The EventFinder Suite

Written: May, 2020, ~~Updated September 2021 with re-tagging data~~
Updated August 2024 with EventFinderAi

Overview

EventFinder is a suite a programs made at The King's University (Edmonton, Alberta) to manage images from camera traps from collection to classification. We find that solely manual management of camera trap images to be tedious and error prone. EventFinder tries to balance the advantages of software with human expertise. Currently EventFinder runs on Windows operating system and Java. Any version of Java 1.7.0_07 or newer should run EventFinder. You may determine which version of Java you have by typing "java -version" in a command line prompt.

EventFinder usage works in three stages, as shown in Figure 1. The first stage, the collection of images from the camera cards, uses the CameraTransfer portion of the suite. The images are renamed and moved into folders, each representing a site for processing. The second stage is computer processing using the images from CameraTranfer as input, and producing a comma separated values file as output (csv). There are three paths through this stage depending on the level of processing required. EventSplitter segments images into events based on the time difference between images. EventFinder processes sets of images to identify sequences containing only noise, and to identify the best image for human classification. EventFinderAi classifies each image set with its corresponding class. By default these classes include Vehicle, Human, Animal, nothing. The third stage is the human classification phase. EventTagger uses the camera trap images and the csv file produced in phase two to present an interface to the user with customizable tags.

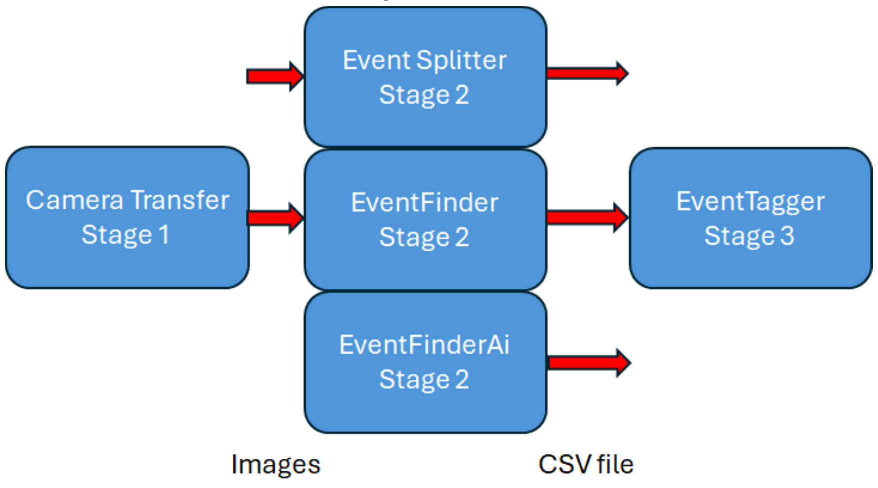


Figure 1: Workflow of the EventFinder Program

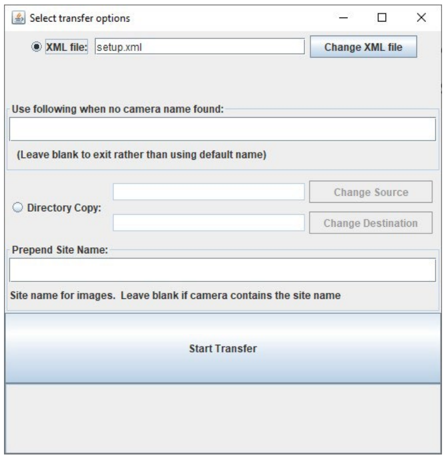
The EventFinder program was made first to reduce the number of images requiring human classification. In our research we found it could remove approximately 90% of the images, while retaining 90% of the events[1]. If more than 90% event retention is required, options in the current software enable examination of all captured images. Our use of EventFinder highlighted inefficiencies leading to creation of other software to handle additional camera trap image tasks. Hence the software is labeled as the EventFinder suite, event though only one program in the suite is named EventFinder.

[1]Janzen, M., Ritter, A., Walker P.D. and Visscher D.R. EventFinder: a program for screening remotely captured images. *Environmental Monitoring and Assessment* **191,** 406 (2019).

The EventFinder Suite is free to use. Portions of its creation have been supported by grants from Alberta Parks and The Alberta Conservation Association. If you use our program in your work we ask that you list us in the Acknowledgments section of resulting publications, and we would appreciate an email to let us know that the software is being used (Michael.Janzen@kingsu.ca). Knowing that the software is useful to researchers and others is encouraging for further development, and important for justifying the continued development time spent on the software. Donations to supporting the research are always appreciated, but not required for full use of the software. Interested parties can contact development@kingsu.ca to pursue financial donation.

Stage One: CameraTransfer

The CameraTransfer program, in the directory labelled CameraTrapTransfer, is designed to move images into suitable directories for processing by later components of the program. The program organizes images for efficient future processing, chiefly that all images from a site are in the same directory. There are two ways of using CameraTransfer: "xml transfer" and "directory transfer". The xml transfer method is shown as selected in Figure 2. We built CameraTransfer with xml transfer in mind. When using the xml transfer method each camera should be labelled with the site location. For example, see Figure 3 where the Reconyx camera has the camera name "K20", for King's Camera site 20, in both the image and metadata. Using the site name as the camera name works well with Reconyx


Select transfer options

[✓ XML file:] setup.xml

Use following when no camera name found:

(Leave blank to exit rather than using default name)

Directory Copy:

Prepend Site Name:

Site name for images. Leave blank if camera contains the site name

Figure 2: CameraTransfer Options

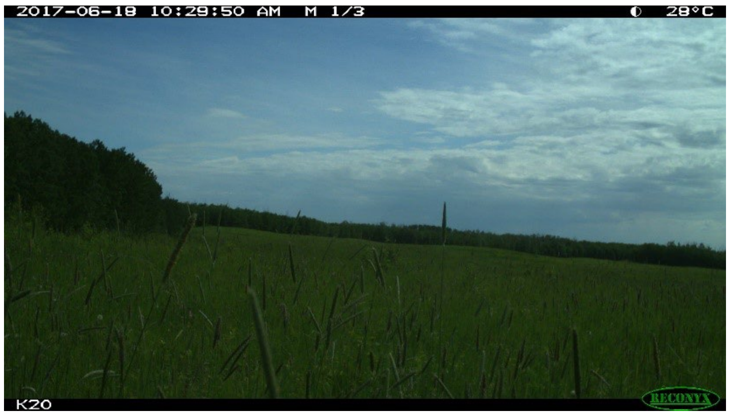


Figure 3: Sample Reconyx Image named K20 after King's site 20.

branded cameras, and our tests include the models HC500 HYPERFIRE, HC600 HYPERFIRE, PC800 PROFESSIONAL, PC900 PROFESSIONAL, and HF2 PRO COVERT. Other Reconyx models will also likely work, and camera models without a camera name in the images can work with a user provided name when transferring (for example, with Spypoint camera models).

Organizational Tip:

The EventFinder suite can fill in repetitive information in the workflow if you use expected directory names. Parent directories can be assigned as suits the user, but the directory to transfer images into should end in a directory called "input". For example, "C:\Users\Janzen\ResearchStudy\Site01\input". Subsequent programs will look for the word "input" to pre-fill in later options.

Stage One: Camera Transfer - XML transfer method

The xml method of transferring images uses an xml file to determine where to move camera trap images to, based on their camera/site name. In our usage we place a camera memory card into our computer, run the program, and the images are automatically placed into the appropriate directories based on the xml file. The default xml file name is "setup.xml". You may edit this file for your needs or create a new xml file and specify that xml file each time you run the program. It is designed to be human readable and editable, but please note that the program expects a rather strict structure. An abbreviated xml file is shown in Figure 4. Please note that each tag has a matching closing tag, and these opening and closing tag names must match. Text inside a <comment> is ignored by the CameraTransfer program. Text inside each <sourceDirectory> tag should contain the path of one directory to look for images, and multiple <sourceDirectory> tags should be used to have the program look through multiple directories. In our example, we specify the memory card location that is mapped to the D: drive of our computer. This example only lists two directories for brevity, while our working file lists additional directories from 100RECNX through 109RECNX.

```
<cameraTransfer>
  <version>
    <major>1</major>
    <minor>1</minor>
  </version>
  <comment>Where to look for the SD camera card</comment>
        <sourceDirectory>D:\DCIM\100RECNX</sourceDirectory>
        <sourceDirectory>D:\DCIM\101RECNX</sourceDirectory>
  <comment>One or more directories to place images</comment>
```

```
<validDestination>
  <directory>Z:\ResearchStudy\K01\input</directory>
  <cameraName>K01</cameraName>
  <deleteWhenMoved>no</deleteWhenMoved>
</validDestination>
<validDestination>
  <directory>Z:\ResearchStudy\K20\input</directory>
  <cameraName>K20</cameraName>
  <deleteWhenMoved>no</deleteWhenMoved>
</validDestination>

<comment>One or more camera models</comment>
<validCameraModel>HC500 HYPERFIRE</validCameraModel>
<validCameraModel>HC600 HYPERFIRE</validCameraModel>
<validCameraModel>PC800 PROFESSIONAL</validCameraModel>
<validCameraModel>PC900 PROFESSIONAL</validCameraModel> </cameraTransfer>
```

Figure 4: Abbreviated Sample xml setup file

Valid destinations specify a mapping from a camera name to a directory to move the images. In our example all images taken with camera K01 (site K01) are copied to "Z:\ResearchStudy\K01\input". Please note that the destination directory must exist before running CameraTransfer. Only two pairings are shown for brevity, and your program should have a <validDestination> for each camera you use in your study. The last section of the xml file lists camera models that are known to work. This functions as a flag for CameraTransfer - if the model of camera is listed then CameraTransfer will proceed. You may add additional camera models but test each one to ensure correct functionality. We have found that adding additional Reconyx models seems to work without further modification. The option to delete images when moved is not functional in this version of CameraTransfer and we delete the images when the memory card is returned to the camera.

