Chicken Map

A 2D coordinate mapping program for monitoring the location of chickens.

Table of Contents

Key knowledge

Windows

- To open a command prompt, press the Windows key or click the Start Menu, type *cmd*, and press enter.
- To execute a command, type the command and press Enter.
- To paste into a command prompt, right-click.
- To re-run a previous command, navigate between them using the up and down arrow keys, then press Enter.

MacOS

- To open a Terminal, press Cmd + space, type *terminal*, and press Return/Enter.
- To execute a command, type the command and press Return/Enter.
- To paste into a Terminal, right-click.
- To re-run a previous command, navigate between them using the up and down arrow keys, then press Enter.

Prerequisites

Windows

Tesseract 5.x. (tested with 5.3.1). Download the latest here for Windows and install.

MacOS

Tesseract 5.x (tested with 5.3.2). First, install homebrew if not already installed:

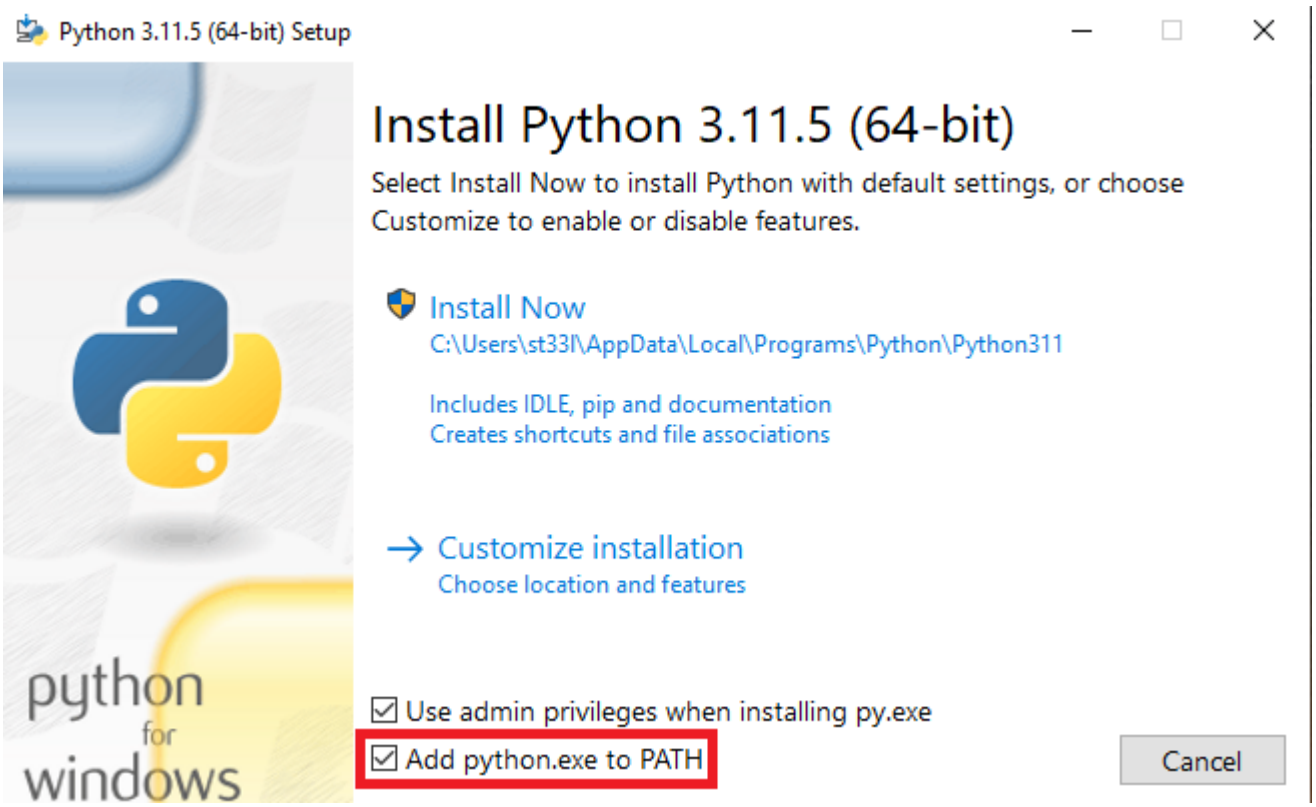
```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

and then install Tesseract:

