

The EventFinder Suite

Updated June 2026 with EventFinderAi and Neural Network Classification.

Overview

EventFinder is a suite of programs made at The King’s University (Edmonton, Alberta, Canada) to manage images from camera traps from collection to classification. We find 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, MacOS and Linux operating systems and uses Java (and Python for EventFinderAI).

EventFinder 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 CameraTransfer 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 (some of which can also be automatically classified by species – see EventFinderAi section), and the “noise” class which means that none of the above listed classes was detected. The third stage is the human classification. EventTagger uses the camera trap images and the csv file produced in phase two to present an interface to the user with customizable tags.

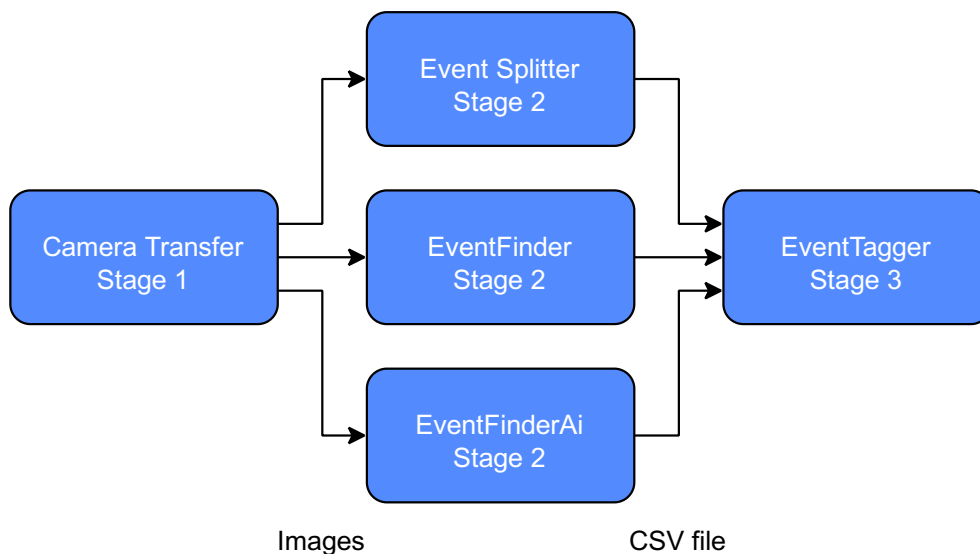


Figure 1: Workflow of the EventFinder Program

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 events¹. 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.

The EventFinder Suite is free to use. Portions of its creation have been financially 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



Figure 1: Sample Reconyx Image named K20 after King’s site 20.

the site name as the camera name works well with Reconyx 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

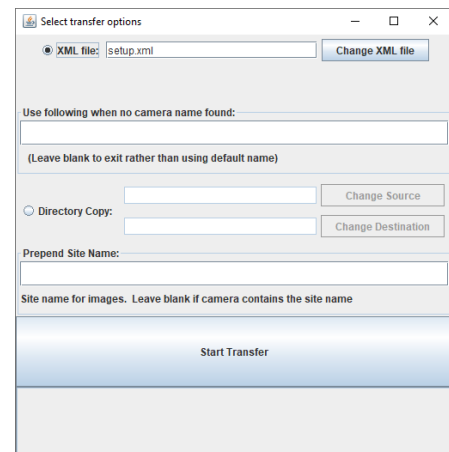


Figure 2: CameraTransfer Options

¹Janzen, 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).

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>
</cameraTransfer>
```

```
<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 3: 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.