If you have a camera model that does not record the name in the image, or the name is not correctly found by CameraTransfer, you may specify a default camera name to use when CameraTrap does not find a name. In this case specify the "Use following when no camera name found" name text field to manually enter the camera/site name (Figure 2). Please note that even when you provide the name of the camera you must have a matching camera/site name in your xml file so that CameraTransfer can look up the destination directory. When using the xml transfer method
CameraTransfer will exit if it doesn't find a corresponding camera model name in the xml file. Lastly, the option to prepend a site name to output image names will function with an xml transfer but may be left blank if your camera name is your site name.

To run CameraTransfer you may select the CameraTrapTransfer.bat batch file or type "java CameraTransfer" from the command line in the appropriate directory. The command line method may provide additional details if something goes wrong when transferring images. We use the "setup.xml" file to specify the camera / directory mappings, and no prepended site name. Consequently, we just click “Start Transfer” to move images after placing a camera memory card in our computer. CameraTransfer will copy the images from the input directory (the memory card in our case) to the destination directory, renaming each image during the copy. The image names will be of the format SiteName_CameraName_Date_Time_Number.JPG. For example, the image in Figure 3 will be named “K20_2017_06_18_10_29_50_1.JPG” since we have no prepend site name, and this is the first image with this date and time. For images taken less than one second apart the final number will increment, to avoid overwriting images. Using this formatting of images gets around the problem where the camera has multiple images with the same name. For example, we may encounter a IMG0001.JPG in multiple directories, or during subsequent transfers.

How does CameraTransfer find the camera name in the metadata?

CameraTransfer does not know the Reconyx metadata structure so it first looks for the camera model name, and then examines fields in the metadata until a match for the camera name is found with one in the xml file. So, don’t name your site name as just the year (example: 2019) since this field will be encountered first.

Stage One: Camera Transfer - Directory Transfer Method

In some studies, there may be multiple cameras used at the same site, at different times. In this case the user may have already transferred images into directories based on site name, but from different cameras. In this case the directory copy method may be used to copy and rename images (the second option in Figure 2). Here the user selects the “Directory Copy” method, and provides a site name to prepend to copied image file names. The program will copy all images from the source directory to the destination directory renaming the images using the same format as with the XML copy method (see above). When using the directory transfer method, if no matching camera name is found in the xml file, the camera name is omitted in the output image filename. In order for the camera name to be included in the image file name the camera name must appear in the <validDestination> pair in the xml file. In fact, the directory transfer method works the same as the xml transfer method, but the destination directory from the <validDestination> is replaced with the one provided by the user. The user may repeat this process, specifying different directories each time, until all the files are copied.

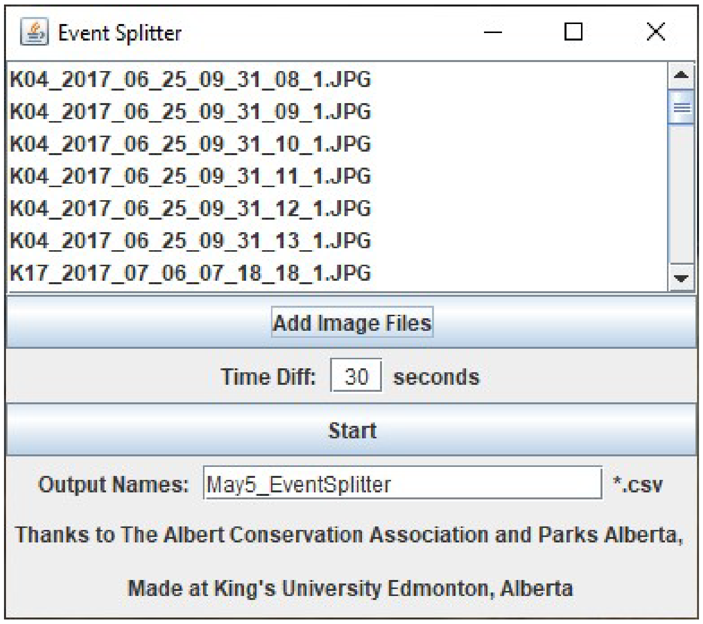
Stage Two: Computer Image Processing

At this point there are images in directories that ultimately need to be classified. If you used CameraTransfer in Stage One, then these images should be in directories based on site locations. If you have manually move images, then there may be another directory structure. Other directory structures should work, so long as the times don't overlap. For example, two cameras at different locations taking pictures at the same time should be processed in separate runs of the program in stage two.

There are two options for dividing camera trap images into sets: EventSplitter and EventFinder. Both options use the time difference between images – the names of the images are mostly irrelevant at this stage (hence the reason you should not have two sites in the same directory). EventSplitter divides sequences of images into image sets of events based on the time difference between sequential images, where EventFinder does everything EventSplitter does, plus uses image processing to try to determine sets containing only noise and identify the best image in sets containing events.

Stage Two: Computer Image Processing – EventSplitter

EventSplitter divides sequences of images into image sets of events based on the time difference between sequential images. The default time difference to divide image sets is 30 seconds, but this is a user settable parameter, as shown in Figure 5. Images are added using the "Add Image Files" button and CTRL-A on a Windows machine is handy to select all the files in a directory. All the files in a directory should come from the same site (although we have events from cameras K04 and K17 in our example to demonstrate different events later in stage 3). Please name your csv file in a meaningful way. We typically use site name and date.


Event Splitter

K04_2017_06_25_09_31_08_1.JPG
K04_2017_06_25_09_31_09_1.JPG
K04_2017_06_25_09_31_10_1.JPG
K04_2017_06_25_09_31_11_1.JPG
K04_2017_06_25_09_31_12_1.JPG
K04_2017_06_25_09_31_13_1.JPG
K17_2017_07_06_07_18_18_1.JPG

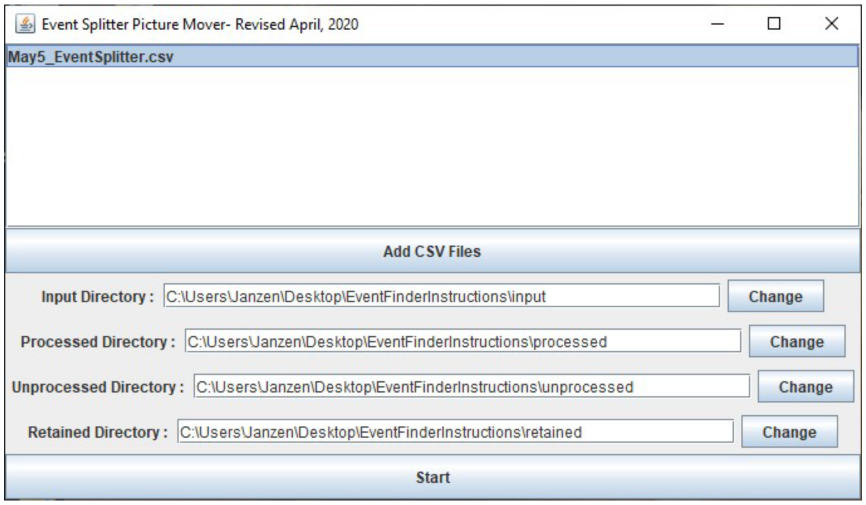
Time Diff: 30 seconds

Output Names: May5_EventSplitter *.csv

Thanks to The Albert Conservation Association and Parks Alberta,

Made at King's University Edmonton, Alberta

Figure 5: EventSplitter



May5_EventSplitter.csv

Input Directory : C:\Users\Janzen\Desktop\EventFinderInstructions\input

Processed Directory : C:\Users\Janzen\Desktop\EventFinderInstructions\processed

Unprocessed Directory : C:\Users\Janzen\Desktop\EventFinderInstructions\unprocessed

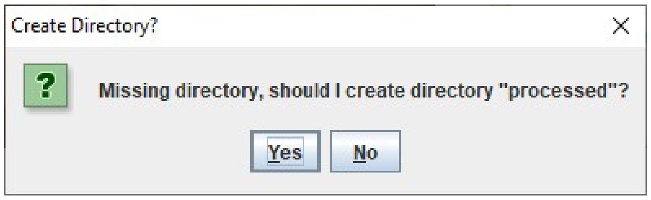
Retained Directory : C:\Users\Janzen\Desktop\EventFinderInstructions\retained

Figure 6: Event Splitter Mover program

After EventSplitter completes, a prompt displays asking to move the files. Selecting "Yes" will bring up the "Event Splitter Picture Mover" program, shown in Figure 6. Alternatively, the Picture Mover program can be run later by using the "EventSplitterMover" batch file in the CameraTrapEventSplitter directory.

The picture moving software requires a .csv file created by EventSplitter. This will be filled in automatically if "Yes" was selected from the prompt. While EventSplitter should be able to handle multiple csv files from the same site, it is best to run it with just one csv file at a time (the default way it runs if run from the prompt). The three new directories have meanings as follows:

- Processed: Images processed by EventSplitter are put here
- Unprocessed: Any images not processed by EventSplitter (should be empty)
- Retained: Empty for EventSplitter as the best image in a set is not determined



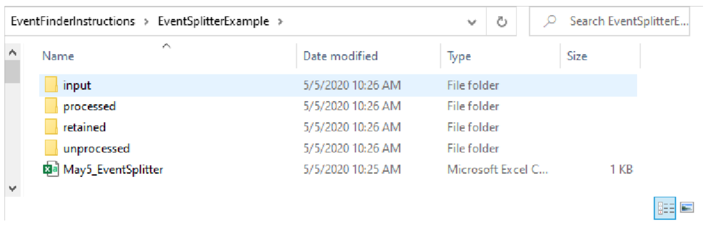
Missing directory, should I create directory "processed"?

