Ecological Informatics 24 (2014) 11–16

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecolinf

Software for minimalistic data management in large camera trap studies

Yathin S. Krishnappa *, Wendy C. Turner

Centre for Ecological and Evolutionary Synthesis (CEES), Department of Biosciences, University of Oslo, P.O. Box 1066, Blindern, 0361 Oslo, Norway

ARTICLE INFO

Article history:
Received 1 April 2014
Received in revised form 13 June 2014
Accepted 16 June 2014
Available online 21 June 2014

Keywords:
Camera traps
Data acquisition and management software
Free and open source software
Wildlife monitoring

ABSTRACT

The use of camera traps is now widespread and their importance in wildlife studies is well understood. Camera trap studies can produce millions of photographs and there is a need for a software to help manage photographs efficiently. In this paper, we describe a software system that was built to successfully manage a large behavioral camera trap study that produced more than a million photographs. We describe the software architecture and the design decisions that shaped the evolution of the program over the study's three year period. The software system has the ability to automatically extract metadata from images, and add customized metadata to the images in a standardized format. The software system can be installed as a standalone application on popular operating systems. It is minimalistic, scalable and extendable so that it can be used by small teams or individual researchers for a broad variety of camera trap studies.

1. Introduction

The use of camera traps for wildlife surveys, population estimations and monitoring is now a common scientific method with many established mathematical models, frameworks and protocols (e.g., O'Connell et al., 2011). Camera traps are also beginning to break new ground in animal behavioral studies (VerCauteren et al., 2007; Jennelle et al., 2009; Prasad et al., 2010). Existing camera trap data management solutions and protocols are primarily targeted at surveys and monitoring (e.g., Fegraus et al., 2011), and sometimes involve potentially error prone steps in management of data (e.g., Harris et al., 2010), while more generic solutions (e.g., Sundaresan et al., 2011) are difficult to extend, customize, or scale up to handle large or unconventional projects. Some solutions have limitations by their choice of underlying software systems. For example, Camera Base (Tobler, 2007) is limited to handle tens of thousands of files because it uses Microsoft Access as its database management system. With camera trap technology being affordable and accessible, projects in the future may be capable of producing millions of photographs in a short period of time and there is a need to help manage the impending camera trap data deluge.

We introduce a minimalistic, scalable and extendable camera trap data management software program that can deal with an arbitrarily large number of photographs, limited only by physical limits of commodity computer hardware. While the basic frameworks and protocols exist to manage data for large camera trap studies, they still exist as disparate tools and methods that have not been combined specifically for use in ecological camera trap studies.

The Aardwolf software program combines these tools and methods to provide an intuitive interface to deal with camera trap studies that may contain millions of photographs. The program is intended for use on personal workstations, and can be installed on several popular operating systems. It is a free and open source software, and uses various open standards and only depends on other free and open source software libraries. Aardwolf is built to manage a camera trap study's workflow from downloading of photographs to exporting of study-specific data for statistical analysis.

2. Methods

2.1. Protocols and tools

Camera trap data management workflow can be seen as having three distinct stages: file management, data annotation and data extraction. Analysis that follows data extraction is specific to a study and thus beyond the reach of generalization.

Camera trap studies often have dozens of students, assistants, volunteers and other collaborators helping researchers with management of data, or subsets of it. This may result in data inconsistencies and irrecoverable data losses if strict protocols for data collection, annotation and sharing are not enforced. One method for enforcing such protocols is to use a single software program to manage the entire data workflow from file management to data extraction. Using a single software program may also help reduce data losses.

Organizing an arbitrarily large number of related digital files, like photographs in a camera trap study, requires a system to have physical independence from its logical structure. Files may need to be spread over several physical storage devices that are either connected locally or via computer network protocols. Regardless of storage mediums

* Corresponding author.
E-mail address: yathin@gmail.com (Y.S. Krishnappa).

http://dx.doi.org/10.1016/j.ecoinf.2014.06.004
1574-9541/

and network protocols, today's computer users are already well-equipped to deal with a large number of files, and are familiar with organization of files in folder hierarchies. Individual researchers can readily use their personal machines with an array of commodity storage devices to organize millions of camera trap photographs.

