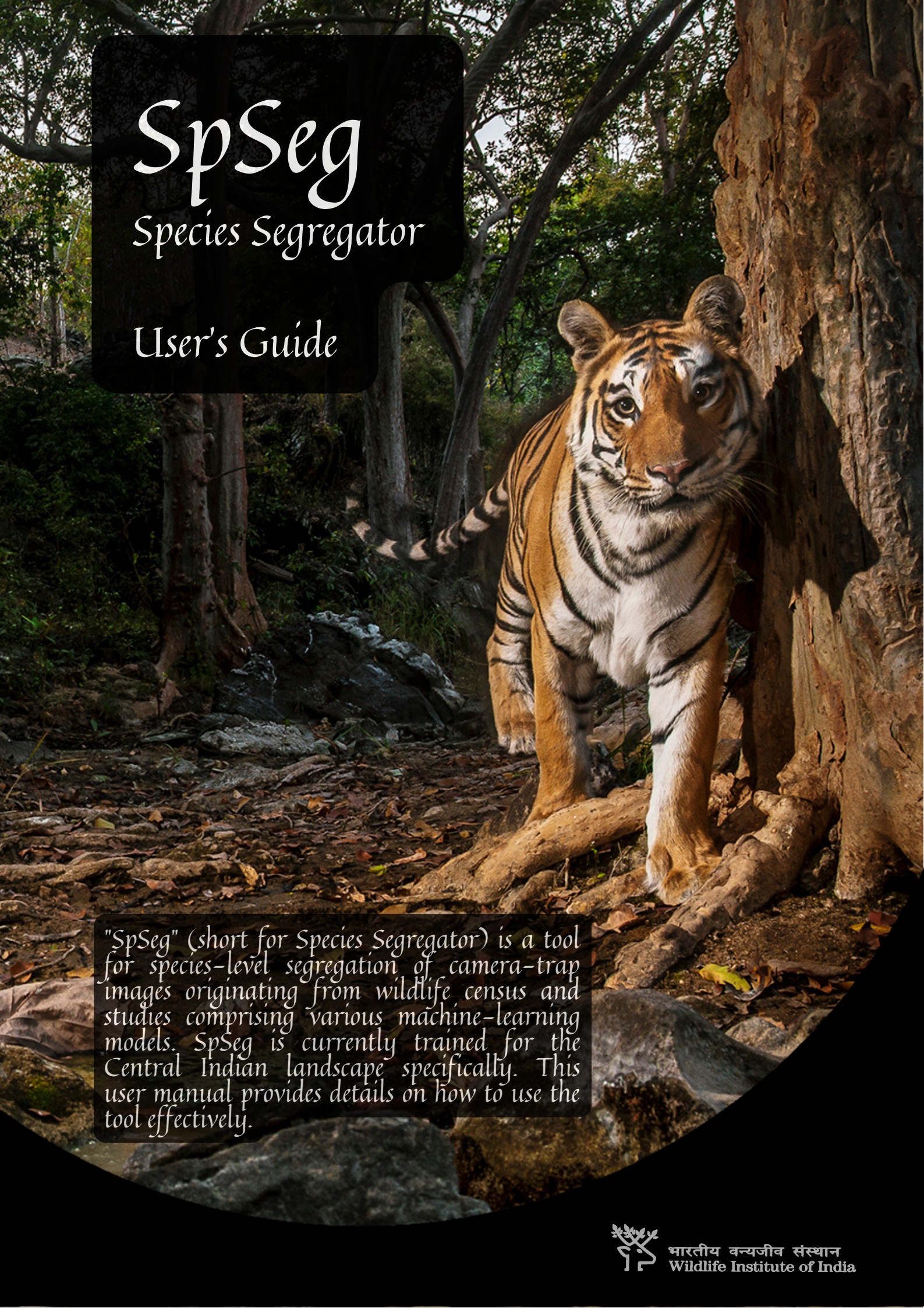


SpSeg

Species Segregator

User's Guide

"SpSeg" (short for Species Segregator) is a tool for species-level segregation of camera-trap images originating from wildlife census and studies comprising various machine-learning models. SpSeg is currently trained for the Central Indian landscape specifically. This user manual provides details on how to use the tool effectively.

भारतीय वन्यजीव संस्थान
Wildlife Institute of India

SpSeg
Species Segregator

User's Guide

Version 1.0
Last updated: 3rd October 2022

Shivam Shrotriya
Venkanna Babu Guthula
Indranil Mondol
Bilal Habib

"SpSeg" (short for Species Segregator) is a tool for species-level segregation of camera-trap images originating from wildlife census and studies comprising various machine-learning models. SpSeg is currently trained for the Central Indian landscape specifically. This user manual provides details on how to use the tool effectively.

भारतीय वन्यजीव संस्थान
Wildlife Institute of India

SpSeg
Species Segregator

User's Guide

Version 1.0
Last updated: 3rd October 2022

Technical Report No. 2022/29

"SpSeg" (short for Species Segregator) is a tool for species-level segregation of camera-trap images originating from wildlife census and studies comprising various machine-learning models. SpSeg is currently trained for the Central Indian landscape specifically. This user manual provides details on how to use the tool effectively.