Figure 7: Prompt to create a directory

If your input path ended with the directory name "input" then the moving software will automatically suggest directory names for the processed, unprocessed, and retained directories. After clicking "Start" Event Splitter Picture Mover will prompt to create each of the three new directories if they do not exist, as shown in Figure 7.

At the end of running EventSplitter and Event Splitter Picture Mover, the images from the "input" directory should be moved to "processed" and the "input", "retained", and "unprocessed" directories should be empty. In the EventSplitter directory should be the csv file, in our case named "May5_EventSplitter.csv". This is our example name, please use a name that better refers to the images processed such as site location and date.

The csv file can be moved to the directory containing the images, so long as it's above the "processed" and "retained" directories, as shown in Figure 8. We find this workflow helps to keep track of which pictures require processing as future camera memory card transfers can now be copied into the "input" directory.


EventFinderInstructions > EventSplitterExample >

Search EventSplitterE...

Name	Date modified	Type	Size
input	5/5/2020 10:26 AM	File folder	
processed	5/5/2020 10:26 AM	File folder	
retained	5/5/2020 10:26 AM	File folder	
unprocessed	5/5/2020 10:26 AM	File folder	
May5_EventSplitter	5/5/2020 10:25 AM	Microsoft Excel C...	1 KB

Figure 8: Directory after running EventSplitter

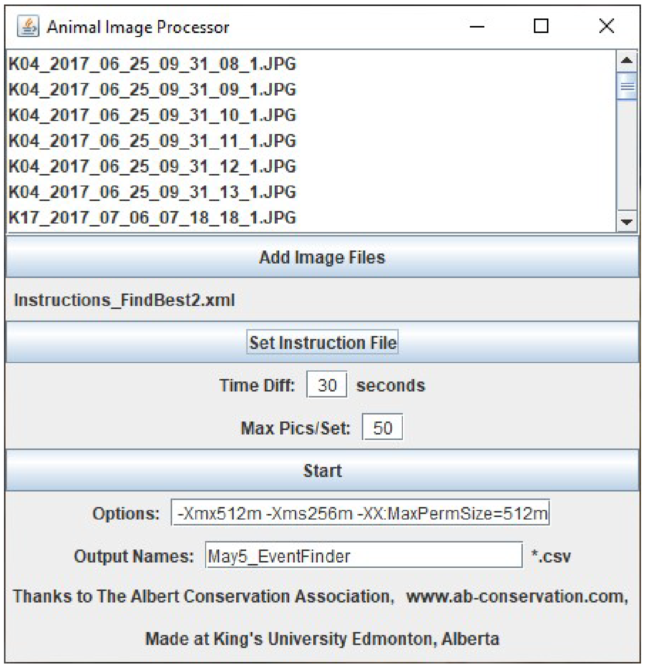
EventFinder was the earliest of the programs written and has been adapted for different purposes. The "PictureProcesser" program is for an earlier version that considers sequences of image with a linear crossing (fences) and identifies sequences where the animals cross (mostly ungulates) while removing animals that don't (mostly cattle).

Janzen, M., Visser, K., Visscher, D.R., MacLeod, I., Vujnovic, D. and Vujnovic, K. Semi-automated camera trap image processing for the detection of ungulate fence crossing events. *Environmental Monitoring and Assessment* 189, 527 (2017).

Stage Two: Computer Image Processing – EventFinder

EventFinder does the same task as EventSplitter, but also removes image sets containing noise and moves the best image in a sequence to the "retained" directory. EventFinder is started with the batch file "PictureProcessorBest". This processes the images and identifies the "best" image.

PictureProcessorBest allows the user to specify the images to process, shown in Figure 9. Files are added after clicking the "Add Image Files" button. As with EventSplitter, all images selected should come from the same directory corresponding to one site (we have both K04 and K07 in our example to demonstrate different events in the next stage).


Animal Image Processor

```
K04_2017_06_25_09_31_08_1.JPG
K04_2017_06_25_09_31_09_1.JPG
K04_2017_06_25_09_31_10_1.JPG
K04_2017_06_25_09_31_11_1.JPG
K04_2017_06_25_09_31_12_1.JPG
K04_2017_06_25_09_31_13_1.JPG
K17_2017_07_06_07_18_18_1.JPG
```

Add Image Files

Instructions_FindBest2.xml

Set Instruction File

Time Diff: 30 seconds

Max Pics/Set: 50

Start

Options: -Xmx512m -Xms256m -XX:MaxPermSize=512m

Output Names: May5_EventFinder *.csv

Thanks to The Albert Conservation Association, www.ab-conservation.com,

Made at King's University Edmonton, Alberta

EventFinder next requires a xml instructions file. These instructions govern how each set of images will be processed. The instructions used in our work are in the instructions folder and named “Instructions_FindBest2.xml”.

The time difference specifies the time in seconds between images, the default being 30 seconds.

“Max Pics/Set” can be used to force a division between image sets even if the time difference between each image is less than the specified “Time Diff”.

Options increase the available memory on some Java RunTime Environments and can be left as they are.

Figure 9: Event Finder Options

Lastly, select a csv name descriptive of your image set. We usually use site name and date.

“PictureProcessorBest” must run on Windows, as it creates a batch file to process each set of images, named “ImageListToProcess.bat”. If the “PictureProcessorBest.bat” file is used to start EventFinder, then this batch file is called automatically after the “Start” button is clicked.

We have found that some computers have antivirus programs or program execution controls that may limit your ability to successfully run EventFinder. In this case you can use a different computer or use EventSplitter since EventSplitter does not require batch files to run.

EventFinder may take a while to run. We typically leave it running overnight on multiple virtual servers, each running a different instance of EventFinder. When complete, it creates a csv file in the same format as EventSplitter (since either program can create a csv file for EventTagger). A line from the csv file created is shown below:

```
C:\Users\Janzen\Desktop\EventFinderInstructions\input\K04_2017_06_25_0
9_31_08_1.JPG,C:\Users\Janzen\Desktop\EventFinderInstructions\input\K0
4_2017_06_25_09_31_13_1.JPG,6,Y,C:\Users\Janzen\Desktop\EventFinderIns
tructions\input\K04_2017_06_25_09_31_08_1.JPG,46,1,1.6666666,22309,201
7:06:25,09:31:08,09:31:13
```

Here each column is separated by a comma (as it’s a csv file) and can be opened in Excel or other program if desired. The meaning of each column is described in order as follows:

- The starting image filename in the set of images for the event
- The ending image filename in the set of images for the event
- The number of images in the event
- If an event is present, “Y” for yes, “N” for no. EventSplitter will enter “U” for unknown

- The filename of the best image in the set of images
- The average x direction of the largest fragment in the set
- The average y direction of the largest fragment in the set
- The average number of animals in the set
- The time required for EventFinder to process the set
- The date the first image was taken (the date of the event)
- The time the first image was taken
- The time the last image was taken

The best image is determined by using background subtraction. EventFinder uses all the images in the set to create a background containing no animals, and then examines the differences between each image and the background images. These pixel differences in the image are grouped into fragments. If there are no large fragments in the set, then the camera is assumed to be triggered by noise, such as wind. Otherwise, the best image is the one with the largest fragment that does not touch the edge of the image.

Images with differences covering most of the image are removed as they are considered more likely to be camera movement, sudden light changes, or an animal licking the camera "#$% (Yes, this has happened, but a large blurry tongue does not normally help classify what licked the camera)

The x and y directions of movement are determined using the largest fragment from each image. We have found this is “sort of” and “kind of” useful as it can be wrong. For example, if an animal moves left off the image and another animal moves left into the image then the largest fragment moves right.

The number of animals in the set is determined by averaging the number of large fragments in each image. This again is “sort of” and “kind of” useful as we find it is about 70% correct in determining zero, one, or more than one animal. For example, a baby overlapping a mother animal will likely be counted as one animal. Herds of animals may only be identified as more than one animal (although EventFinder will attempt to assign a number for the count).

After EventFinder runs, the csv file produced can be processed by the “PictureMoverBestDated” batch file. This moves the images based the version of EventFinder that determines the “best” image and knows which files are in each image set based on the date and time the image was taken. PictureMoverBestDated runs in the same fashion as Event Splitter Picture Mover above (although it does not start automatically). After running there should be four directories:

- input: This directory should be empty, and ready for new camera trap images
- processed: All images processed by EventFinder (either with or without an event)
- retained: The best image from each event should be copied here
- unprocessed: Images that were not processed end up here

Images in the unprocessed are any remaining images from the input directory that have not been moved to processed. They may have been moved if an image set contains less than three images, such as what can happen if one image in a three-image set is corrupt, or other reasons.

As shown with EventSplitter in Figure 8 the csv file can be moved to the directory that contains the input, processed, retained, and unprocessed directories.

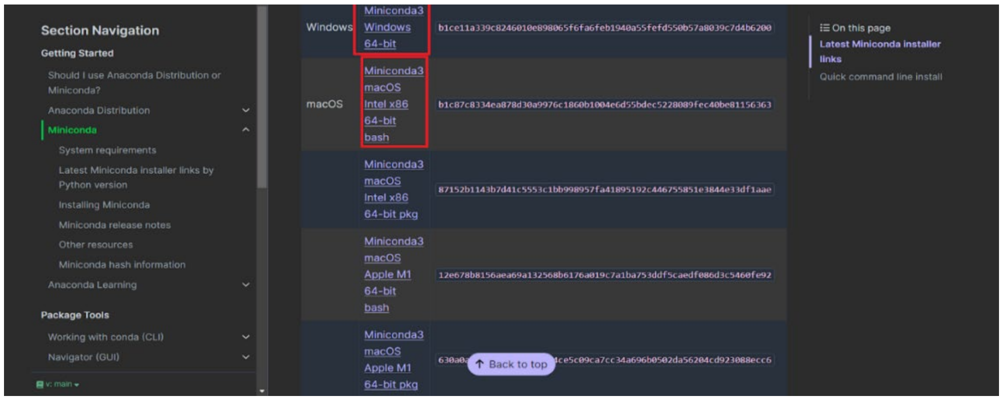
Stage Two: Computer Image Processing – EventFinderAI

EventFinderAI aims to process and divide sequences of images into image sets of events based on the time difference between sequential images. The following sets are then independently analyzed for motion events which are recorded in the outgoing CSV file. This is done using EventFinderAI's integrated software in combination with Microsoft's MegaDetector. EventFinderAI can classify an event in three ways - Human (H), Vehicle (V), or Animal (Y). Depending on the classification the image set is labeled as H, V, or Y, which allows for the user to filter visible images in EventTagger. If no event is detected the output is N, which is filtered out of EventFinderAI by default (this can be changed to ensure zero false negatives).