Data annotation for camera trap studies largely means identification of objects and features in individual photographs or groups of photographs. While automatic image classifiers for ecological subjects may help automate this task (e.g., Lu and Weng, 2007; Bolger et al., 2012), ecological camera trap studies may still require human experts to identify and quantify objects and actions within each photograph. There are two ways to store annotation information: within the images, or in an external database. Storing annotation information in an external database is efficient when fast retrieval of data is required, and this has been the preferred way for software developed to deal specifically with camera trap data (e.g., Fegraus et al., 2011). However, this method can potentially suffer from data inconsistency and verifiability issues that arise from related data being stored in two (or more) locations. One way to minimize data errors is to store all related data together, which means storing file-specific annotation information within individual files.

Digital photographs can already be annotated with metadata like time, date and geographical coordinates. Popular storage formats for digital images, like the Joint Photographic Expert Group format (JPEG), also support storage of custom metadata through open standards like Exchangeable Image File format (EXIF) and Extensible Metadata Platform (XMP). This means that all kinds of non-variable annotations (e.g., species in a photograph) can be stored within the image file itself. However, retrieval of annotation information will require reading and parsing all the files in a collection, and this is generally many times slower than data retrieval from an external database.

2.2. Data management

Generally, a software program dealing with camera trap data needs to manage two kinds of data: it needs to organize files and it needs to manage metadata associated with individual files. Both kinds of data can be stored in an external database. Files, for example, are generally stored as binary fields in most popular relational database systems. However, a more intuitive approach is to store the files in an operating system's filesystem, and only store file metadata in an external database. Metadata stored in an external database can further be thought of as a cache of the metadata in individual files, and this ensures that files are the single source of truth.

A minimalistic way of organizing files in a camera trap study is to store them in a three-level directory hierarchy. The top level directory is the name of the project, with each project directory containing a number of camera directories, and each camera directory containing a number of directories (folders) corresponding to individual download operations. The low level directory names can contain a timestamp in their names to allow for uniqueness in name as well as intuitive management. Since we can only guarantee filename uniqueness within an individual directory in most operating systems, this physical structure also ensures copying of directories from a camera trap storage card which can be done as-is.

Camera manufacturers may add their own custom metadata (e.g., make and model of camera) in addition to standard EXIF and XMP tags (e.g., photograph creation time) that make up a photograph's metadata. Identification of features and objects in photographs can be seen as additional metadata, and these data can be stored either in an external database or as a custom data structure within the file. The advantage of storing in an external database is that the database serves as an index and does fast search and retrieval of data; the disadvantage is that data are not co-located with other metadata. We propose a hybrid approach, where such metadata are stored in an external database as well as in individual files. Keeping annotation metadata within the files in an open and standardized format means that it can be extended as well as be usable by a wide range of third-party software programs. The high-level relationship between entities in such a minimalist design can be represented by an Entity-Relationship (ER) model (Fig. 1), and this model can be directly mapped on to a relational database implementation. Annotation information stored as in-file metadata using XMP can be stored as a simple one-dimensional collection of name-value pairs.

3. Results

We developed Aardwolf because there was no other software available that could be customized to efficiently manage and annotate our camera trap data that contained over a hundred thousand photographs within the first few months of the study's commencement. Eventually, the study contained nearly 1.2 million photographs generated from 26 cameras over a three year period (Turner, unpublished data). During the progression of the study we realized that more of the workflow had to be automated to reduce cumbersome human-caused data errors, such as mislabeling of folders or copying to wrong destinations. We also wanted to create a flexible and extendable system because of evolving ways of looking at, and annotating, data over the course of a multiyear study. This could otherwise demand massive changes in the software. In the end, we had a refined system that is simple in design, yet powerful.

The software program is called Aardwolf after *Proteles cristata*, a mesocarnivore that feeds primarily on termites (Skinner and Chimimba 2005). We think of our program as analogous to an aardwolf, swiftly devouring termites (i.e. photographs) from termite mounds (i.e. camera traps).

