Alternative titles:

**Detecting small objects and many of them: Computer vision and deep learning enables high-throughput counting of collembola in petri dish samples**

**In an imperfect world: Automating collembola counting with deep learning in messy data**

Authors:

Hjalte M. R. Mann1, Mónica André, Jamie Alison, Toke T. Høye, James J. Scott-Fordsmand

Author affiliations:

1: Department of Ecoscience, Aarhus University, C.F. Møllers Allé 4-8, 8000 Aarhus C, Denmark

Corresponding author contact information:

Hjalte Mads Rosenstand Mann  
E-mail: [mann@ecos.au.dk](mailto:mann@ecos.au.dk)  
Phone: +45 31178585

ORCID IDs:

HMRM: 0000-0002-4768-4767, MA: 0000-0000-0000-0000, JJSF: 0000-0002-2260-1224, JA: 0000-0000-0000-0000 TTH: 0000-0001-5387-3284

Keywords: Collembola, computer vision, deep learning, messy data, slicing, linear optimization

Article type: Research article

Word count: 0

# Abstract

Computer vision methods show huge potential.

Real-world data is messy.

Human error is an inevitable fact.

Ground truth is not universal truth.

Ecooxicology risk assessments rely on standardized laboratory processes.

These involve manual counting of collembola in petri dish samples.

Recent advances in computer vision and deep learning methods enable image-based automation of such a task.

# Introduction

* Why do we need to count collembola in petri dishes
* What is the problem with the established method -Developments in computer vision
* How cancomputer vision help solve our problem
* What are we presenting

Ecotoxiological laboratory studies on soil invertebrates deliver important assessments of compound toxicities.

Counting of collembola in petri dish samples in ecotoxicology experiments requires substantial manual labor by trained technicians. The process is time-consuming while the result is a simple census of the present individual collembola. Here, we investigated the feasibility of automating collembola counting based on a single image of a sample taken with a handheld camera and a deep learning object detection model.

Manual live counts are problematic because of movement of individuals and risk over missing individuals or double-counting. Counting based on images can alleviate this problem. Images also allow validation of counts.

In order to ease the manual counting of animals in an image, it is often done by taking an image of the sample and performing the counting in image processing software. This allows for marking of counted individual to prevent double-counting and missed individuals and validation of result.

One approach has been semi-automated background removal to create a binary blob image ideally containing only animals and all the animals in a dish. This has allowed for subsequent automated counting of blobs and thus individuals. However, this process still requires substantial manual effort to clean the image and separate individuals to produce reliable results. Ideally, counting would be entirely automated with just a single RGB image as input. The basis of this work was the output of manual processing of images following a standard protocol.

Deep learning object detection models have gained significant attention for their ability to locate specific objects of interests in images. However, reliable detection of very small and very large numbers of objects remains a challenging task. In this regard, the present task is particularly challenging owing to the very small size of objects, low distinguishability of animals from background noise, large variation in number of objects in an image, visual variation between samples, and variation in the conditions in which the image was taken, how the image was taken, and the camera and image resolution that was used.

* Wider implications
  + Deviations from protocol
  + Variation in images
  + Human error
  + Messy data

We publish the code together with the trained model to facilitate others to implement automated analysis of ecotoxicology samples.

# Material and methods

* Overview of method
* Translating binary to bounding box format for model training
* Splitting image/annotation data into training, validation, and testing sets
* Standard method vs slicing method
* Slicing images/annotations for full resolution training
* Training and testing neural network with standard approach and slicing approach

The method involved several steps: Translating binary filter images to bounding box format; splitting images into training, validation and testing sets; slicing images and associated annotations to a standard size in order to not loose information from large images because of downscaling; training and testing the neural network with a sliding window approach.

## Raw data

Description of raw data.

## Generating training and testing data

Training and testing object detection models requires annotated training and testing data. Specifically, images containing the objects of interest together with associated annotation data containing coordinates for bounding boxes delimiting the objects are required. Here, generation of annotation data was done through automated processing of existing outputs from human analysis. The human analysis results in a binary mask output of each petri dish image with black blobs delimiting animals. To convert these to bounding boxes, we applied automated contour detection and bounding box creation using Python OpenCV. The contour detection produces some errors when there are many objects very close to each other. Although the error rate seems low, for this test, images with a very high number of objects were discarded. Images from four tests were used (Test A-D). Only one image of each sample was used (i.e. for A1.1, A1.2, etc., only one image was included).

Contour detection and bounding box derivation was performed for a total of 127 images resulting in a total of 110.553 objects detected and annotated. See table x for distribution of images and objects between groups.

Group Images Objects A 44 66056 B 39 17479 C 16 17059 D 28 9959

To decrease risk of objects not contained in bounding box, we expanded the widths and heights of boxes with 10%.