The initial installation of EventFinderAI is operating system dependent and is accessible on Windows, Linux, and Mac. The operating system may be selected in the main menu of the program, which will adjust the scripting used in the workflow. If the user has a Cuda compatible system and a compatible GPU, the computation can be pushed off the CPU onto the graphics card. This can allow for a faster execution time. If the program must be run on the CPU it will relay the same results, but slower processing is expected. Finally, the program must be run with the proper environments installed. Once this process is completed and the environments are properly installed, the user can run EventFinderAI indefinitely.

Conda/Environment Installation Guide

1. To use EventFinderAI the user must ensure they have access to the required Conda environments. They must first install the appropriate Conda browser at https://docs.anaconda.com/miniconda/ (Figure 1).



Windows	Miniconda3 Windows 64-bit	b1ce11a339c8246010e898065f6fa6feb1940a55fefd550b57a8039c7d4b6200
macOS	Miniconda3 macOS Intel x86 64-bit bash	b1c87c8334ea878d30a9976c1860b1004e6d55bdec5228089fec40be81156363
	Miniconda3 macOS Intel x86 64-bit pkg	87152b1143b7d41c5553c1bb998957fa41895192c446755851e3844e33df1aae
	Miniconda3 macOS Apple M1 64-bit bash	12e678b8156aea69a132568b6176a019c7a1ba753ddf5caedf086d3c5460fe92
	Miniconda3 macOS Apple M1 64-bit pkg	630a0a 4ce5c09ca7cc34a696b0502da56204cd923088ecc6

Figure 1: Conda installation webpage.

2. The user should find the installation file and execute it.

3. Once miniconda is installed the user should open the command line by searching for Terminal Command Prompt.

4. The user should type the following into the terminal (example line in Figure 3):

“Path to the scripts file of your miniconda” + \conda.exe init

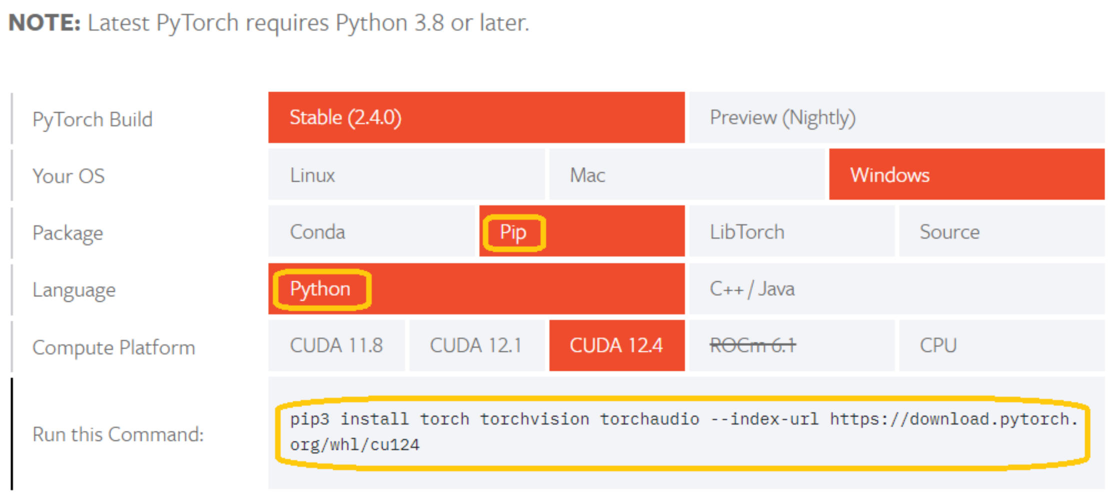
conda create -n PytorchWildlife python=3.8 -y

conda activate PytorchWildlife

pip install PytorchWildlife

pip uninstall torch torchvision torchaudio -y

The following step is dependent on the user’s hardware, go to https://pytorch.org/get-started/locally/ and fill out specific requirements, ensure you select Pip, Python, and simply copy the command.

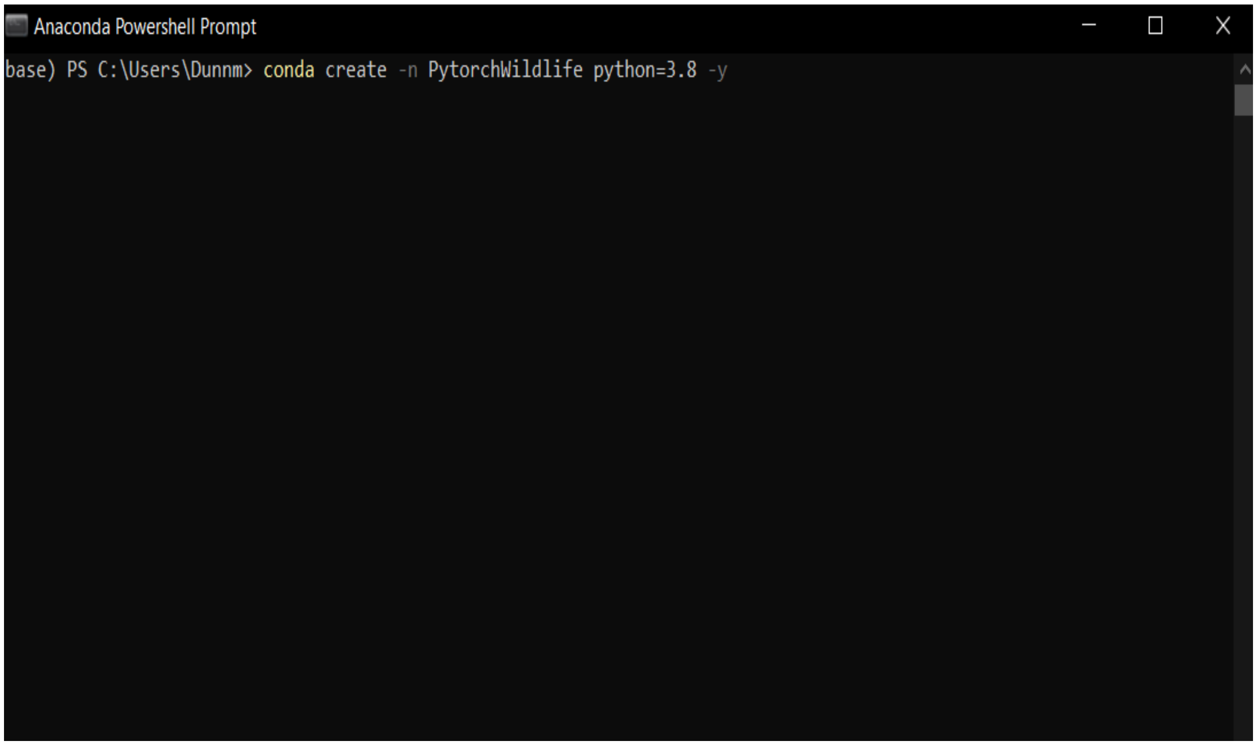


NOTE: Latest PyTorch requires Python 3.8 or later.

PyTorch Build	[✓ Stable (2.4.0)]	Preview (Nightly)			
Your OS	Linux	Mac	[✓ Windows]		
Package	Conda	[✓ Pip]	LibTorch	Source	
Language	[✓ Python]	C++ / Java			
Compute Platform	CUDA 11.8	CUDA 12.1	[✓ CUDA 12.4]	~~ROCm 6.1~~	CPU
Run this Command:	`pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu124`				

```
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.
org/whl/cu124
```

Figure 3: This website will allow the use to select their specific hardware and get the specific command they need. Note that Cuda is not supported on Mac, requiring users to rely only on their CPU.