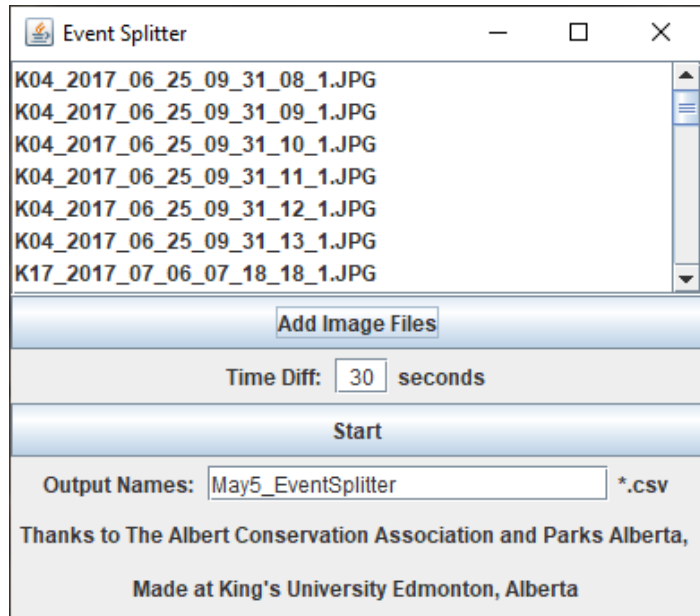


Figure 4: EventSplitter

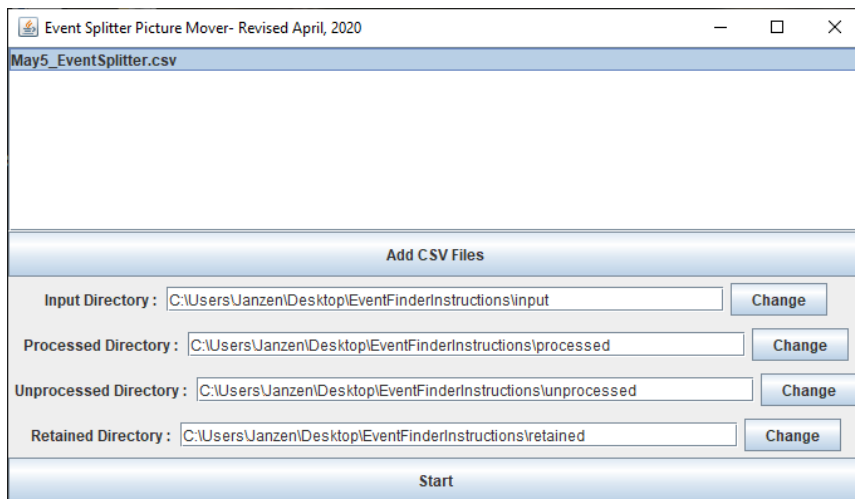


Figure 5: Event Splitter Mover program

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

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.

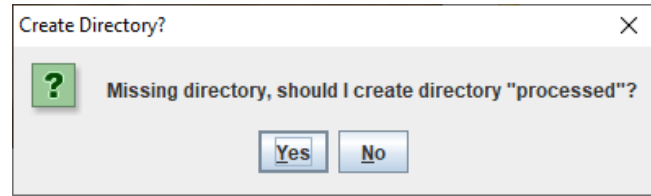


Figure 6: Prompt to create a directory

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.

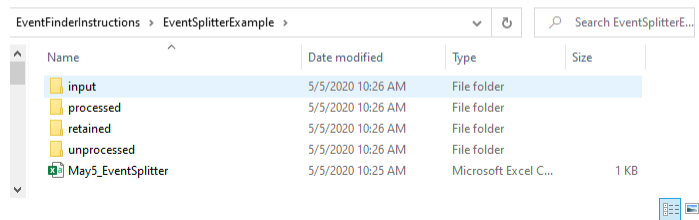


Figure 7: Directory after running EventSplitter

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).

EventFinder was the earliest of the programs written and has been adapted for different purposes. The “PictureProcessor” 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).