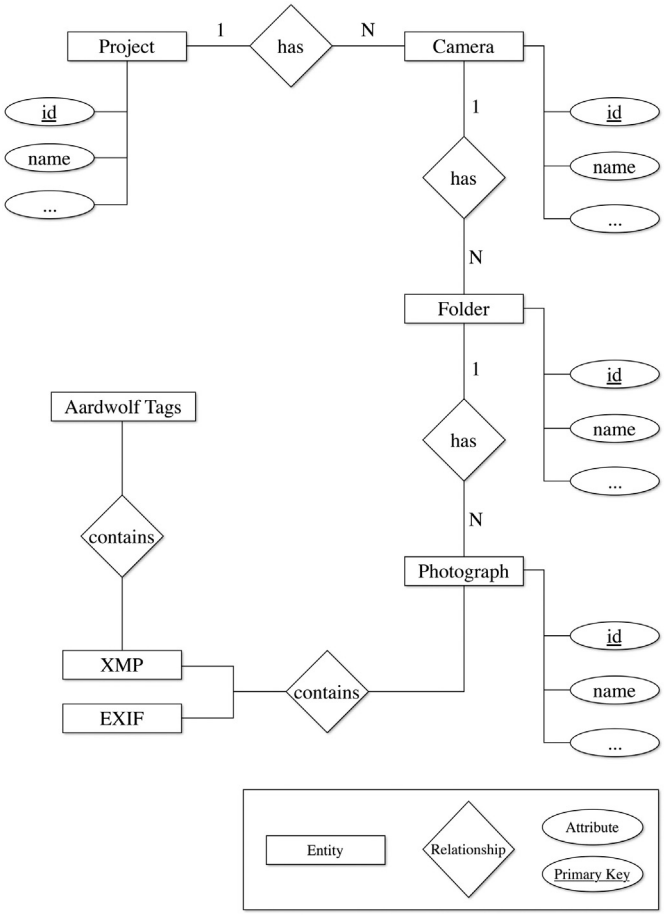

Fig. 1. Simplified data model showing relationships between entities.

3.1. Data losses

During the three-year period of our study, we had 90 incidents which interfered with collection of photographic data. These incidents, called data outages, were caused by detectable physical camera failure (e.g., rain damage, dead batteries, card failures, destruction by wildlife) and by human errors (e.g., failure to copy data, copying to unknown location). The first two years of the study had 74 outages out of which 14 were human errors (15.5%). After management of data downloads was added to the program, there was only one human error out of the 16 outages that followed (6.2%).

3.2. User interface

Aardwolf's user interface follows the well-understood file browser interface, both in visual appearance and response to input devices, that is popular in commonly used operating systems, namely Microsoft Windows, Mac OS and Linux. The program works on all of these operating systems and an Aardwolf project can be shared seamlessly between them. Further, project files can reside on a central storage device, like a shared hard disk or a network filesystem, and multiple users on any of these operating systems can work on it simultaneously. After an Aardwolf project is created, a user is required to create one or more cameras before any files can be downloaded. The project and camera names are not editable after creation, and can only be composed of alphanumeric characters and hyphens. One or more tag groups must then be defined for annotations, without which Aardwolf is essentially an image browser for files downloaded into the program. The general data flow in Aardwolf is represented in Fig. 2. Data from cameras are downloaded, and can then be annotated by users or other specialized computer programs, and annotated data can be extracted for further analysis using the Aardwolf user interface. The Aardwolf datastore is accessed by an operating system's filesystem interface. Modern filesystems allow for remote sharing of files. This means multiple users can simultaneously use different computers to read or modify an Aardwolf datastore. A screen shot during the annotation stage is shown in Fig. 3, and Fig. 4 is a screen shot of Aardwolf annotated data viewed within a third-party software program.

3.3. File management

Dedicated distributed filesystems may require complex or specialized hardware and software configurations. A minimalistic approach to distributing a large collection of files is to partition them over multiple storage devices and rely on an operating system's filesystem to manipulate them.

Aardwolf supports logical partitioning of data, with an option for a partition to be physically located anywhere in an operating system's filesystem. Relying on an operating system's filesystem interface gives Aardwolf the flexibility to separate logical organization of files from its physical location, and gives it a reliable framework that is well understood and documented. This means partitions can be stored on additional hard drives or networked filesystems (e.g., Fig. 5), thereby providing the ability to work with an arbitrarily large number of files.