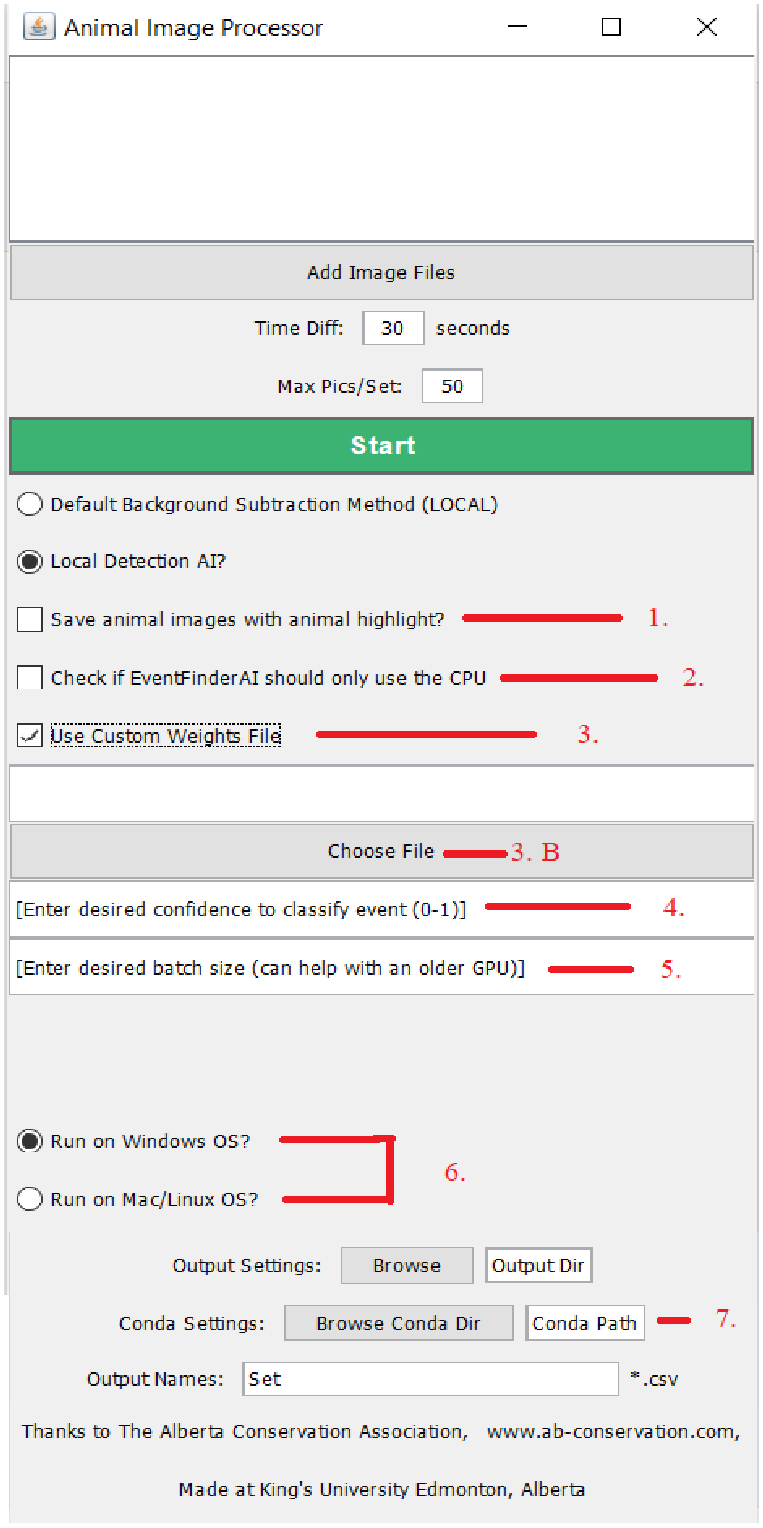


```
base) PS C:\Users\Dunnm> conda create -n PytorchWildlife python=3.8 -y
```

Figure 4: Example first line typed into terminal.

At this point the user should be able to run EventFinderAI on their GPU. If using a CPU they should check the CPU box on the EventFinderAI interface.

With this set of commands, the program will automatically activate the environment prior to running its entire workflow



Animal Image Processor

Add Image Files

Time Diff: 30 seconds

Max Pics/Set: 50

Start

Default Background Subtraction Method (LOCAL)

[✓ Local Detection AI?]

Save animal images with animal highlight? 1.

Check if EventFinderAI should only use the CPU 2.

[✓ Use Custom Weights File] 3.

Choose File 3. B

[Enter desired confidence to classify event (0-1)] 4.

[Enter desired batch size (can help with an older GPU)] 5.

[✓ Run on Windows OS?]

Run on Mac/Linux OS? 6.

Output Settings: Browse Output Dir

Conda Settings: Browse Conda Dir Conda Path 7.

Output Names: Set *.csv

Thanks to The Alberta Conservation Association, www.ab-conservation.com,

Made at King's University Edmonton, Alberta

EventFinderAI User Guide

Refer to Figure 4:

1. This batch image box will save copies of each image with the event highlighted in a box. This is storage inefficient for large image loads.

2. This box will force EventFinderAI to only use the CPU for processing and will ensure that the GPU is not used.

3. Checking this box implies the user intends on using their own CKPT file instead of the default EventFinderAI weights files.

B. Here the user specifies the path to the CKPT weights file.

4. Here the user inputs their desired confidence for classification of any given image. Higher confidence requires the network to be more certain about the prediction, meanwhile lower confidence allows for loose classifications (e.g. 0.2).

5. Here the user inputs the desired batch size for this iteration of EventFinderAI, lower batch sizes can make the computation smoother, but slower. This is a parameter that is dependent on the user's hardware.

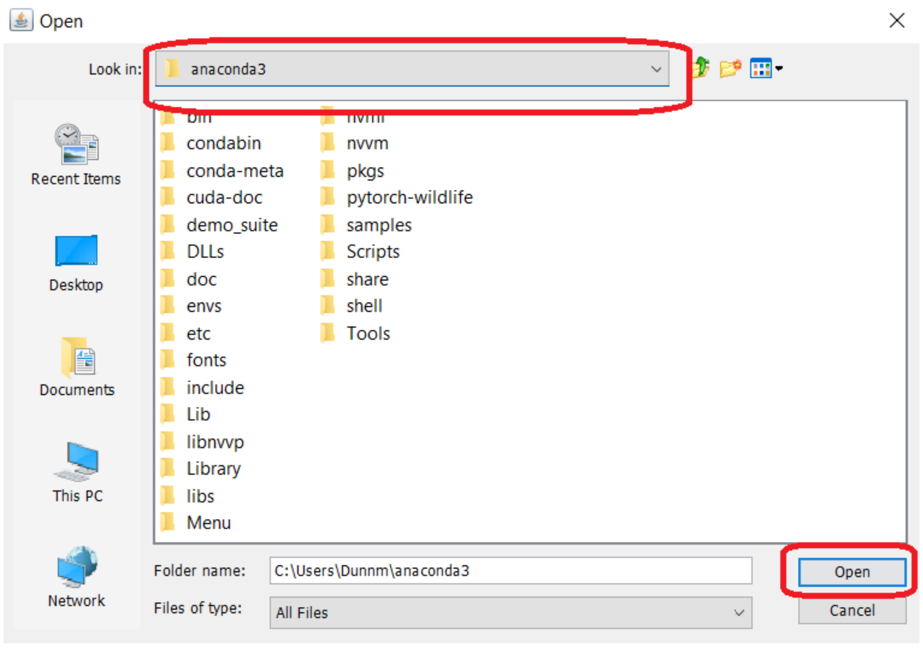
6. Here the user can select the operating system they are using. EventFinderAI will automatically assess the predicted system; however, the user can override this.

7. Here the user must find the Anaconda/Miniconda directory that they installed in the prior step. They must then navigate to this directory and select it.

Once these options are completed EventFinderAI should be ready to start.

If issues arise, here are a few common problems:

- Batch size is too high for the given hardware, causing the program to crash.
- The environment is incorrectly named or installed.
- Ensure that when selecting the Conda file:
    - If operating on Windows, the user should open the Anaconda file and select it (Figure 5).



Folder name: C:\Users\Dunnm\anaconda3
Files of type: All Files

Figure 5: Open Anaconda/ Miniconda file on Windows to select.

- If operating on Linux, the user should click on the Conda directory and then click open (Figure 6).

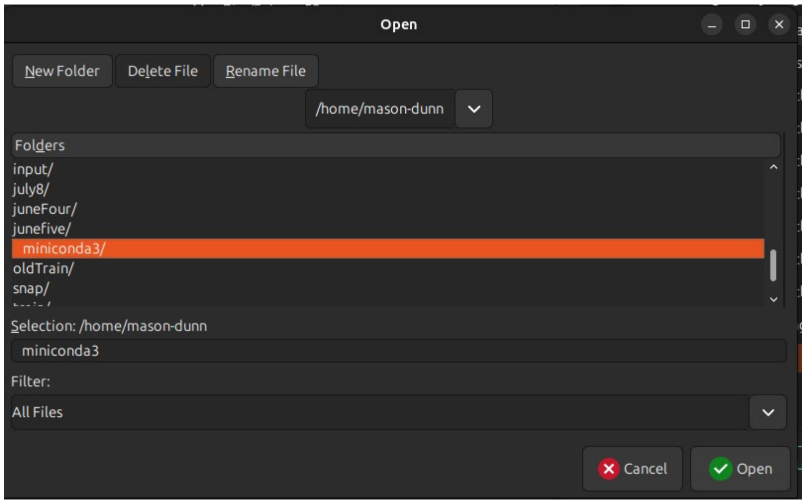


Figure 6: Select Conda directory on Mac/Linux to open.

Stage Two B: Experimental Species Classification

If the user desires to input their own weights for species classification, they can select their own .CKPT file. If this is done the user must modify the public "tagOptions.csv" and the "testLabel.csv" file. These 2 files are specific to the classes for any given network, tagOptions.csv has the classes listed in order, with their label. This will create check boxes in Event Tagger for the user to specify what images to show. Similarly, the user will want to modify the "testLabel.csv" with the following format:

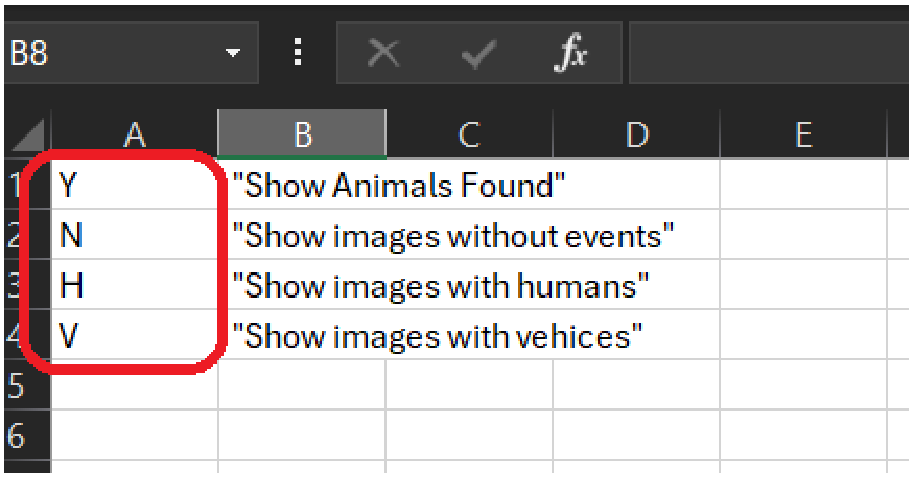
```
class_index, label

0,H

1,M

2,D

3,E

4,C
```

Here the user has the label name that the network will be classifying image sets by, as well as the corresponding output (0,1,2,3….). In this example the first output node (0) represents a human. Note that the classes should also be in numerical order.