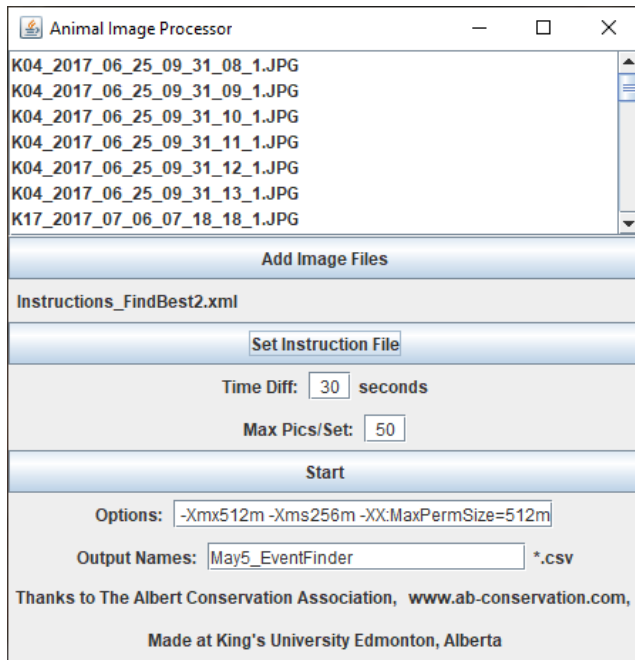


Figure 8: Event Finder Options

“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_09_31_08_1.JPG,C:\Users\Janzen\Desktop\EventFinderInstructions\input\K04_2017_06_25_09_31_13_1.JPG,6,Y,C:\Users\Janzen\Desktop\EventFinderInstructions\input\K04_2017_06_25_09_31_08_1.JPG,46,1,1.66666666,22309,2017:06:25,09:31:08,09:31:13
```

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

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.

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

- 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

These conda and Pytorch Wildlife installation instructions are up to date as of August 2025.

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).

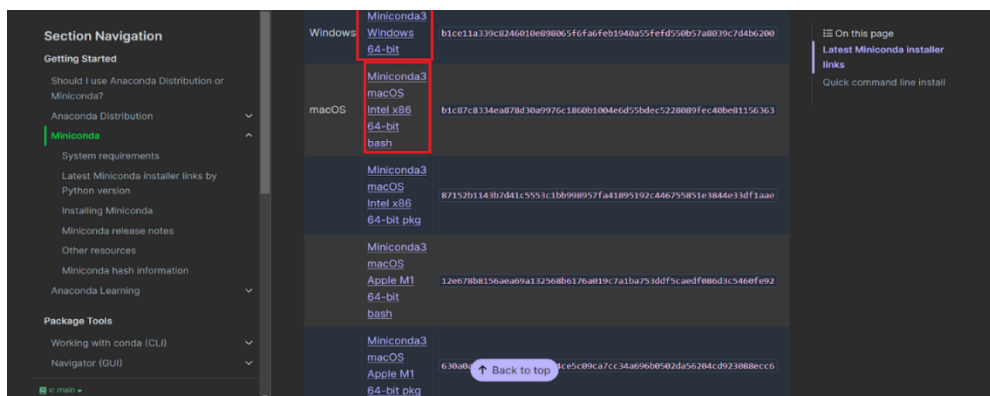


Figure 10: Conda installation webpage.

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==1.0.2.15
```

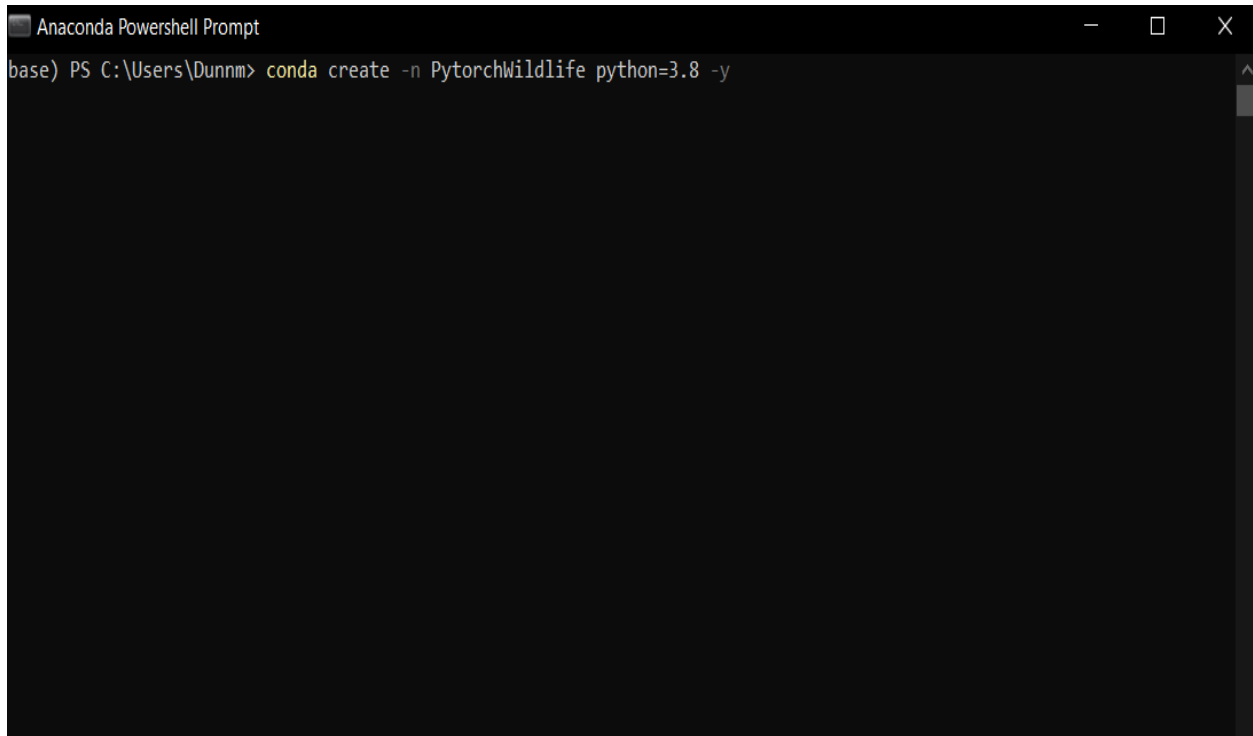
```
pip uninstall torch torchvision torchaudio -y
```