Each filesystem location or device must be registered as a partition in Aardwolf's partition manager. Photographs are copied without modification from a camera, or memory cards, and organized into folders under camera directories of an Aardwolf project. In its default operation, a user has to choose a camera from a list and a download folder will be created automatically under the camera directory in the program's primary partition. However, a user can choose a secondary partition if more control over distribution of folders is required (e.g., distributing over multiple storage devices). Aardwolf hides these physical locations and presents the file data only in its logical form, as a three-level structure: project, camera and download folder. Although this organization of data offers no computational advantage, it is nonetheless intuitive for users to visualize file data. In our study (Turner, unpublished data), addition of file management to Aardwolf reduced file collection and download errors.

3.4. Data annotation

Every image in an Aardwolf database is annotated with user-defined camera information (e.g., location) and project information (e.g., aim of the study and experimental design). Each image can then be annotated with user-defined metadata that is identified in an image by a human expert. These metadata can currently be in the form of free text (e.g., notes or quantitative observations) or checkboxes (e.g., species presence/absence or demography), and are collectively called “tags” in Aardwolf. Tags are grouped into “tag groups”, with the option of viewing only a subset of such groups during the process of annotation. The freeform nature of tags and tag groups gives users flexibility for designing annotations in a variety of camera trap studies.

If the option to add metadata to the image files is selected during project creation, all metadata that Aardwolf adds to image files will also be stored in a XMP namespace called ‘actns’, which stands for Aardwolf Camera Trap NameSpace. Aardwolf will only modify image metadata in this namespace and will only read operations on other metadata in the image files. This means that metadata in an image that existed prior to annotation by Aardwolf remain unchanged by the program.

Image annotation is often a time-consuming process because of the lack of good automatic image classifiers for ecological subjects. Aardwolf helps speed up this time-consuming process by providing an option to add helper modules called “groupers”. Groupers cluster similar images together so that images in a group can all be selected by a single click and then annotated with the same information, if applicable. Groupers can be based purely on statistical heuristics involving in-camera metadata (e.g., timestamp of the photograph), or they can read the image files and apply image classification techniques to group similar images. Aardwolf's default grouper uses a simple statistical heuristic to group together photographs that were taken within a user-defined time value of its chronologically preceding and succeeding

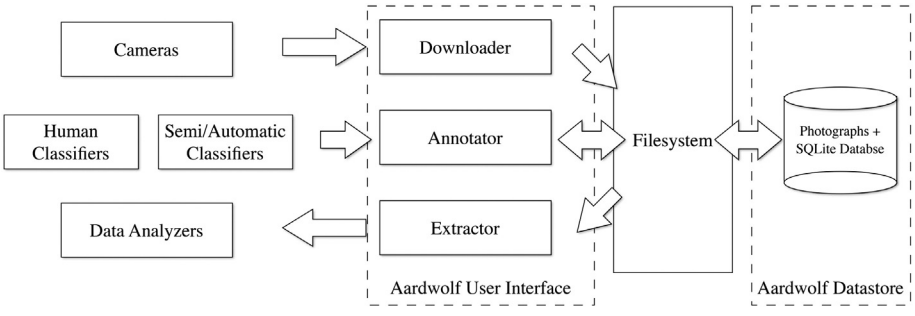

Fig. 2. Data flow in Aardwolf program.

Fig. 3. Screenshot of the Aardwolf program running on Mac OS.

neighbors. Aardwolf provides an extendable interface that can be used to add other types of groupers to the program.

Filters give a user the option of viewing only files that match a condition (e.g. viewing files that are tagged to contain a particular animal species). This helper utility can facilitate multi-pass annotations. For example, identifying a springbok (*Antidorcas marsupialis*) can be an easier task than identifying what it is doing in a frame (e.g., attempting to browse or graze), and this means that the first iteration (identification of species) can be done quickly or by a less skilled person than the second iteration which may require more time, expertise and knowledge.

The search option gives a user the option to see images that match a search term within a folder, a camera or the entire project. This can further help in maintaining consistency in annotation tasks, for example when trying to consistently identify behavioral aspects in a photograph.