Ensure the network is trained with:

- PlainResNetClassifier50 layer.
- Correct amount of output layers for number of classes.

Once the processing is complete the output csv file will have classified each image set with the corresponding label. To use EventTagger with the new labels the user must also modify TagOptions.csv with each of the possible output class labels (these will be the same as what was specified in the testLabel.csv file).

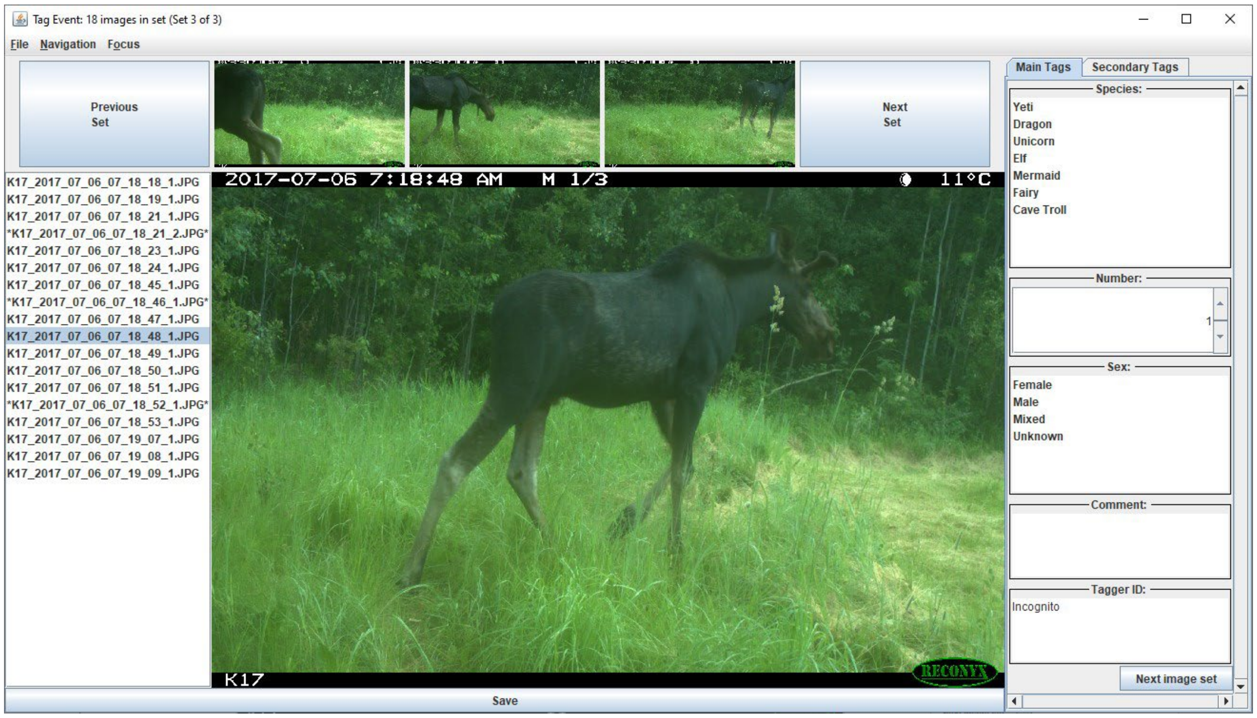


	A	B	C	D	E
1	Y	"Show Animals Found"			
2	N	"Show images without events"			
3	H	"Show images with humans"			
4	V	"Show images with vehices"			
5					
6					

Figure 7: EventTagger intakes its parameters from tagOptions.csv. All this does is change the created checkboxes in EventTagger allowing the program to display images with said label.

Stage Three: EventTagger

At this point, whether EventSplitter or EventFinder has been used, there is a csv file to process and a set of directories {input, processed, retained, unprocessed} of which the retained and processed are important. Originally, we thought that a human tagger could simply tag only the images in the retained directory, but we found they would keep a parallel workflow open in the file system to look at the processed images when the retained image did not seem to provide enough information for tagging. Hence, we created EventTagger, as shown in Figure 10, to bring this information together in one user interface, and provide customized tagging options we found difficult to use in the software we were previously using. EventTagger also assumes that events are tagged, rather than tagging individual images.



K17_2017_07_06_07_18_18_1.JPG
K17_2017_07_06_07_18_19_1.JPG
K17_2017_07_06_07_18_21_1.JPG
K17_2017_07_06_07_18_21_2.JPG
K17_2017_07_06_07_18_23_1.JPG
K17_2017_07_06_07_18_24_1.JPG
K17_2017_07_06_07_18_45_1.JPG
K17_2017_07_06_07_18_46_1.JPG
K17_2017_07_06_07_18_47_1.JPG
K17_2017_07_06_07_18_48_1.JPG
K17_2017_07_06_07_18_49_1.JPG
K17_2017_07_06_07_18_50_1.JPG
K17_2017_07_06_07_18_51_1.JPG
K17_2017_07_06_07_18_52_1.JPG
K17_2017_07_06_07_18_53_1.JPG
K17_2017_07_06_07_19_07_1.JPG
K17_2017_07_06_07_19_08_1.JPG
K17_2017_07_06_07_19_09_1.JPG

Number: 1

Tagger ID: Incognito

Figure 10: EventTagger User Interface

Various components of EventTagger are shown in Figure 10. If EventFinder was used to create the csv file, then initially the large image in the middle is of the window is the best image as determined by EventFinder. If EventSplitter was used to create the csv file, then the large image is initially the first image in the set. Clicking on the large image toggles the zooms in to full size or zoom back to show the entire image. The three small images at the top provide a quick context of the event and each comes from a different third of the images (first third, second third, last third). One of these images will be initially displayed, and the other two are randomly selected.

The list of images on the left include the names of all the images in the set. Clicking, or traversing them using arrow keys, changes the large image displayed. The image file names with asterisks around them indicates they are one of the three small images in the top of the window.

The two buttons “Previous Set” and “Next Set” allow the user to move to difference events. There is also a “Jump to Set” by number from the Navigation Menu near the top of the frame. For quick use while tagging, there is an additional “Next image set” at the bottom of the tagging pane.

The tagging can be done by selecting options or entering text in the pane on the right-hand side. If using a keyboard, pressing Tab (or Enter) will move to the next tag. The “Next image set” button can be “clicked” from the keyboard by pressing the spacebar. If an option is selected, it can be deselected by clicking on the selected option with the mouse pointer. The user can see additional tags by clicking on the “Secondary Tags” pane.

Saving can be done either by clicking the “Save” button at the bottom of the Window or selecting the “Save” option from the File menu. When saving, two files are written: a tagged csv file and a project file. The csv file has the additional tags appended to the original csv file, and the project file allows the tagger to resume tagging an image set later. Default names are suggested for each based on the original csv file, but the user may select a different name if desired.

Accelerators enable quick navigation around the interface. They are listed in the menu and are as follows:

- CTRL-S: Save
- CTRL-P: Go to previous event
- CTRL-N: Go to the next event
- CTRL-M or CTRL-1: Go to the main panel with the focus set to the first tag
- CTRL-2, CTRL-3: Go to the secondary or tertiary panel if available
- CTRL-I: Set the focus on the list of images on in the left of the window

For example, the user can quickly look through the image set with CTRL-I and then using the up and down arrow keys to change the large image.

Exiting can be done either by clicking the "x" in the upper right corner or by selecting "Exit" from the file menu. The user is asked to confirm to exit and are reminded of unsaved tagging changes.

If not all the features appear, it may be that you need to increase the size of the window. The layout will be determined by Java's layout manager, but the example in Figure 11 is too small to be useful.

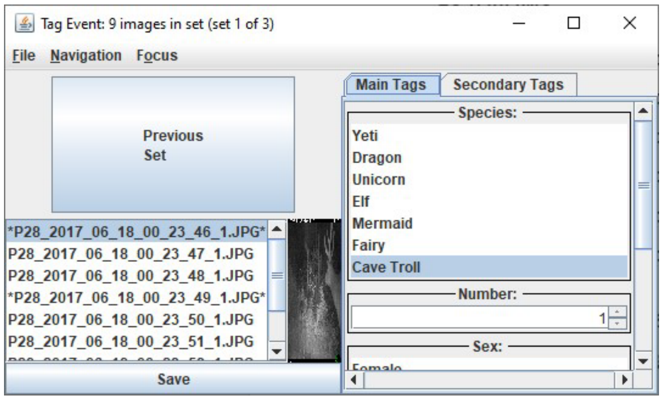

Tag Event: 9 images in set (set 1 of 3)
File
Navigation
Focus
Previous Set
Main Tags
Secondary Tags
Species:
Yeti
Dragon
Unicorn
Elf
Mermaid
Fairy
Cave Troll
Number:
1
Sex:
P28_2017_06_18_00_23_46_1.JPG
P28_2017_06_18_00_23_47_1.JPG
P28_2017_06_18_00_23_48_1.JPG
P28_2017_06_18_00_23_49_1.JPG
P28_2017_06_18_00_23_50_1.JPG
P28_2017_06_18_00_23_51_1.JPG
Save

Figure 11: EventTagger with a too small window

If an image set is displayed without a detected event, then the large window will initially be blank until the tagger selects an image to display (also the window title will change). This only happens when EventFinder creates the csv file.