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

Figure 11: 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 12: Example first line typed into terminal.

UPDATED:

- On some Windows computers, conda virtual environment may not be accessible through a default terminal, in that case, search for “Anaconda Prompt” app that was installed with conda, it will play the role of a terminal with conda environment.
- **IMPORTANT:** make sure Java is also installed on your computer. If not, download it from the official website.

Collect the images you would like to process into a folder and name it exactly “input” (all lowercase). This will help the program to automatically identify the input folder later. It is recommended to navigate to folder “imgs” and then “input” in the EventFinder folder that was extracted from the zip archive and load your images in that “input” folder. This will help EventFinder to determine most of your folder paths automatically.

At this point the user should be able to run EventFinderAI. To do that, navigate to the EventFinder folder that was extracted from the zip archive, right click on it, select “open new terminal in folder” (if using Anaconda Prompt, simply navigate to that folder using terminal command “cd DIRECTORY_NAME”).

Depending on your operating system, write in the terminal:

```
.\EventFinder_Windows.bat
```

or

```
./EventFinder_macOS.sh
```

(Important: use backslash (“\”) for Windows and forward slash (“/”) for MacOS and Linux)

This will start EventFinderAi with its first stage: Animal Image Processor (Picture Selector).

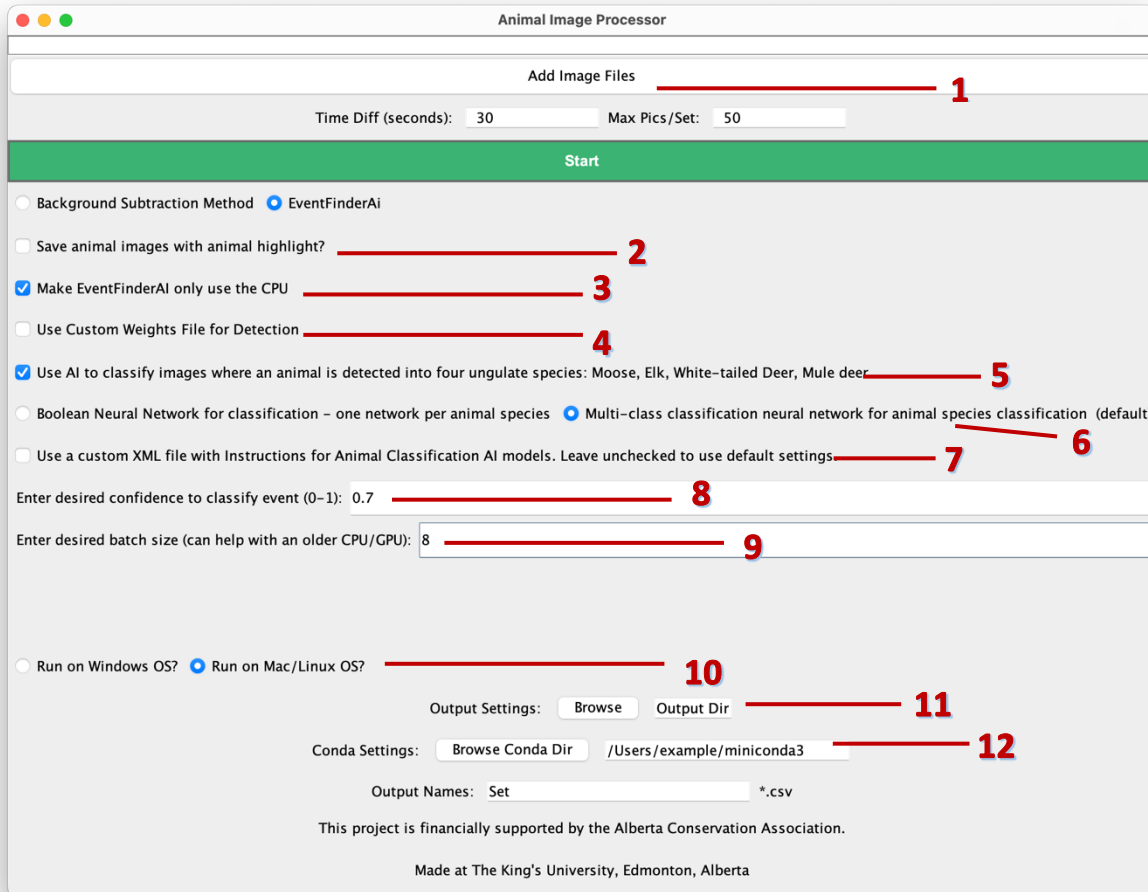


Figure 13: Settings for Animal Image Processor (Picture Selector).

1. To use EventFinderAi, click on “Add Image Files,” in the opened window, navigate to your “input” folder and select images that you wish to process.
2. 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.
3. This box will force EventFinderAI to only use the CPU for processing and will ensure that the GPU is not used.
4. Checking this box implies the user intends on using their own CKPT file instead of the default EventFinderAI weights files.
5. **NEW:** Checking this box enables the Classification Neural Network pipeline that takes sequences where an animal was detected and classifies them into species (currently: Moose, Elk, Mule Deer and White-Tailed deer).
6. **NEW:** Selection between a Classification Neural network (one network that classifies all four species), or Boolean Networks, one network trained for each species. Classification network is

- more accurate; however, the user can select Boolean if they wish to look for only one species (i.e. only Moose) in the images, or if they trained their own Boolean network for a different species (i.e. for Coyotes) and would like to add it into the structure which can be done via the XML file.