Check for the new version of the tools and this manual at
https://github.com/bhlab/SpSeg

भारतीय वन्यजीव संस्थान
Wildlife Institute of India

Citation

Cite the use of SpSeg models as:

> *Shrotriya, Shivam; Guthula, Venkanna Babu; Mondal, Indranil; and Habib, Bilal (2022). SpSeg User's Guide, version 1.0. TR No. 2022/29. Wildlife Institute of India, Dehradun, India.*

And, don't forget to cite MegaDetector (without which SpSeg won't even exist):

> *Beery, Sara; Morris, Dan; and Yang, Siyu (2019). Efficient Pipeline for Camera Trap Image Review. arXiv preprint arXiv:1907.06772*

License

Acknowledgements

Dr Bilal Habib's lab at Wildlife Institute India, Dehradun, India is a collaborator in the development, evaluation and use of MegaDetector models. Development of SpSeg was supported by Microsoft AI for Earth (Grant ID:00138001338). We thank Mahrastra Forest Department for funding long term research in the state of Maharashtra. Kathan Bandopadhyay helped in preparing training dataset. The researchers and volunteers are thanked for their time for reviewing and testing the tools in real workfow.

Contents

The problem of camera-trap data sorting

Biodiversity conservation is often challenged by the lack of information on species presence and their populations. Scientific methods to obtain such information are time-consuming not only during field surveys but also during data processing, thus delaying the decision making. Auto-triggered camera-traps to detect the wildlife species have become a popular tool for remote wildlife monitoring. However, segregation of the animal images out of millions of images generated in camera-trapping exercises and tagging those with species identification remains a daunting task.

With the modern advancement in technology, several Artificial Intelligence (AI) based tools have come to the rescue. Microsoft AI for Earth team's MegaDetector is one such solution developed by Microsoft AI for Earth team (Beery et al. 2019).

> *"The approach here is to first classify the images into Animal, Person, Vehicle and Blank, followed by second level classification of animal images into useful taxonomic classes."*

📝 Check out this detailed overview of the approach in a Medium article and an explainer by Dan Morris on why this two-step process.

Microsoft AI for Earth team's MegaClassifier model was an effort towards second level of classification. However, global models of species classification may not perform well for site-specific dataset due to high morphological confusion among closely related species. For example, the latest global MegaClassifier 'v0.1_efficientnet-b3' model resulted in only 15.95% for the Central Indian Landscape datasets during our testing. High inaccuracy of global models for landscape-specific data required the development of models fine-tuned for the landscape and species.

SpSeg: A solution we bring out

SpSeg is abbreviation for 'Species Segregator', a set of models and tools that we have developed for species-level classification of camera-trap images at the Wildlife Institute of India, Dehradun, India. SpSeg currently works for the data from Central Indian Landscape, which is the Deccan Plateau part of Madhya Pradesh, Chattisgarh and Maharshtra.

We initially trained different CNN architecture models from keras (see the results). We found two models performing well - `SpSeg_x.hdf5` and `SpSeg_r152v2.hdf5` - and have tested those on various field datasets. `SpSeg_x.hdf5` is an Xception model that achieved 88.9% test accuracy, and `SpSeg_r152v2.hdf5` is a ResNet152v2 model that achieved 89.2% test accuracy. Find the models from a publicly shared SpSeg_Models Google Drive folder.

Training dataset

SpSeg models can currently segregate 36 commonly encountered species (or taxonomic class) in camera-trap surveys in the Eastern Vidarbha Landscape, Maharashtra, India.

Check our GitHub repo for updates in the species number and wildlife image data.

Species (Scientific name)	Image set	Species (Scientific name)	Image set
00_barking_deer (*Muntiacus muntjak*)	7920	18_langur (*Semnopithecus entellus*)	12913
01_birds (Excluding fowls)	2005	19_leopard (*Panthera pardus*)	7449
02_buffalo (*Bubalus bubalis*)	7265	20_rhesus_macaque (*Macaca mulatta*)	5086
03_spotted_deer (*Axis axis*)	45790	21_nilgai (*Boselaphus tragocamelus*)	6864
04_four_horned_antelope (*Tetracerus quadricornis*)	6383	22_palm_squirrel (*Funambulus palmarum* & *Funambulus pennantii*)	1854
05_common_palm_civet (*Paradoxurus hermaphroditus*)	8571	23_indian_peafowl (*Pavo cristatus*)	10534
06_cow (*Bos taurus*)	7031	24_ratel (*Mellivora capensis*)	5751
07_dog (*Canis lupus familiaris*)	4150	25_rodents (Several mouse, rat, gerbil and vole species)	4992
08_gaur (*Bos gaurus*)	14646	26_mongooses (*Urva edwardsii* & *Urva smithii*)	5716
09_goat (*Capra hircus*)	3959	27_rusty_spotted_cat (*Prionailurus rubiginosus*)	1649
10_golden_jackal (*Canis aureus*)	2189	28_sambar (*Rusa unicolor*)	28040
11_hare (*Lepus nigricollis*)	8403	29_domestic_sheep (*Ovis aries*)	2891
12_striped_hyena (*Hyaena hyaena*)	2303	30_sloth_bear (*Melursus ursinus*)	6348
13_indian_fox (*Vulpes bengalensis*)	379	31_small_indian_civet (*Viverricula indica*)	4187
14_indian_pangolin (*Manis crassicaudata*)	1442	32_tiger (*Panthera tigris*)	9111
15_indian_porcupine (*Hystrix indica*)	5090	33_wild_boar (*Sus scrofa*)	18871
16_jungle_cat (*Felis chaus*)	4376	34_wild_dog (*Cuon alpinus*)	7743
17_jungle_fowls (Includes *Gallus gallus*, *Gallus sonneratii* & *Galloperdix spadicea*)	4760	35_indian_wolf (*Canis lupus pallipes*)	553

A maximum of 5000 images were randomly selected for each species from our image dataset to train the models. The models were trained with 70% of the selected data, while 15% images were used for validation and remaining 15% were used for testing. Therefore, species with more than 5000 images in our image dataset have better accuracy, except sexually dimorphic species like Nilgai.