```
brew install tesseract -y
```

Both

Python 3.x (tested with 3.8, 3.11). Download the latest for your system here. On Windows, when installing, make sure to check the *Add python.exe to PATH* box, then click *Install Now*.



- At the end, you have the option to *Disable path length limit*. While not necessary for this program, it's a good idea to click that option.
- After installation, verify that Python3 and pip3 are installed in a command prompt/Terminal:

```
py --version
pip3 --version
```

If both respond with a version number, you're good to go.

Installation

Download a zip of this code, then extract. Required Python libraries for this program:

- OpenCV-Python
- Python-tesseract
- Pillow
- openpyxl

These can be installed automatically by opening a command prompt/Terminal, navigating to the folder, and using `pip` to read the requirements file:

Windows

```
cd %USERPROFILE%\Downloads\ChickenMap-main\ChickenMap-main
pip3 install -r requirements.txt
```

MacOS

```
cd ~/Downloads/ChickenMap-main
pip install -r requirements.txt
```

How To Use

- Press `q` to quit the program. Clicking the X (Windows) or red dot (MacOS) will just replace the video with another window.
- Left-click anywhere to produce a coordinate at your cursor.

    - Coordinates are saved along with their video timestamps (from the top-left corner of the video) in an Excel file in the `sheets/` directory. You can find this .xlsx file in the `ChickenMap-main/` folder. Filenames are based on your system's date and time when the program started.
    - Coordinates remain on screen for 5 seconds after click by default. Press `c` while a coordinate is on-screen to clear it from the screen and remove it from the Excel sheet. Once the coordinate is off-screen, the coordinate cannot be cleared from the Excel sheet.
    - Coordinates and timestamps are printed to the Command Prompt/Terminal window as a backup and are not removed when `c` is pressed.
- Right-click to annotate at your cursor.
    - The video will freeze/pause. While a flashing typing cursor will not appear on screen, each key you press will, at the location you right-clicked.
        - Press `Enter` to save the annotated image and resume the video.
        - Press `Esc` to cancel annotating and resume the video.
        - Annotations will stay on screen for 5 seconds by default.
    - Annotated images are saved in the `annotated_images/<timestamp>` directory, where `<timestamp>` is the system date/time when you ran the program. You can find these .jpg files in the `ChickenMap-main/` folder. Filenames are based on the timestamp in the top-left corner of the video; annotations at the same timestamp are given a `_#` suffix to prevent overwriting.

Usage

See Examples for platform-specific instructions (what you should actually type into the command line). `VIDEO_PATH` is a required argument; arguments encapsulated by brackets are optional.

```
py chickenMap.py VIDEO_PATH [out_dir] [anno_dir] [exit_key] \
    [clear_key] [duration]
```

You can view command-line options by typing:

```
py chickenMap.py -h
```

Short Argument	Long Argument	Description	Default
-od	--out_dir	Name of output folder for Excel files	sheets/
-ad	--anno_dir	Name of output folder for annotated images	annotated_images/
-e	--exit_key	Key to quit program (a-z, 0-9)	q
-c	--clear_key	Key to remove coordinate from screen and Excel file (a-z, 0-9)	c
-d	--duration	Duration of coordinates on screen, in seconds	5

Full options are available in the `options.json5` file. You can open this file with Notepad (Windows) or TextEdit (MacOS), or your favorite text editor, if you have one. Make sure to save the file after you change options. Any option not entered at the command line will default to the one stored in this file. Here, you can also edit font, font color, font scale, and font thickness. See the comments in the file for limitations.

You can change all the settings you want in `options.json5` and just type `py chickenMap.py VIDEO_PATH` into the command line, and the program will use the settings you entered into `options.json5`. Options entered at command line are saved to `options.json5` so you don't have to retype them each time.

Windows

In a new command prompt:

```
cd %USERPROFILE%\Downloads\ChickenMap-main\ChickenMap-main
py chickenMap.py VIDEO_PATH
```

`VIDEO_PATH` should not be typed out; it should be the filename of the video you want to play. You can drag a video file from File Explorer into the command prompt window and press enter to run

the program; this makes it easy if your video is stored on an external hard drive. Just make sure to add a space after `py chickenMap.py` before dragging a video file into the window.

MacOS

In a new Terminal:

```
cd ~/Downloads/ChickenMap-main
python chickenMap.py VIDEO_PATH
```

`VIDEO_PATH` should not be typed out; it should be the filename of the video you want to play. You can drag a video file from Finder into the Terminal window and press enter to run the program; this makes it easy if your video is stored on an external hard drive. Just make sure to add a space after `python chickenMap.py` before dragging a video file into the window.

Examples

Basic, uses the options in `options.json5`:

```
py chickenMap.py test.mp4
```

Set the exit key to `Esc` and the duration of on-screen coordinates and annotations to 2 seconds, with the other options being filled in from `options.json5`:

```
py chickenMap.py test.mp4 -e Esc -d 2
```

Compatibility

Tested with:

- Devices and Platforms
    - Windows
        - AM4 PC running Windows 10 Pro 22H2 (build 19045)
        - Samsung laptop running Windows 11 Home 22H2 (build 22621)

        - 2015 MacBook Pro (Intel) running Windows 10 Home via Boot Camp
    - MacOS
        - 2020 MacBook Air (M1) running MacOS 13
        - 2015 MacBook Pro (Intel) running MacOS 12.6.3
        - AM4 PC running MacOS 12.6 via OpenCore
    - *Linux probably works since MacOS works, but I offer no detailed instructions for it (you run Linux — you can figure it out).*
- Software
    - Tesseract-OCR 5.12, 5.11
    - Python 3.11, 3.9
        - OpenCV-Python 4.8.0.74
        - Python-tesseract 0.3.10
        - Pillow 10.0.0
        - openpyxl 3.1.2

Privacy

This program does not store or transmit any user data to an external source and can run without connection to the Internet. Your OS/platform (Windows, MacOS) is determined at runtime to point `pytesseract` to the Tesseract-OCR executable on Windows, and it is not stored after the program exits. Program errors and platform info (OS version, Python version, processor name) are stored in `error_log.txt` and are not transmitted by this program; if you have errors, see Support.

Support

For non-guaranteed support, email me at logan.orians@gmail.com or message me on Discord. Please attach `error_log.txt` to your message and describe what you were doing when the error occurred.

Development

Decisions

Q: Why did you make the mouse callback function for OpenCV a function in an object/class? Couldn't you have just defined the mouse callback in `main()`?

A: Sure, but I would still need global variables for coordinates and annotations, as `cv2.setMouseCallback()` doesn't allow the callback function to return anything. I wanted to avoid using global variables (where possible) because it was getting messy, so I needed coordinate and annotation classes. By putting those objects with the mouse_callback function in a class, I could significantly reduce the number of global variables and local/global namespace conflicts I tended to miss when debugging.

Q: Why didn't you use a proper UI library or toolkit?

A: I don't have experience with UIs in Python, but I have experience with OpenCV. OpenCV was able to do what I needed.

Style and Formatting

This code attempts to follow both PEP 8 and the Google Python Style Guide, with programmer's freedom on conflicting elements, and line width set to 100, not 80, characters because it's 2023 and we have high-resolution and ultrawide monitors.

Tools Used

- Sublime Text 4, Notepad++ and VSCode for text editing and programming
- MarkText for README editing

License