7. Check this box if you would like to replace our XML file for neural network structures with your own. You can look at XML files in the project directory to follow the structure, as well as modify it to add your networks to it, or to deactivate our networks out of the Boolean structure by deleting their <species> blocks.
 8. Here the user inputs their desired confidence for classification of any given image (default and the most optimal is 0.7). Higher confidence requires the network to be more certain about the prediction, meanwhile lower confidence allows for loose classifications.
 9. 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 (most modern computers allow for batch sizes of 8 and higher, while on older machines it is better to stay with 1).
 10. Here the user can select the operating system they are using. EventFinderAI will automatically assess the predicted system; however, the user can override this.
 11. Here the user must select the output folder. It is recommended to use the folder "imgs" in the EventFinder directory which has "input" folder in it to keep all the data organized.
 12. 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 it is selected, EventFinder will remember the location for the future uses of the software, however it can always be overridden if the user selects a new location for conda (it is not recommended to change conda location as it may introduce bugs).

Once these options are completed EventFinderAI should be ready to start. Click "START" button and wait for Ai models to complete animal detection and classification. You will be able to see the updates in the Terminal window.

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 the user selects the "miniconda3" folder.
- **IMPORTANT:** Ensure that all of the names of directories in the path to EventFinder or miniconda or images (anything that EventFinder has to work with) contain **ABSOLUTELY NO** whitespaces (i.e. use names "myFolder" or "my_folder" and NOT "my folder").
- As of August 2025, the version of Python must be 3.8 and version of Pytorch Wildlife must be 1.0.2.15 (when writing the command for installation, a double equal sign must be used (the full command should be: "pip install PytorchWildlife==1.0.2.15").

Once detection and classification are complete, a new window will open (see Figure 14). Ai created a csv file with the output of its detection and classification with the name specified in the previous step (default: "Set.csv"). Once you click on "Add CSV Files" a directory will open in file browser, simply sort it by time created and you will see the csv file at the top of the list, select it. All the other fields should be selected automatically if you select a valid csv, and your input folder was originally named "input". If automatic selection does not happen, browse to manually select all four directories for arranging your images. Click start once all fields are filled out.