The tags presented to the user are customizable, with a default arrangement in TagList.xml. You may modify this file or create a new xml file and select it when EventTagger starts. An abbreviated xml tag list file used to create the tags in Figure 10 is presented below.

```
<TagList>
        <Tag>
                <CategoryName>Species</CategoryName>
                <Meaning>Species that was identified in the sequence</Meaning>
                <OptionName>Yeti</OptionName>
                <OptionName>Dragon</OptionName>
                <Priority>main</Priority>
        </Tag>
```

```
<Tag>
        <CategoryName>Number</CategoryName>
        <Meaning>Number of individuals in the sequence</Meaning>
        <Number>
                <Minimum>0</Minimum>
                <Maximum>42</Maximum>
                <Increment>1</Increment>
        </Number>
        <IndividualCountTag>true</IndividualCountTag>
        <Priority>main</Priority>
</Tag>
<Tag>
        <CategoryName>Sex</CategoryName>
        <Meaning>Gender of Animals in Image</Meaning>
        <OptionName>Female</OptionName>
        <OptionName>Male</OptionName>
</Tag>
<Tag>
        <CategoryName>Comment</CategoryName>
        <Meaning>Additional Notes from the Tagger</Meaning>
        <TextField>true</TextField>
        <Priority>main</Priority>
</Tag>
<Tag>
        <CategoryName>
            Behaviour
        </CategoryName>
        <Meaning>Wildlife behaviour in sequence</Meaning>
        <OptionName>Skiing</OptionName>
        <OptionName>Cooking</OptionName>
         <OptionName>Queueing</OptionName>
        <OptionName>Quantity Surveying</OptionName>
       <Priority>                              secondary
```

```
                    </Priority>
            </Tag>
    <Tag>
      <CategoryName>Direction</CategoryName>
      <Meaning>Average direction of movement</Meaning>
      <OptionName>Left</OptionName>
      <OptionName>Right</OptionName>
      <OptionName>Neither</OptionName>
      <DirectionTag>true</DirectionTag>
      <Priority>secondary</Priority>
    </Tag>
    <Tag>
      <CategoryName>Temperature</CategoryName>
      <Meaning>Current temperature</Meaning>
      <Number>
            <Minimum>-40</Minimum>
<Maximum>42</Maximum>
            <Increment>1</Increment>
      </Number>
      <TemperatureTag>true</TemperatureTag>
      <Priority>secondary</Priority>
    </Tag>
    <Tag>
      <CategoryName>Tagger ID</CategoryName>
      <Meaning>Identification of person who tagged this image set</Meaning>
      <TextField>true</TextField>
      <TaggerIDTag>true</TaggerIDTag>
      <Priority>main</Priority>
    </Tag>
</TagList>
```

The indentation is optional, but helpful to make it more human readable. As before, each element in the xml file has a matching opening and closing tag. The entire tag list begins with a <TagList> and closed with the corresponding </TagList>. Similarly each tag begins with a <Tag> and ends with a </Tag>. Each tag has a category name, which is the title of the tag in the interface and the title of the column in a saved csv file when this option is selected. The meaning is the text displayed when the tagger hovers over the tag title with the mouse.

As shown in the example Taglist file, there are three types of tags available.

- The default options tag. Here there is a list of options, each in a <OptionName> tag of which the user can select one.
- The text tag. Here there is a <TextField> tag with a value of true indicating that the user can write text in this tag. There are no options listed when <TextField> is true. Please note that commas are not permitted when tagging a text field, since it will be saved into a csv file

presenting confusion in determining columns. If the tagger enters a comma in this field they will be prompted to change it to a <comma> instead.

- The number tag. This is similar to a text field, but the user can only enter numbers. Inside the <Number> tag are the <Minimum>, <Maximum>, and <Increment> options for the lowest possible number, the highest possible number, and the amount the number is permitted to change. For a number tag in the interface the tagger can either type the number, use the up and down arrow keys on the keyboard, or click the up and down arrows that appear. Please note that if the tagger types the number then they must press enter to confirm the entry. Text that is not numeric, or outside the range reverts to whatever was previously in the number tag. Currently EventTagger only handles integer numbers. If fractions or decimals are required please specify a text field instead, which allows arbitrary text.

The priority tag specifies which pane to place the tag. "main" puts the tag on the main panel. "secondary" places the tag on the secondary panel. Specifying "tertiary" puts the tag on a third panel. Please note that the tertiary pane only appears if there is at least one tag on it. The main and secondary panels always appear. If no priority is specified, then main is assumed. It is our intention that tags that must be filled in for every event should be placed on the main pane. Tags that are appropriate only sometime should go on the secondary pane. This, however, is not enforced and users are permitted to arrange the tags as they see fit. Please note that the order the tags appear in the tag list determine the order of the columns in the tagged csv file.

There are several special xml tags that enable additional behaviour in EventTagger, such as automatically filling in a field. A unique special xml tag should appear only once in an xml tag list file. For example, there should not be two temperature tags. These special tags are as follows:

- <IndividualCountTag>: When set to true this indicates that this tag is the number of individuals in the event. EventTagger will use the average number of individuals from EventFinder to initially fill in this field.
- <DirectionTag>: When set to true this tag is initially filled in with the direction of the animal based on the average x-direction from EventFinder. This must be an options tag with the first option means left, the second option means right, and the third option means neither. Additional options may be specified after these, but these additional options will never be selected automatically by EventTagger. The tagger may, however, manually select these options while tagging.
- <TemperatureTag>: When set to true this tag is the temperature according to the first image in the set. This tag can either be a text field or a number. As described below, EventFinder will search the image for the temperature so this option will work when either EventSplitter or EventFinder has been used to create the csv file.
- <TaggerIDTag>: when set to true this tag is the name or identity of the person doing the tagging. This is set in the options when EventTagger starts and will copy the tagger ID whenever a tag is changed in the set. Thus, one tagger can begin tagging and another can load the project later to continue tagging. The TaggerIDTag tag will contain the identity of the tagger that last made a change to the event.

As already indicated, an options window appears when EventTagger is first started, similar to the one shown in Figure 12. These allow the tagger to start a new project, by clicking the "Select CSV File" or continue from a saved project, by clicking the Continue button.

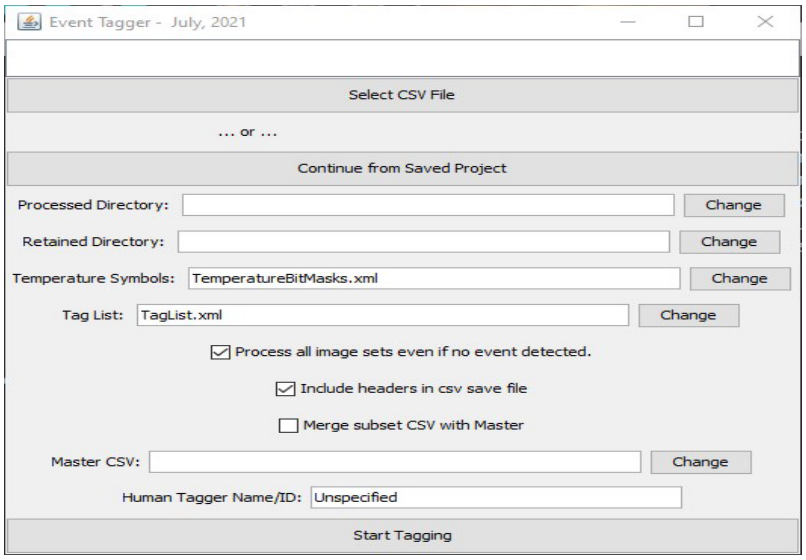

Event Tagger - July, 2021
Select CSV File
... or ...
Continue from Saved Project
Processed Directory:
Change
Retained Directory:
Change
Temperature Symbols: TemperatureBitMasks.xml
Change
Tag List: TagList.xml
Change
☑ Process all image sets even if no event detected.
☑ Include headers in csv save file
☐ Merge subset CSV with Master
Master CSV:
Change
Human Tagger Name/ID: Unspecified
Start Tagging

Figure 12: EventTagger options window

When starting from a CSV file (new project) the processed and retained directories may be automatically filled in. The paths of the processed and retained directories are determined in two possible ways. In the first method EventTagger looks at the path for the first image in the csv file. EventTagger assumes that the file has been moved from "input" to a "processed" directory and fills in that path if such a directory exists. The retained directory is similarly assumed.

If the processed or retained directory from the first image in the csv file no longer exists, then EventTagger assumes the files have been moved from their original location. In this case EventFinder assumes that the processed and retained directories are subdirectories of the directory that contains the csv file. If such directories exists, then the paths will be automatically filled in. If neither of these methods work, then the tagger is left to select the locations of the directories manually.

When continuing from a saved project, the locations of the csv file, the processed directory, and the retained directory are recorded in the saved project file. Similarly, the option to process all image sets, including headers in the csv file, and the tagger identification are also written to the project file. The entire tag list is written to the project file so a tagger cannot change the tags when continuing from a saved project, as this could cause misalignment in the csv columns since different events would have different tags / columns.

When the "Process all image sets event if no event detected" checkbox is clicked then all image sets will be present for tagging, regardless of whether EventFinder detected an event or not. When unchecked, only events detected by EventFinder are presented for tagging. If EventSplitter was used to create the csv file then all image sets are presented to the tagger since EventSplitter does not detect events. The tagger can select whether to include headers in the tagged csv file using the "Include headers in csv save file".