How does it all work?

SpSeg model (either SpSeg_x.hdf5 or SpSeg_r152v2.hdf5) reads the results of MegaDetector and classifies the animal images into species (or a defined biological taxonomic level). MegaDetector detects ***Animal***, ***Person*** or ***Vehicle*** in the image and tags it with a bounding box telling exactly where the item is in the image.

MegaDetector detects an animal within a bounding box on an image.

SpSeg models read the information for only ***Animal*** tags above 0.85 confidence from MegDetecor and classifies those up to species level. Therefore, both the models (detector and classifier) are required to run simultaneously on the images.

SpSeg model identifies an Indian fox within the bounding box.

📝 You can find more about how you can run MegaDetector on your images from this official MegaDetecor GitHub page.

⚠️ Keep in mind that the SpSeg is developed with MDv4. Although MDv5 has been launched, at this moment we recommend to stick with MDv4. We shall integrate SpSeg tools with MDv5 in future updates.

Setting up the tools

We are covering only basic information here. These instructions are quite similar to the instruction for MegaDetector installation, where more details could be found. For technical details check the official websites of the mentioned software.

Step 1. Installation

Download and install Anaconda (or Miniconda), which is a software platfor enabling to create environments to install machine learning frameworks and python tools. Installing Git is optional, but recommended to keep your repo updated. If you are using a GPU, you also need to check if you have the latest NVIDIA drivers.

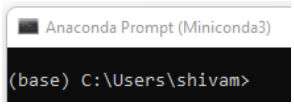
Step 2. Access the detector and classifier models

Download MegaDetector model v4.1.0 and SpSeg models and keep at an accessible location. These instructions will assume that the models are downloaded to a folder called `c:\spseg_models`.

Step 3. Clone git repo and set Python environment

There is no need to setup a separate MegaDetector environment, which is incorporated in the codes here. We are giving the instructions for Windows machines only, which is expected to work on Linux machines in a similar way. The environment is not tested on Mac.

Open Anaconda Prompt and make sure you have not opened an Anaconda Powershell Prompt or Windows Command Prompt. It should look something like this with the name of environment in the parenthesis:



Run the following commands one by one to download SpSeg repo from GitHub:

```
mkdir c:\git
cd c:\git
git clone https://github.com/bhlab/SpSeg
```

The commnads above are suggestions. `git` command will work at any directory and you can download GitHub wherever you find it convinient. Find more about git.

Alternatively, you can download the SpSeg repo as zip folder from https://github.com/bhlab/SpSeg and unzip it to a desired location.

Next, you need to enter where SpSeg repo is located using `cd` command:

```
cd c:\git\spseg
conda env create --file environment_SpSeg.yml
```

You are all set to run SpSeg!

Camera-Trap image data organization

We assume that your camera-trap data was organized in either of the following directory structures:

dataDir/Station
dataDir/Station/Camera

The first structure arrives when a single camera was placed at the station, while the second structure arrives when A and B cameras were placed facing each other. The directory before Station ID (dataDir) could include site ID, range ID and/or block ID.

Running SpSeg

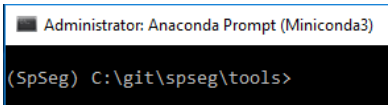
Next time when you open Anaconda Prompt, first you need to activate SpSeg environment:

```
conda activate SpSeg
```

and then locate the SpSeg tools folder.

```
cd c:\git\spseg\tools
```

Which should look something like this:



The main tool to use SpSeg models is **`run_spseg.py`**

Once you are in the right setup (environment and tools folder), the following command calls detector and classifier models and runs over all the images:

```
python run_spseg.py "C:\spseg_models\md_v4.1.0.pb" "D:\dataDir" --recursive --
spseg_model " C:\spseg_models\SpSeg_r152v2.hdf5" "D:\dataDir\data_output.json"
```

"D:\dataDir" is the location (local or on network server) where your organized camera-trap images are stored. You can set the path and name of the output file at "D:\dataDir\data_output.json".

The above command returns a *.json* directory for each image with detector tags for animal, person or vehicle, and species tags for animals (both the tags with a confident value).

💡 The image paths are retuned as absolute, which creates mismatches during further processing of the data specially if processing is carried out on a different system. Therefore, the best practice is to get the image paths as relative and keep the output file within image folder.

Relative paths in the output can be returned by setting the flag for it:

```
python run_spseg.py "C:\spseg_models\md_v4.1.0.pb" "C:\dataDir" --recursive --
spseg_model " C:\spseg_models\SpSeg_ r152v2.hdf5" "C:\dataDir\data_output.json" --
output_relative_filenames
```

How to use the output to process the images?

AI-tools are not 100% accurate. Therefore, a workflow is required to process the output and correct for the errors. Currently, there are two approaches to process the output and finalize the identification/segregation of the images for further analyses-

1. Using Timelapse to review the images (only for Microsoft Windows systems),
2. Using an Image file manager system like Windows File Explorer, ExifPro, IrfanView or FastStone.

📝**Note**: A third approach of transferring annotations to the hierarchicalSubject field of XMP data of the image and processing thorugh DigiKam works with the output of MegaDetector only, and shall be soon available for SpSeg output.

Timelapse workflow

Timelapse is an image analyzer and tagging software sepcifically developed to tag camera-trap images by Dr. Saul Greenberg. It is well integrated with MegaDetector system and is one of the fastest methods we have tested so far. Additional advantage

of Timelapse software is that the data can be exported as spreadsheet after reviewing and correcting. The data from Timelapse can be directly read into R and processed by camtrapR (the most popular R tool to generate basic reports and input files for further analyses on camera-trap data). We have tested it on multiple datasets. Hence, we highly recommend to use Timelapse to review the images.