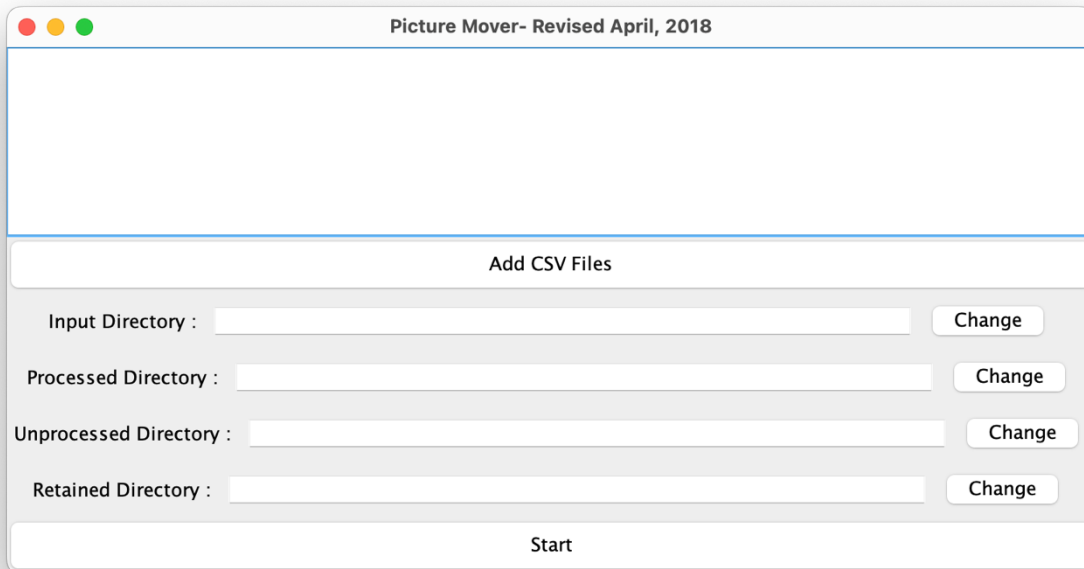


Figure 14. Picture Mover Settings.

Once pictures are moved. EventTagger will be automatically opened (next stage).

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 {processed, retained}. 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 15, 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.