EventTagger can automatically record the temperature when the <TemperatureTag> is set to true for one of the tags. To determine the temperature EventTagger examines the white pixels in the first image of an image set. If the pixels at a location match a bit pattern in the "Temperature Symbols" file, then the temperature is filled in for the corresponding tag. The file "TemperatureBitMasks.xml" contains the location and bit masks for symbols for two Reconyx models of camera. You may create a new bit mask for other cameras should you want.

Re-Tagging Previously Tagged data from different Image Files (new for September 2021)

Event Tagger now supports opening a tagging session from a csv that contains images from multiple image folders. Before, each tagging session was only for one set of images. With this added feature, you can now load up events that were previously tagged in different sessions. Say you want to view all previously tagged events of Moose that you put into one csv file. You can put that into EventTagger using the "Select CSV File" option. EventTagger will detect you are trying to load events from multiple files and will tell you which file to start with in a pop-up alert. Using the "Change" button on the processed directory field, navigate to that directory.

EventTagger will assume that all your image folders are all in the same parent directory. For example:

C:/EventFinderSuite/Images/RR01_2018-07-18_2018-08-13/processed and
C:/EventFinderSuite/Images/SB02_2018-08-08_2018-08-14/processed would both be in the Images folder. If they are not, this feature will not work. From here you can start tagging. Saving and continuing from a saved project work the same as before on these subset csv's.

If you keep a master CSV of all you tagged data, you can also choose the option to merge the changes that you made to your subset data back into your master CSV. Check the "Merge subset CSV with Master" box and using the "Change" button right below that to find your master CSV file. When you are done tagging and want to merge, click File > Merge in the menu bar. By default, this will create a new master file. If you don't check the merge box in the opening options, the merge option will be disabled in the menu bar.

Setting the bitmask for EventTagger to automatically determine the temperature

The start of a temperature bit mask file is shown below. As with other files it is specified using xml to increase its human readability.

```
<TemperatureImage>
  <CameraBrand>RECONYX</CameraBrand>
  <CameraName>HC500 and PC800</CameraName>
  <ErrorsAllowed>3</ErrorsAllowed>
  <Location>
    <Name>Sign HC500</Name>
    <X>1729</X>
    <Y>0</Y>
    <Width>30</Width>
    <Height>30</Height>
  </Location>
```

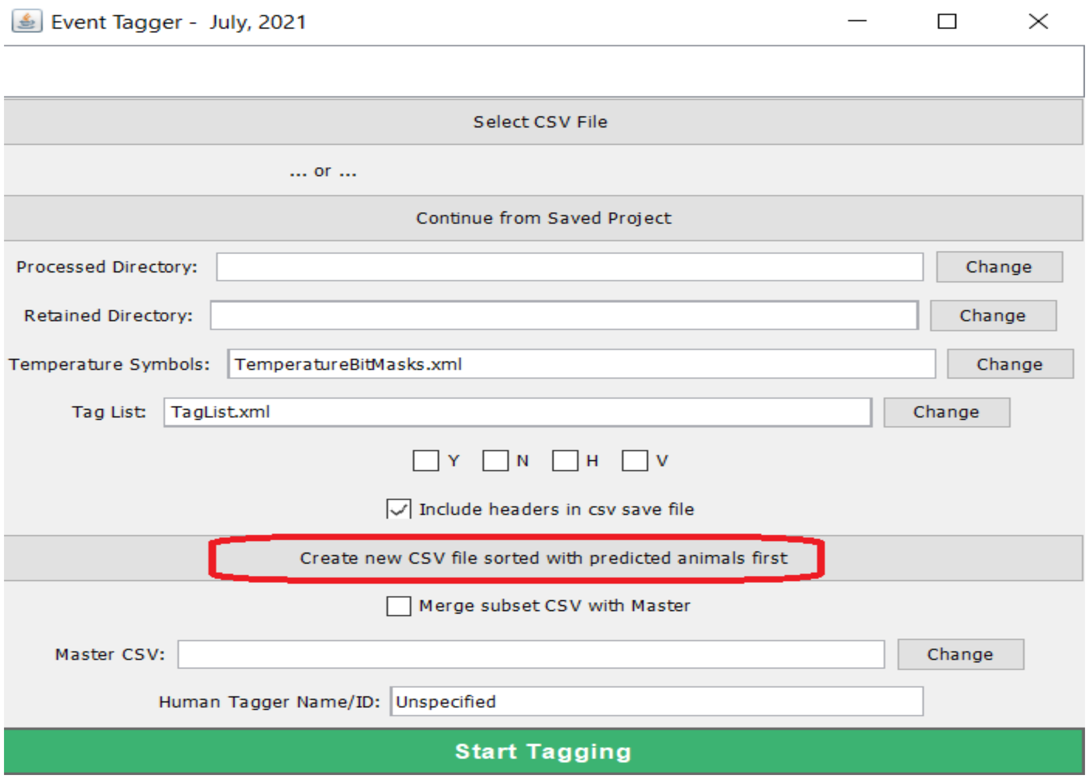
The <CameraBrand> and <CameraName> are primarily for humans to know to which models this file corresponds. The <ErrorsAllowed> indicate how many pixels can be wrong, and the bit mask still considered to match. Each <Location> indicates an area in the image to search. For example, the <Location> in the example indicates where to look for the negative sign with the HC500 model of camera.

For each location EventTagger will match against a set of potential symbols. An example is shown to the right. Notice that the 1’s indicate where there should be white pixels in a bit mask match while the 0’s indicate black pixels. If the white and black pixels at the image location match the bit mask pattern, then the temperature will be written to the tag (in this case a 4).

```
<Symbol>
    <Name>4</Name>
    <Pattern>
      000000000000000000000000000000
      000000000000000000000000000000
      000000000000000001111110000000
      000000000000000001111110000000
      000000000000000111111110000000
      000000000000000111111110000000
      000000000000011111111110000000
      000000000000011111111110000000
      000000000001111111111110000000
      000000000001111111111110000000
      000000000111111001111110000000
      000000000111111001111110000000
      000000011111100001111110000000
      000000011111100001111110000000
      000001111110000001111110000000
      000001111110000001111110000000
      000111111111111111111111111110
      000111111111111111111111111110
      011111111111111111111111111110
      011111111111111111111111111110
      011111111111111111111111111110
      011111111111111111111111111110
      000000000000000001111110000000
      000000000000000001111110000000
      000000000000000001111110000000
      000000000000000001111110000000
      000000000000000001111110000000
      000000000000000001111110000000
      000000000000000000000000000000
      000000000000000000000000000000
    </Pattern>
  </Symbol>
```

Stage Three B: Integrating EventFinderAI with EventTagger

EventTagger allows the user to query image sets based on the assignment applied by EventFinderAI's image classifications. If the user prefers to have all images present, they can select every box, however any permutation of images /image checkboxes can be selected. By default Y = animal, H= human, V= Vehicle, N= no Event. In some instances, the user may prefer to tag with no image sets being removed (to prevent false negatives). In this case the user can opt to select "Create new CSV file sorted with predicted animals first" which will display predicted animals first, making the image tagging process smoother.


Event Tagger - July, 2021

Select CSV File

... or ...

Continue from Saved Project

Processed Directory: Change

Retained Directory: Change

Temperature Symbols: TemperatureBitMasks.xml Change

Tag List: TagList.xml Change

☐ Y ☐ N ☐ H ☐ V

☑ Include headers in csv save file

Create new CSV file sorted with predicted animals first

☐ Merge subset CSV with Master

Master CSV: Change

Human Tagger Name/ID: Unspecified

Start Tagging

Figure 8: EventTagger sort option which creates a new sorted csv file with images sorted with predicted animals first.

The checkboxes are labeled with the corresponding class, which can be modified in "TagOptions.csv" (only change if you have custom weights files).

Conclusion

We have found the EventFinder Suite simplifies our Camera Trap Image processing and we hope you find it useful too. CameraTransfer can manage transferring photos from camera trap memory cards by site name or can be used to transfer from a directory of multiple cameras from the same site. Images are automatically renamed to avoid duplicate names from a memory card's directories, or from multiple copies of the memory card. EventSplitter breaks the images into sets consisting of a single event, based on the time difference between subsequent images. EventFinder goes further than EventSplitter and also attempts to determine the best photo to initially present to a human tagger, as well as remove image sequences that do not contain an event. EventFinder also attempts to identify the number of animals, and the direction of movement in image sets, but currently requires additional human verification. EventTagger present the image sets in a convenient layout. Ideally the image set can be tagged all on one screen using the large image to determine important details, and the three small images to set the context. The remaining images in a set can be quickly selected by clicking on the list or using the arrow keys to traverse the images. Tags completed for every image set can be placed on the main pane, while tags used less frequently can be placed on the second or third pane. Accelerator keyboard keys enable quick traversal through the interface, and projects enable taggers to stop and return to their work later. The use of an xml tag list file enables customization of the tags for a project, and special flags in this file enables EventTagger to automatically fill in some information. Flagging a tag as a tagger identification allows marking of who tagged an image set and updates based on image set (event) rather than project based on who last tagged an event.

While there is required no cost for using the EventFinder suite we ask that you notify us if you use this in your work. This will help us determine the value of continued input into the suite for its use to ecologists, academic researchers, and others. We ask that publications resulting from research that uses The EventFinder Suite add us as an acknowledgement in their publication or reference our paper describing EventFinder[2].

Sincerely,

Michael Janzen, Michael.Janzen@kingsu.ca

The EventFinder Suite, made at The King's University, Edmonton, Alberta, Canada

Document last updated ~~May 13, 2020~~ ~~September 2021 (David Fountain)~~, Aug 19 2024 (Mason Dunn)

[2] Janzen, M., Ritter, A., Walker P.D. and Visscher D.R. EventFinder: a program for screening remotely

captured images. *Environmental Monitoring and Assessment* **191,** 406 (2019).