The binary masks included blobs with very low areas.

## Splitting data into training, validation, and testing

The images varied substantially in the number of annotated objects they contained. For each experimental group, we split the images into training, validation, and testing at a ratio of 70/15/15 at the level of images while optimizing for a ratio of 70/15/15 at the level of objects. If 70% of the images resulted in an unequal remainder or a decimal value, we rounded to a number that would ensure an equal remainder. This was done so that we could ensure an equal split between validation and testing in terms of number of images. We assigned images to groups by treating the problem as a linear programming problem with the number of images assigned to each group constrained to the 70/15/15 split while the distribution of images to groups were optimized towards lowest deviation from the 70/15/15 split at the level of objects assigned to each group.

Splitting images+annotations into training, validation, testing, with linear optimization (at image level but on condition of number of annotations in each set

| Set | Images | Objects |
| --- | --- | --- |
| Training | 89 | 77305 |
| Validation | 19 | 17790 |
| Testing | 19 | 15458 |

## Slicing images and annotations

Images had substantial variation in size, with the smallest image included being XxX and the largest XxX. We sliced to 640 x 640. Images were only sliced on dimensions larger than 640, i.e., an image with size 100x1280 would be sliced to two image of 100x640.

As the suggested method will produce overlap between sliced images when full 640x640 slices cannot be produced, individual animals will sometimes be present in more than one slice.

Slicing produced the following distribution of sliced images and objects:

| Set | Images | Objects |
| --- | --- | --- |
| Train | 4166 | 93557 |
| Validation | 849 | 21795 |
| Test | 846 | 19127 |

## Training and testing neural network with sliding window approach

Standard vs slice+SAHI approach.

Standard YOLOv5m (640 x 640 image input)

Versus sliced training data and Slicing Aided Hyper Inference (SAHI) at inference. This method slices each image during inference in a sliding window manner and performs inference on each slice. The method is particularly relevant for detection of small objects. Training length was set to 2000 epochs with early stopping with a patience of 100 epochs.

Confidence threshold for prediction:

Non-maximum suppression IOU threshold set to 0.45.

Bounding box evaluation IOU threshold:

## Evaluating detection performance

Since YOLOv5+SAHI has a different output format than YOLOv5 on its own, customs scripts were created to 1) translate YOLOv5 output pickle files to standard YOLOv5 text files and 2) calculate detection performance variables (precision and recall) for both methods. Precision is the number of correct predictions out of all predictions made. Recall is the proportion of the true number of objects that were detected by the model. F1 is the weighted average of precision and recall. A value of 0.2 was used as the IOU threshold for whether bounding boxes overlapped sufficiently to be considered correct.

## Resolution per petri dish

To investigate the influence of image resolution on detection performance, we measured the number of pixels within the limits of the petri dish in each image.

# Pixels per object

We calculated the number of pixel of each object to see if the model recall was dependent on object area.

We must assume that small area sizes come with an increased risk of false negatives but also of human error in the annotation process.

# Results

* Overview of original number of images, lowest resolution, highest resolution,
* Translating binary filters to bounding box format o lowest number of annotations, highest number of annotations
* Slicing images and annotations to avoid downscaling and leverage full resolution information
* Resulting number of images (mean number of annotations, sd?)
* Training and testing model
* Prediction accuracy

### Standard approach

The model trained on the standard images reached best validation accuracy at epoch xxx.

### Slicing

The model trained on the sliced images reached best validation accuracy at epoch xxx.

Running inference on image slices with SAHI.

| Metric | Standard | Slicing |
| --- | --- | --- |
| Precision | 0.89 | 0.84 |
| Recall | 0.09 | 0.90 |
| F1 |  |  |

Conf: 0.1

IOU: 0.2

# Discussion

* What did we present
* Who can use it – and how
* Model ready to test
* Code available for training on new data and adaption
* High resolution images taken in a standardized setting for training and production likely to improve results and should be considered in future studies.

The improved precision with increased image dimensions and with the slicing approach underline that a main challenge of the task of detecting collembola is the very small object sizes relative to the full image.

Human error. If the accuracy of these ground truth annotations is in fact low, it will have detrimental effects on model accuracy.

Variation in accuracy between test images. The overall prediction performance was high

Importantly, we underline that the error associated with small object areas can be down to human error in the annotation process.

# Acknowledgements

xxx

# Author contributions

HRMR, JJSF and TTH conceived the idea. HMRM conceived and developed the method with contributions from JJSF and TTH. HMRM wrote all code. HMRM wrote the manuscript with contributions from JJSF and JJSF. All authors gave final approval for publication.

# Data availability

The code and data that supports the results in this paper is openly available at <https://github.com/TECOLOGYxyz/springtail>.

# References