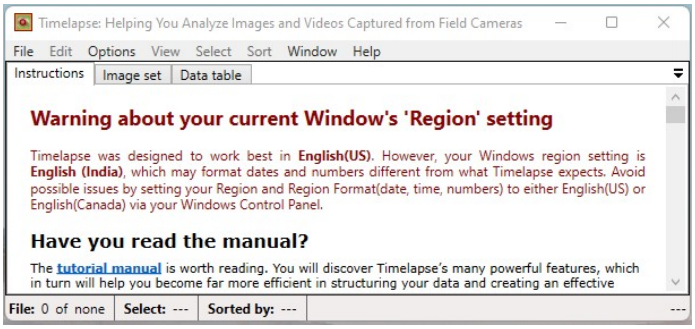
A detailed information on how to use Timelapse and how to use image recognition in Timelapse can be found at the official site of Timelapse. Before you start the review, familiarize yourself with the software and its functions (see the video tutorials for various functions and features of the software).

Here we will cover only the essential steps of the workflow we have implemented to review SpSeg output and tag the images.

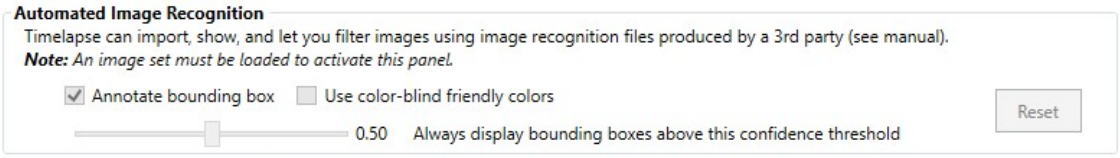
Step 1. Download Timelapse zip folder and unzip it. You don't need to install it.

Step 2. Copy Timelapse_SpSeg.tdb from SpSeg repo folder C:\git\spseg to your image data folder D:\Camera_data (this *.tdb* file is compatible with Timelapse version 2.3.0.0 and above). Also, make sure that SpSeg output in .json file is stored in the same data folder and image paths in the json file are relative.

Step 3. Launch Timelapse software by double-clicking Timelapse2.exe in the unzipped Timelapse folder. The opening screen looks something like this:

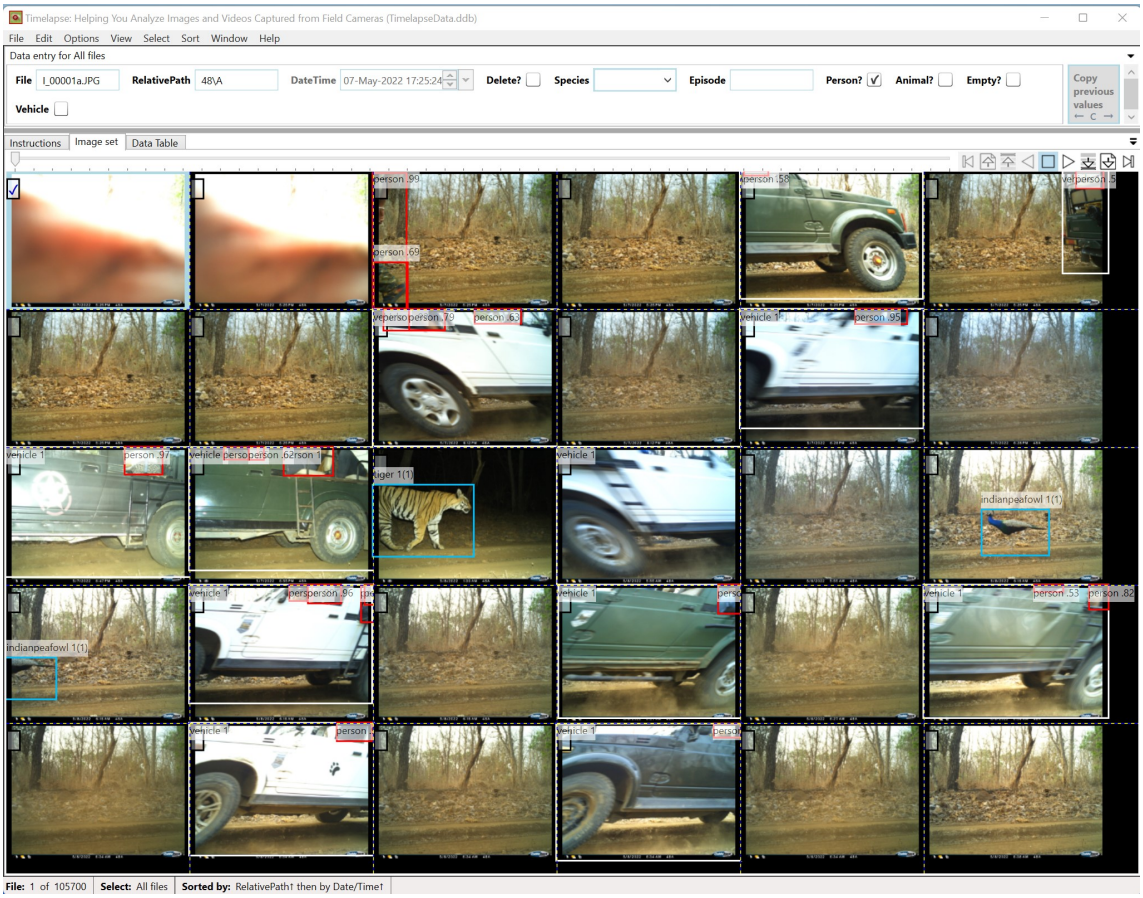


Step 4. To turn on recognition capabilities, select **Option | Preferences** and check the 'Annotate bounding box' and keep the confidence threshold at 0.50:



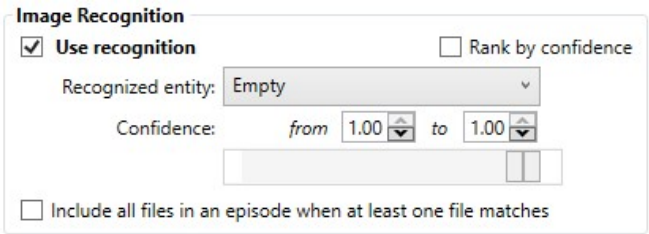
Step 5. Load the template by selecting **File | Load template, images, and video files…** Navigate to the image directory and select the Timelapse_SpSeg.tdb file. Timelapse with take some time to search and load all the images.

Step 6. Import the SpSeg output file by selecting **File|Import image recognition data for this image set**. Navigate to the image directory and select data_output.json file. At this point images with auto-detected objects will start showing bounding boxes around the objects.

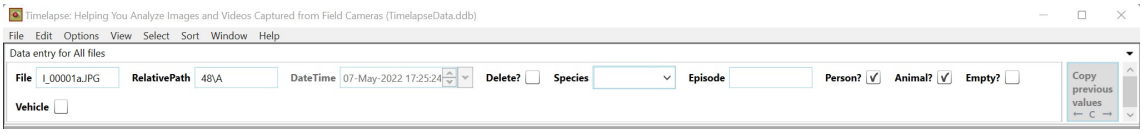


Step 7. The images can be filtered by AI output using **Image recognition** settings in **Select | Custom Selection**.

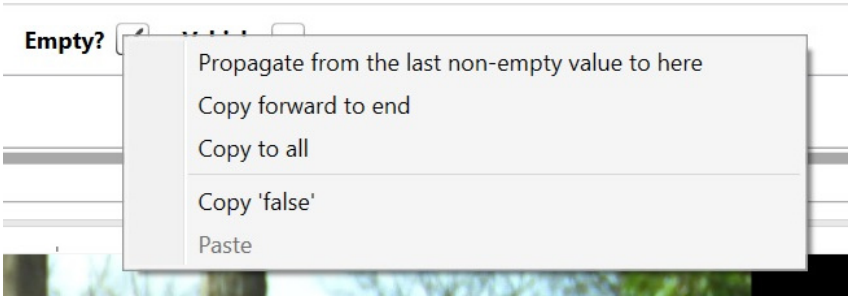
Step 8. First review all the ***Empty*** images by setting image recognition to Empty and confidence limit from 1.0 to 1.0 (as the ***Empty*** images do not have any confidence value).



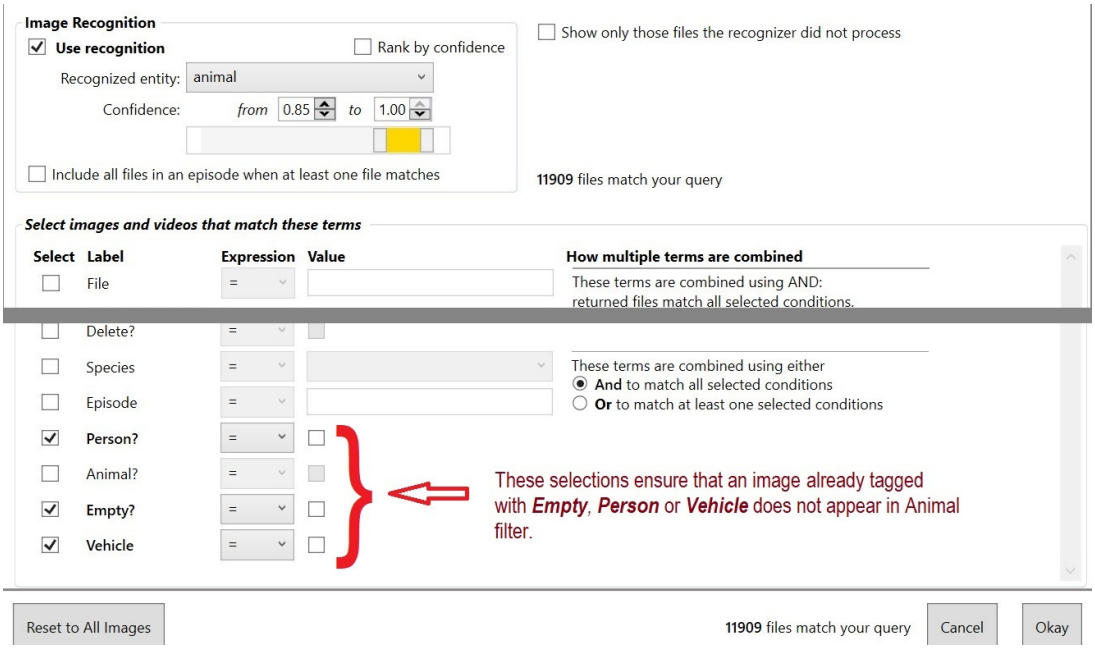
If an image is not empty, check appropriate flags:



After reviewing, check back the custom selection and omit the images that have been tagged as other than ***Empty***. Then tag the remaining confirmed images using 'Copy to all' function by right-clicking Empty flag:



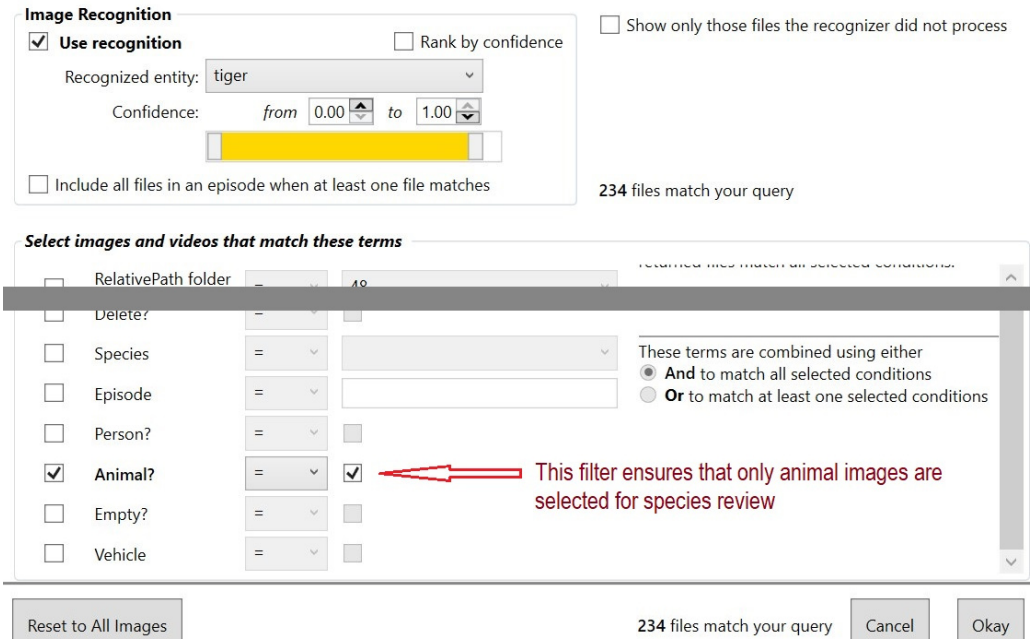
Step 9. Next, review ***Person, Vehicle*** and ***Animal*** images one tag at a time. You can review these images in two segments. Since images above 0.85 confidence level are usually correct, these can be reviewed at a faster speed, and the image with less than 0.85 confidence level can be reviewed more carefully.



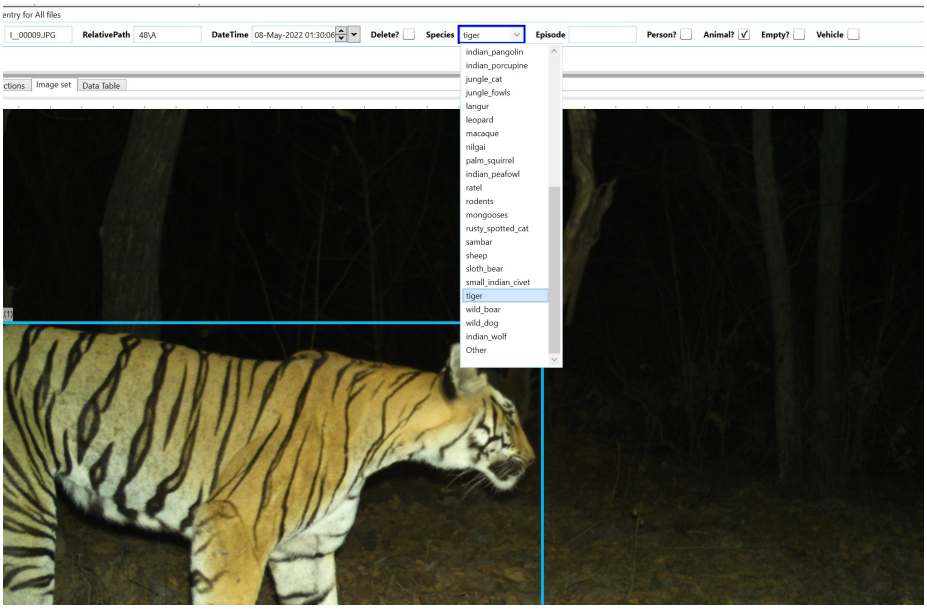
An image wrongly appearing in a filter or if contains two or more types of objects should be corrected immediately and tagged appropriately.

Step 10. After correcting the tags from MegaDetector, species-level image recognition should be reviewed for only the animal images. Images for a particular species filter

can be selected from the Image Recognition window.



Tag a species from Species dropdown menu. All the images in the filter can be tagged to the same species by right-clicking and selecting ‘Copy to all’ option.

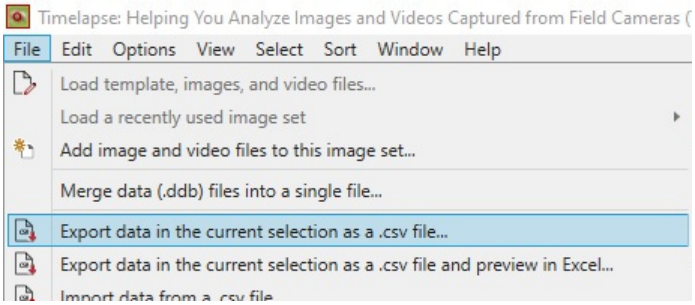


Duplicate images: If an image has more than two species, it should be duplicated by clicking on **Edit | Duplicate this record** and each image should be tagged separately.

⚠ Note that selecting **Rank by confidence** option in the *Image Recognition* window excludes duplicate image from current filter. Therefore, avoid using this option.

Step 11. Double check if all the images are reviewed and correctly tagged by unselecting **Image Recognition** and then selecting data filters.

