Chicken Map

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

Version 2023.11.2

Table of Contents

Command Line 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.
- If you have any issues with commands not being recognized after installing Python, Tesseract, or packages using `pip3`, close all command prompt windows and open a new one.

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.
- If you have any issues with commands not being recognized after installing Python, Tesseract, or packages using `pip3`, close all Terminals and open a new one.

macOS Gatekeeper Override

For macOS users, running `.command` files downloaded from the Internet will likely trigger the Gatekeeper and won't let you run the command. This happens because I am unable to digitally sign the software without a certificate from a Certificate Authority, and this can be an annoyingly lengthy and not free process. Anyway, Apple provides an override tutorial. Basically:

- Open System Preferences (settings)
- Nagivate to Privacy and Security
- Click *Open Anyway* next to the `.command` file in the Security section.

Short Installation Instructions

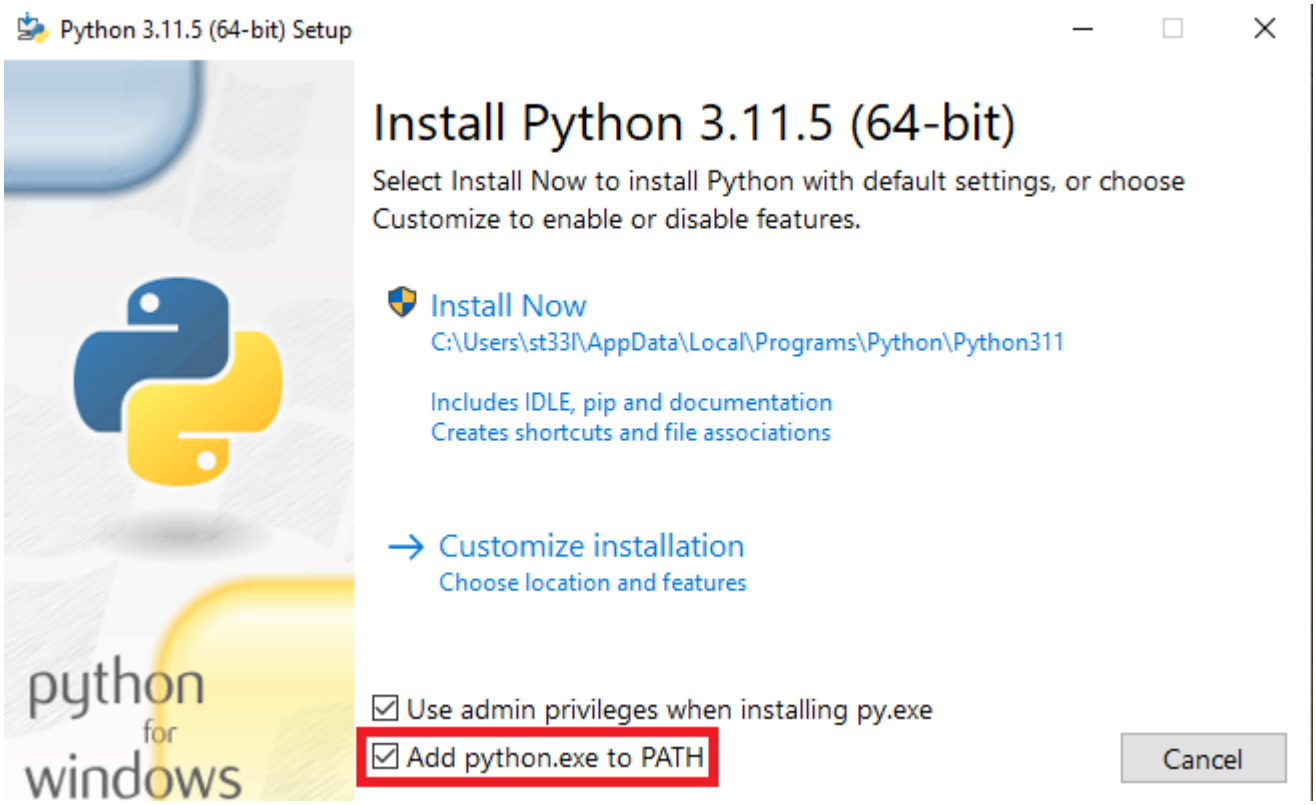
Please continue with the sections below for full instructions.

- Install Python 3.7+ and Tesseract 5.x
- Download zip of code
- Double-click `REQS_WIN.cmd` or `REQS_MAC.command`

Prerequisites

Windows

- Tesseract 5.x. (tested with 5.3.1). Download the latest here for Windows and install.
- Python 3.7+ (latest recommended). Download the latest for your system here and install. If you have Python installed already and want to see if your currently installed version is sufficient (i.e., >=3.7), type `py --version` in a command prompt.



- When installing, make sure to check the *Add python.exe to PATH* box, then click *Install Now*. At the end, you'll have the option to *Disable path length limit*. While not necessary for this program, it's a good idea to click that option.

macOS

- Tesseract 5.x (tested with 5.3.2).
  - First, install Homebrew by typing the below command into Terminal. If you already have Homebrew installed, entering `which brew` will show you the install location, and you can skip to the next step:

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

  - Then install Tesseract:

    ```
    brew install tesseract
    ```

- Python 3.7+ (latest recommended). Download the latest for your system here and install.

Required Packages

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

- NumPy
- OpenCV-Python
- openpyxl
- Python-tesseract
- Pillow
- sv-ttk (optional, but it makes the options GUI look better)

These can be installed by double-clicking the `REQS_WIN.cmd` file on Windows or `REQS_MAC.command` on macOS. macOS users will likely be prompted with a security pop-up; follow these instructions.