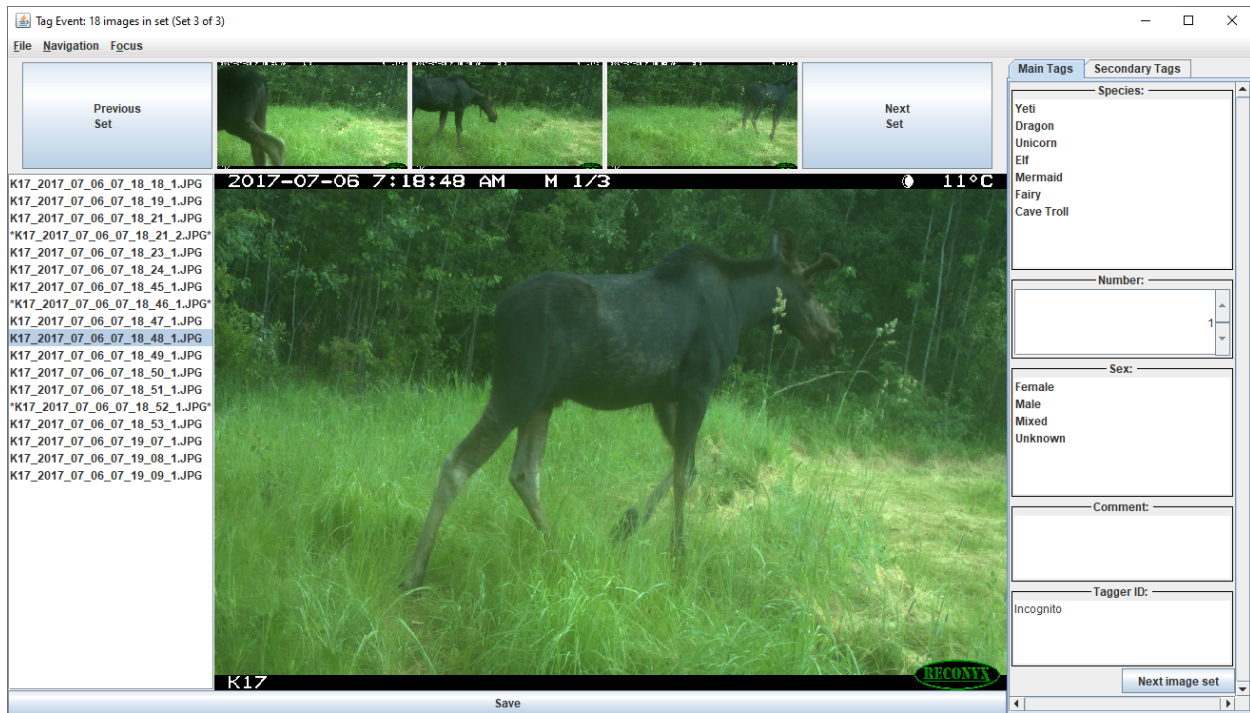


Figure 15: EventTagger User Interface

Various components of EventTagger are shown in Figure 15. 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.

NEW: if you used EventFinderAi for classification, animal species as well as the number of individuals can be pre-selected for some of the sequences and the user can simply review it and correct as necessary.

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 16 is too small to be useful.

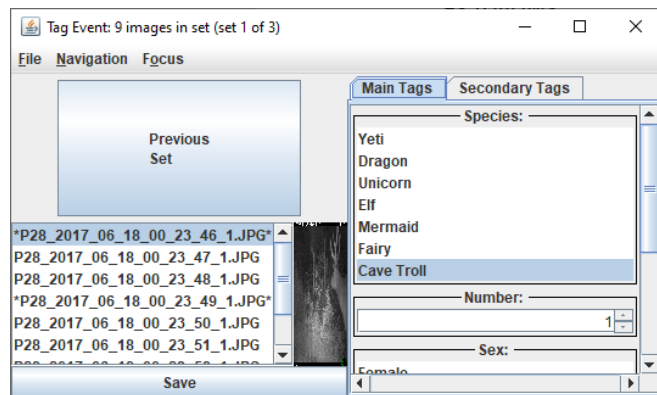


Figure 16: EventTagger with a too small window

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. When using EventFinderAi, “species” must remain as the first tag in the XML. An abbreviated xml tag list file used to create the tags in Figure 17 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>
<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>

```

Figure 17: An abbreviated xml tag list file used to create the tags.

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 18. 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.

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.

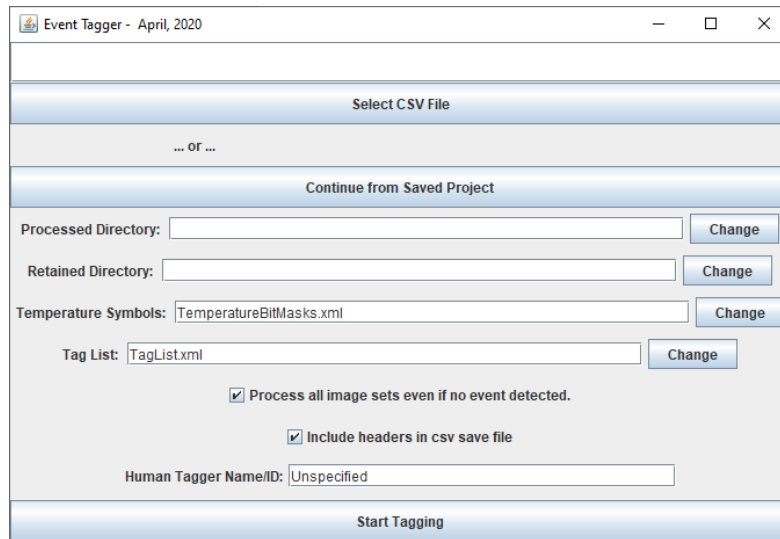


Figure 18: EventTagger options window

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. EventFinderAi uses Microsoft's MegaDetector to detect animals in images and our neural networks based on PytorchWildlife to classify them into four ungulate species in Alberta, Canada: Moose, Elk, White-Tailed Deer, and Mule Deer. EventTagger presents 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 no cost required 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 EventFinder².

Sincerely,

Michael Janzen, Michael.Janzen@kingsu.ca

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

Document last updated:

June 11, 2026 (Aleksei Antoniukov)

Previous updates:

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).

This project is financially supported by the Alberta Conservation Association.

<https://www.ab-conservation.com/>

