DeepLabCut should work in real-time.

4.4.1 Project selection

By using the "Select" button, you can select your DeepLabCut project of choice. Conversely, by clicking on the "Clear" button, you can un-set the project.

When a project is selected, the panel shows the body-part labels being registered in the project.

In addition, as long as a project is selected here, body-part position estimation occurs during video-capture processes. Estimated positions will also be stored in the data file in the case of the *ACQUIRE mode*.

4.4.2 Pose evaluation

You can enable pose evaluation by ticking the "Enable evaluation" button. Evaluation occurs using **the boolean expression entered in the "Expression" field**. The "expression" can be any Python one-line expression, but it has to be evaluated to be a boolean.

When specifying the boolean expression, you can use a **placeholder-based reference** to body part positions. For example, by entering `{Tip1.x}`, you can use the X coordinate of `Tip1` as a parameter. Other than the `x` property, you can also use the `y` and `p` properties of a body part to refer to the Y coordinate and the probability.

In computation of the expression, some major libraries can be used: use `math` for representing the `math` standard library, and use `np` to refer to the `numpy` library. For example, the expression below calculates the Euclidean distance between two body parts, `Tip1` and `Tip2`:

```
math.sqrt( ({Tip1.x} - {Tip2.x})**2 + ({Tip1.y} - {Tip2.y})**2 )
```

In addition, to enable testing of the output latency at the trigger-generation step, the custom placeholder, `{EVERY10}` is there. By using the following expression, you can toggle trigger output on and off every 10 frames:

```
{EVERY10}.get()
```

4.5 "Trigger generation" panel

Fig. 5: "Trigger" mode control

Here, you can test and control trigger generation.

4.5.1 Enabling communication with FastEventServer

By ticking "Enable trigger output", it starts sending the result of evaluation (true/false value) to FastEventServer. Receiving the result, FastEventServer, in turn, sends command to the Arduino-based output board to generate the corresponding output.

4.5.2 Manually toggling the trigger

When trigger-output based on evaluation results is disabled, you can manually toggle the trigger output on and off, using the "Toggle manually" button.

Caution: For the time being, the "trigger UDP port" cannot be specified; if Pose-Trigger fails to connect to FastEventServer on port 11666 at the beginning of its running session, it disables the trigger-output functionality during the whole running session.

4.6 "Storage" panel

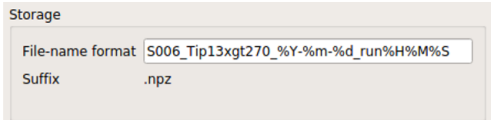

Fig. 6: "Storage" control

Here, you can control how acquired data are stored.

File names are automatically generated using the text entered in the "File-name format" field.

You can use the following **format directives**. These fields are passed on straight to the `datetime.strftime` method (refer to the python datetime module documentation on the specific format directives).

Caution: Be cautious of Pose-Trigger **automatically overwriting existing files**! Try to include (at least) the minutes/seconds directive into the file-name format, so that you do not unexpectedly delete your previous videos.

To the extent possible under law, the authors has waived all copyright and related or neighboring rights to Pose-Trigger documentation (under the legal code CC0 1.0).