Step 12. Export the data from **File | Export data in the current selection as a .csv file…** option. Clear all the filters and custom selections before exporting the data.



 An R script TL2RT.R is available in the SpSeg folder C:\git\spseg to read the *.csv* output of Timelapse and convert it into *recordTable* format of camtrapR.

Handling large datasets

Sometimes the image dataset is too large and Timelapse stops responding. We observed that datasets above 2,00,000 images often could run into such issues. In such cases, it is advised to split-up the datasets into smaller parts (and the corresponding SpSeg output *.json* file) for easier handling. In fact, the partitions could be reviewed by multiple reviewers simultaneously.

subset_json_detector_output.py from MegaDetector's Postprocessing tools splits SpSeg output at defined node in file path. We split the ouput for top-level folder, which is usually the camera-trap ID. You don't need to download the tool separately. It is included in the SpSeg repo and you can directly run it from the SpSeg environment:

```
python subset_json_detector_output.py D:\Camera_data\Camera_data_output.json
D:\Camera_data --split_folders --split_folder_mode top
```

The tool will result in separate output *.json* files for each camera-trap. You can also stitch together these files as per your partition of the data. Let us illustrate this with an example. We had a dataset with 100 camera-traps. We first ran SpSeg model to get single output file. And then split it into 100 smaller files for each trap. Now we divided the entire dataset in two smaller parts of 50 camera-traps each. Hence, we needed to stitch first 50 *.json* files together and then remaining 50 *.json* files. We created an empty folder and copied 50 files that we need to stitch together. **merge_json.py** tool can read all the *.json* files in a folder and merge those into a single output:

```
python merge_json.py D:\Camera_data\part1 D:\Camera_data\Camera_data_part1.json
```

First you need to write the path of the folder containing split *.json* files, and then you need to give a path and name to the output file. In out example, data_output_part1.json will contain the information for only first 50 camera-traps.

If a large dataset was processed by splitting into smaller parts, the resulting *.csv* files for each part can be stitched together by rbind command in R or any other spreadsheet program.

Rearranging raw data after correcting the image identifications

At times the data is needed to be arranged in species level organization, e.g. for individual identification of the tigers and leopards. A tool **`sort_spseg.py`** can be used to reorganize the raw data using the *.csv* output of Timelapse in multiple ways.

You need to first open Anaconda Prompt and activate SpSeg environment (see Running SpSeg section).

If you need to export whole data reorganized in species-level folders within camera-trap ID folders for archiving or running camtrapR:

```
python sort_spseg.py D:\dataDir --recursive D:\dataDir\TimelapseData.csv --output_dir
D:\data_segregated
```

D:\data_segregated is a new location where reorganized data will be exported.

Let's export the data for only tigers to process for individual-level identification:

```
python sort_spseg.py D:\dataDir --recursive D:\dataDir\TimelapseData.csv --output_dir
D:\data_tigers --species tiger
```

You can also export multiple species:

```
python sort_spseg.py D:\dataDir --recursive D:\dataDir\TimelapseData.csv --output_dir
D:\data_ungulates --species wild_boar nilgai spotted_deer
```

Or you could export all images without blank (empty) images:

```
python sort_spseg.py D:\dataDir --recursive D:\dataDir\TimelapseData.csv --output_dir
D:\data_detections --exclude blank
```

And if you require only wild animal images:

```
python sort_spseg.py D:\dataDir --recursive D:\dataDir\TimelapseData.csv --output_dir
D:\data_wild_animals --exclude blank person vehicle dog goat
```

You can custom export by using either `--species` or `--exclude` tag followed by a list of items.

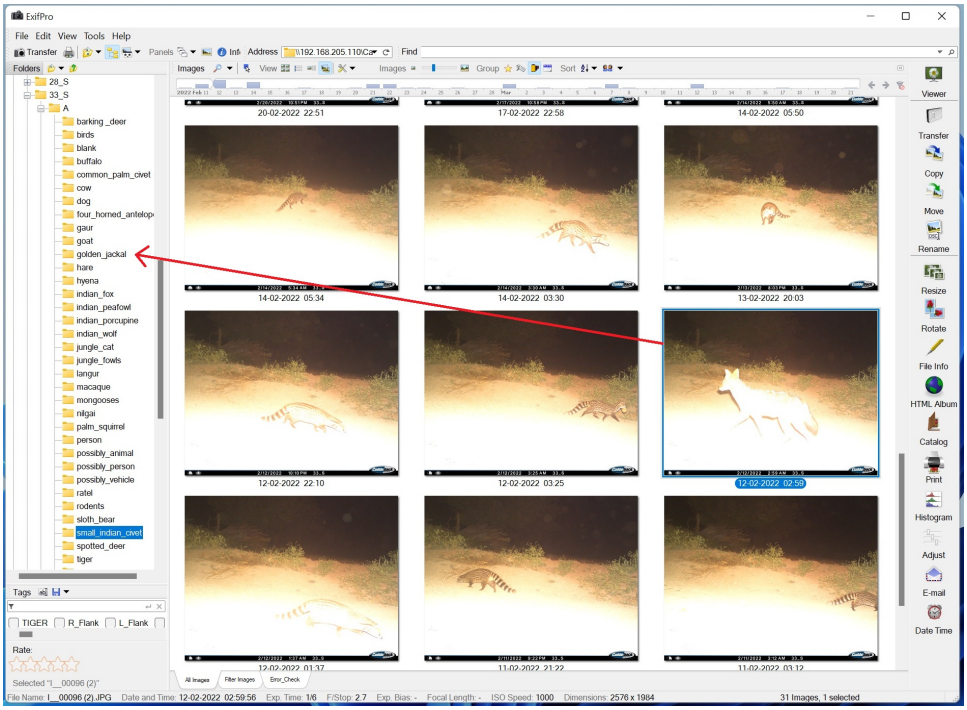
Image file manager workflow

If you wish to use a file manager system (Windows File Explorer, ExifPro, IrfanView or FastStone) to review and correct the errors instead of Timelapse, you can export a copy of the data reorganized into species-level folders directly based on the SpSeg output:

```
python sort_spseg.py D:\dataDir --recursive D:\dataDir\data_output.json --output_dir
D:\data_for_review
```

