**PenguinCounter User Guide for PointBlue**

Tim DeBenedictis ([tdebenedictis-RA@pointblue.org](mailto:tdebenedictis-RA@pointblue.org))

Version 1.0 • 21 Feb 2023

PenguinCounter is a program that refines YOLO’s raw penguin predictions and visualizes them on a scaled-down version of an orthomosaic. It generates statistics on the sizes of penguins predicted by YOLO, and compares the predictions against human-validated penguin labels. Finally, it generates a miniaturized version of an input orthomosaic.

Terminology:

* **Orthomosaic:** a single image of an entire penguin colony stitched together from many individual drone images, usually hundreds of thousands of pixels on a side.
* **Tile:** a small subsection of an orthomosaic, usually 512x256 pixels. Tiles may overlap by 20 pixels!
* **Prediction:** a penguin found by YOLO within a tile. Has a **class** (adult, adult-stand, or chick), a **probability** (likelihood of belonging to that class), and a **bounding box** with a center position (x,y) and size (w,h).
* **Validation Label:** a penguin labelled by a human within a tile. Has same properties as a Prediction, but probability is 1.0.
* **Validated Tile:** a tile that has been inspected by a human, and contains zero or more validation labels.

PenguinCounter is written in C++. It runs on Linux or MacOS, and can probably be compiled for Windows, too. These instructions describe how to compile and run PenguinCounter.

**Running**

These instructions assume you have an orthomosaic on your local machine, as well as directories containing YOLO predictions and validation labels. All inputs to PenguinCounter are supplied as command line arguments, in the following order:

ortho index adults stands chicks validations val\_map raw\_pred\_map ref\_pred\_map small\_ortho

Here is a description of each argument.

* ortho input path to full-size orthomosaic in GeoTIFF format
* index input path to tile index in CSV format
* adults input path to directory containing adult penguin predictions in TOLO .txt format
* stands input path to directory containing adult standing penguin predictions in TOLO .txt format
* chicks input path to directory containing penguin chick predictions in TOLO .txt format
* validations input path to directory containing human-validated labels in YOLO .txt or .csv format
* val\_map output path to validations map image
* raw\_pred\_map output path to raw predictions map image
* ref\_pred\_map output path to refined predictions map image
* small\_ortho output path to small version of input orthomosaic

The following example assumes you are running on a Mac or Linux host.

* You have an orthomosaic **croz\_20211127.tif** located in a directory **orthos/2021-11-27**
* The tile index file **croz\_20211127\_tilesGeorefTable.csv** is located in **tiles/2021-11-27**
* Your YOLO predictions are located in **counts/2021-11-27/adults\_s2\_best, adult\_stand\_s5\_best,** and **chick\_s\_best**
* Your validation data **croz\_20211127\_validation\_labels.csv** located in **counts/2021-11-27**
* You want the output images **validations.png, raw\_predictions.png**, **refined\_predictions.png**, and **croz\_20211127\_small.jpg** to be saved in a directory **output/2021-11-27**

Here is the command to run PenguinCounter to generate counts and output images from those inputs. Yes, this is all one line!

./counter orthos/2021-11-27/croz\_20211127.tif \

tiles/2021-11-27/croz\_20211127\_tilesGeorefTable.csv \

counts/2021-11-27/adult\_s2\_best \

counts/2021-11-27/adult\_stand\_s5\_best \  
counts/2021-11-27/chick\_s\_best \

counts/2021-11-27/croz\_20211127\_validation\_labels.csv \

output/2021-11-27/validations.png \

output/2021-11-27/raw-predictions.png \

output/2021-11-27/refinded-predictions.png \

output/2021-11-27/croz\_20211127\_small.jpg

An example of successful output looks like this:

[TBD]

The output maps are scaled-down versions of the original orthomosaic. Colored pixels in the output maps represent individual penguins:

* **Red** pixels are adult penguins
* **Green** pixels are adult\_stand penguins
* **Blue** pixels are chick penguins

The final output image is a small (1/32 scale) version of the original orthomosaic. Each pixel in the scaled-down version is the average color of the corresponding 32x32 pixel area in the full-size orthomosaic.

**Compiling**

[TBD]

A few constants are hard-coded in main.cpp. Their names, descriptions, and current values are as follows.

|  |  |  |
| --- | --- | --- |
| **Constant** | **Description** | **Value** |
| kTileWidth | tile width in pixels | 512 |
| kTileHeight | tile height in pixels | 256 |
| kTileOverlap | tile overlap in pixels | 20 |
| kOutputScale | scale of output images relative to input ortho | 1/32.0 |
| kMinProbability | minimum probability of YOLO predictions to accept | 0.01 |

You may wish to change them and recompile the program if (for example) a different tile size is used in future YOLO analysis.

**Under the Hood**

Here is what PenguinCounter does.

1. Reads metadata (width, height, geotransform) from the input orthomosaic, but does not read raster pixel data.
2. Reads tile index and allocates storage for all tiles generated from the orthomosaic.
3. Reads YOLO predictions of adult, adult-standing, and chick penguins.
4. Reads validation data.
5. Counts validation labels of adult, adult\_stand, and chick penguins.
6. Counta predictions of adult, adult stand, and chick penguins in validated tiles
7. Converts all penguin positions (both predictions and validations) from tile (local) to ortho (global) coordinates.
8. Writes output maps of validation labels and raw predictions.
9. Get statistics on the positions and sizes of adult, adult stand, and chick penguins predicted by YOLO, and validated by humans.
10. Deletes YOLO penguin predictions with very low probabilities below kMinProbability as defined above.
11. Deletes YOLO penguin predictions larger or smaller than the largest/smallest validation labels for each class.
12. Deletes YOLO penguin predictions and human-validated labels duplicated across overlapping tile edges.
13. Counts refined (de-duplicated) predictions both in validated tiles, and in all tiles.
14. Writes refined (de-duplicated) penguin prediction map.
15. Generates confusion matrix and classification matrix.
16. Writes scaled-down (small) version of original input orthomosaic.

For de-duplication, the program looks for penguins which overlap each other. Any penguin predictions whose bounding-box center falls inside the bounding box of another penguin is considered to be a duplicate. In this case, the lower-probability penguin is deleted, and the higher-probability penguin survives.