Once the script finishes installing the packages, if you see "INSTALLATION COMPLETE. YOU MAY NOW CLOSE THIS WINDOW" toward the bottom, you are all set to run the program. If you don't see this message or you see an error message (in red on macOS), please contact the author.

How To Use

Command line usage can be found under Usage, but you shouldn't need to use a command prompt or Terminal anymore! To test things out, run the test video by double-clicking the `chicken_map.py` file. If this opens in a text editor instead of running the program, double-click `chicken_map_WIN.cmd` on Windows or `chicken_map_MAC.command` on macOS. macOS users will likely encounter a security warning; bypass this by following these instructions.

After testing out the program with the instructions below, double-click the `options_gui.py` file to edit program options, including choosing the video file to be played. A detailed list of program options can be found here. As before, if this opens in a text editor, double-click the `options_gui.cmd` file on Windows or `options_gui.command` file on macOS.

chicken_map Instructions

- Press `q` to quit the program. Clicking the X in the corner (on Windows) will just replace the video with another window.

- Press `p` to pause the video. Press `p` again to resume. The video is automatically paused while annotating but will resume once `Enter` is pressed, unless you pressed `p` beforehand.
- 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. Each key you press will show up on screen, at the location you right-clicked.
        - Press `Enter` to save the annotated image and resume the video.
        - Press `Esc` to cancel annotating. If the video was not manually paused before, the video will resume.
        - Press `Backspace` just as you would normally to remove letters from the annotation.
        - Annotations will stay on screen for 5 seconds by default.
    - Annotated images are saved as `.jpg` files in the `annotated_images/<timestamp>` directory, where `<timestamp>` is the system date/time when you ran the program. 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

```
chickenMap.py [-h] [-o]
```

You can set program options via a GUI with:

```
chickenMap.py -o
```

Please do not edit the `.options.json` file directly (if you see it).

options_gui

Change the options as you see fit. There is a font preview at the bottom to show how your selected font options will look in `chicken_map`. Press Save to save your options. Press Close to close the program. Press Defaults to reset the options to default.

Input video file: Use the file dialog window to choose the chicken video

3D?: If you want to get 3D coordinates, check the box and select *Aviary* or *Floor*

Spreadsheet folder: the folder containing the Excel sheets of timestamped coordinates

Annotated images folder: the folder containing annotated frame captures

Exit key: the key to exit `chicken_map.py`

Clear key: the key to clear the on-screen coordinate from the Excel sheet

Coordinate duration: the length, in seconds, that coordinates will stay on screen after clicking and annotations will stay after pressing Enter

Font: the font for coordinates and annotations

Font color: 16.7 million color combinations for the font

Font scale: how big the font is

Font thickness: how thick the font is

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 13 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 3.12, 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 for `chicken_map` and to determine the system theme (light or dark) for `options_gui`. Program errors and platform info (OS version, Python version, processor name) are stored in `error_log.txt` and are not transmitted for telemetry or error reporting; if you have errors, see Support.

Development

Third-Party Resources

- NumPy
- OpenCV-Python
- openpyxl
- Python-tesseract
- Pillow
- Sun Valley theme by rdbende

Tools Used

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

Style and Formatting

This code attempts to follow PEP 8 and the Google Python Style Guide for style and formatting, with programmer's freedom on any conflicting elements. Line width is set to 100 characters (not 80) because it's 2023 and we have high-resolution and ultrawide monitors.

Type Hints

This code follows PEP 484 + PEP 604 for type hints (for function calls only) to lend some static typing to the program. mypy was used for type checking. Please note that most of the type hint formatting follows conventions supported in Python 3.9+, and 3.10+ for union types.

Decisions (Nerd Questions)

Q: Why did you change the mouse callback function for OpenCV so much?

A: *Stop stalking my commits* All the tutorials suck and just have you use a global variable if you need to get something like `x` and `y` from the callback (you can't grab the return value of a

callback function). Nothing explicitly wrong with globals, I just like to avoid them when I can so I don't risk interfering with something unexpected. You can set the callback function to a class method, define it in main() as a nested function, or actually read the documentation and notice that the `param` argument might as well be nearly purpose-built for this, and yet no one uses it. Just pass in an *object* for `param` and set a value in the callback and bam, problem solved.

Q: Why did you make *x* a `SimpleNamespace` instead of a `dict`?

A: I really like the dot notation for accessing object attributes and often find myself trying to use it on dicts. If I don't need to do anything fancy with key-value pairs, why not make it easier on myself? Plus it looks cleaner, in my opinion.

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

A: I didn't have experience with modern UIs in Python, but I had experience with tkinter/Tcl/Tk and OpenCV. OpenCV and tkinter (with a good theme) were able to do what I needed. Feel free to rewrite using PyQt5, DearPyGUI, PySimpleGUI, or other modern cross-platform framework and fork/pull request. Not sure what else exists for OpenCV, but go nuts.

Q: Why did you use a third-party theme for the GUI instead of just the built-in ones?

A: Honestly, I didn't want to spend the time to make a dark version of the default theme for Windows (and yes, a dark mode was totally necessary). macOS's default `aqua` theme handled it automatically without me telling it to, but Windows would have just taken too long to get right (unless I'm missing something obvious). It's only a few lines of code to determine the system theme, so adding a theme where I can just tell it "light" or "dark" was far less of a headache. And the theme looks better, in my opinion.

Q: Why did you make certain files and folders hidden?

A: If it's harder for the user to find, it's harder for the user to mess with.

Support

For support, email me at logan.orians@gmail.com with "chicken map" in the subject line, or message me on Discord and I will get back to you as soon as possible. Please attach `error_log.txt` to your message (and copy+paste or screenshot+attach any errors present in Command Prompt/Terminal) and describe what you were doing when the error occurred.

License