The tool exports images with <0.85 confidence level (for ***Animal***, ***Person*** or ***Vehicle*** tags) from MegDetector model to the folders named ***'possibly_animal'***, ***'possibly_person'*** and ***'possibly_vehicle'*** to find inaccuracies easiliy. Images with a high confidence (>0.85) ***Animal*** tag are segregated in species-specific folders. Although these images could also have inaccuracies and require a review.

The reviewer can access each folder from programs like ExifPro and move wrongly identified images to correct species folder within the same camera-trap.

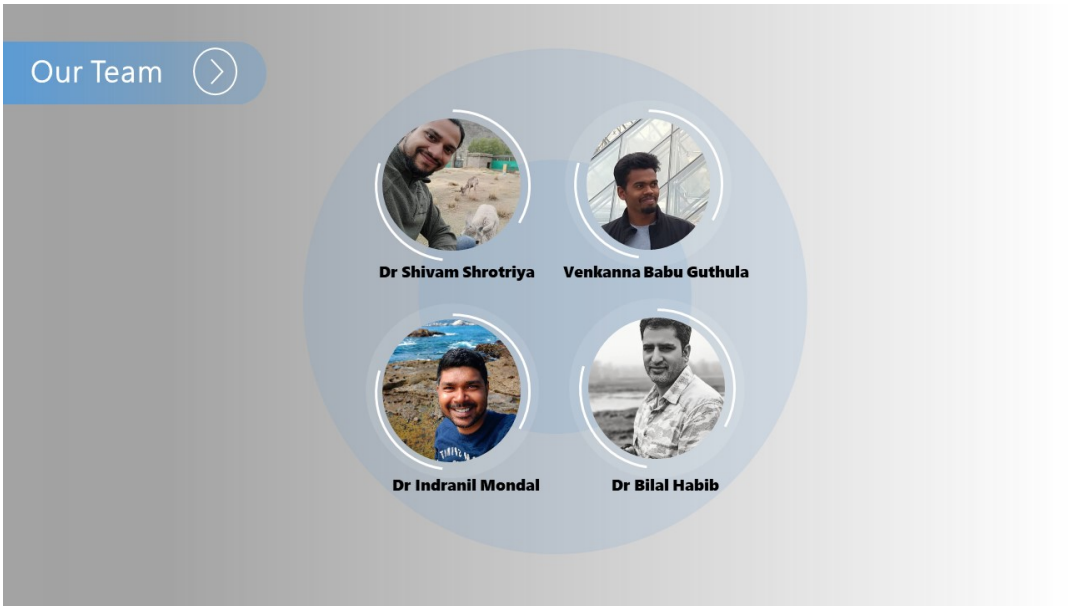


Further development

We have in pipeline- 1. integration with MegaDetector version5. 2. developement of DigiKam based workflow, 3. Further refinement of the SpSeg models (particularlly for dimorphic ungulate species and species with small dataset), and 4. Addition of more species, e.g. we missed out on the elephant in the current models.

So stay tuned!

Also, we would love to hear back from you. If you have any suggestion for additional species or would like to add to training dataset, reach out to us (bh@wii.gov.in) to Participate!



Dr Shivam Shrotriya is leading and managing this project at the lab and the GitHub repo. He is an ecologist with a PhD on the Himalayan wolf ecology and holds field research experience in the Himalayas and the Central India. Apart from this manual, he tested, edited and partly wrote SpSeg codes.

Mr Venkanna Babu Guthula is a python developer and machine learning enthusiast, who loves to work on AI-tools for sustainability and nature research. He has been instrumental in developing the codes and training the SpSeg models. He handles the backend of the project.

Dr Indranil Mondal is a Geographic Information Systems specialist and a technology (hardware/ networking) enthusiast who has played a pivotal role in designing the hardware specifications for this project and is looking at the tool dissemination and adoption by the managers and policymakers

Dr Bilal Habib is the principle investigator of the this project. SpSeg is develeped and hosted by his lab at the Wildlife Institute of India. He is a wildlife scientist involved in studying population dynamic of large carnivore, their prey and movement ecology at large scale studies in the Central India and the Himalayas. He is a patron of adopting advanced technologies like AI for conservation. Drop an email to bh@wii.gov.in for any querries and/or suggestions regarding SpSeg.

Find us on the web: www.wii.gov.in, https://bhlab.in/

SpSeg
Species Segregator

User's Guide

Version 1.0
Last updated: 3rd October 2022

Technical Report No. 2022/29

"SpSeg" (short for Species Segregator) is a tool for species-level segregation of camera-trap images originating from wildlife census and studies comprising various machine-learning models. SpSeg is currently trained for the Central Indian landscape specifically. This user manual provides details on how to use the tool effectively.

Cover Pic: Nayan Khanolkar

Check for the new version of the tools and this manual at
https://github.com/bhlab/SpSeg

Dr. Bilal Habib
Scientist E/Project Investigator
Department of Animal Ecology and Conservation Biology
Wildlife Institute of India
E-mail: bh@wii.gov.in

भारतीय वन्यजीव संस्थान
Wildlife Institute of India