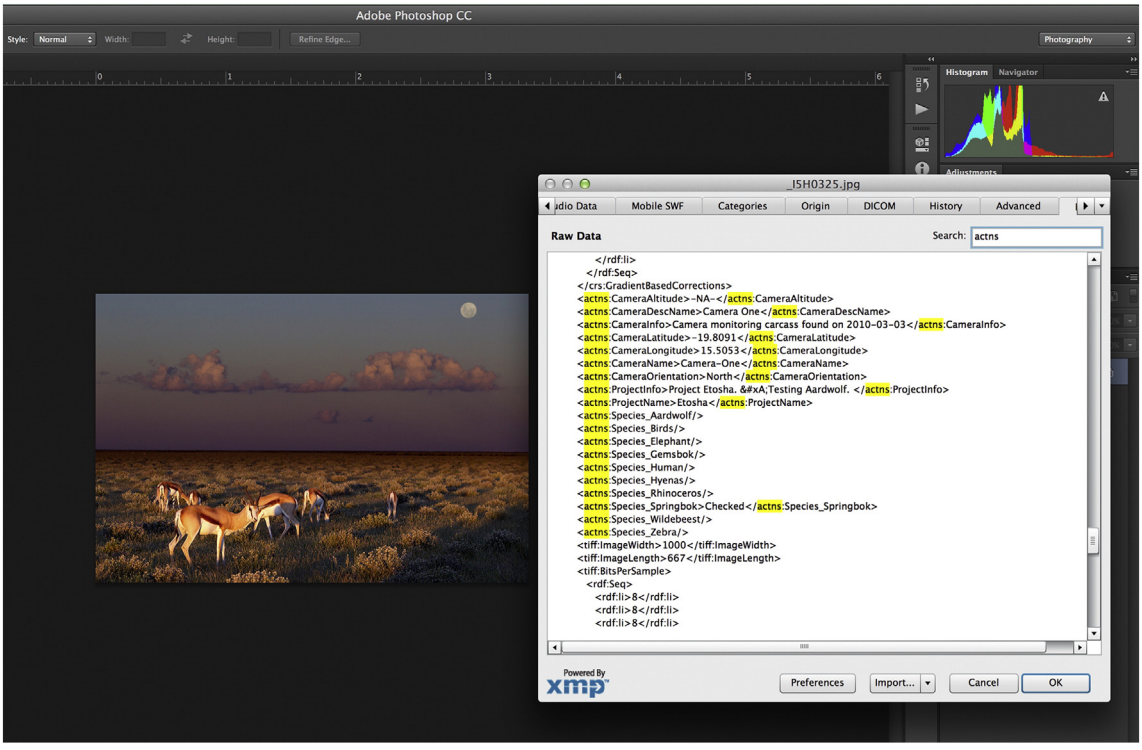

Fig. 4. Data annotated with Aardwolf can be read by any other image editor that understands XMP (e.g. Adobe Photoshop CC).

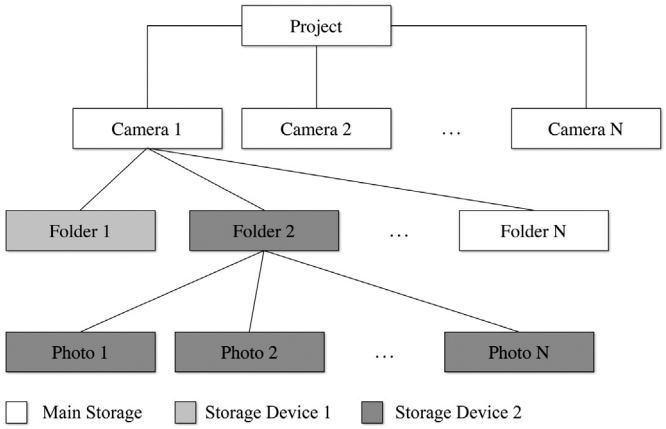

Fig. 5. Hierarchical structure for physical location of photographs, with an example of a project using multiple storage devices.

3.5. Data extraction

Aardwolf is built to use relational databases that provide a Structured Query Language (SQL) interface, and it uses a minimalistic, popular and lightweight relational database management system, SQLite, to store all of its operational data and photo metadata. SQLite uses a single file to store all of its data; this means that all project data can be kept together in a single location and there is low setup and maintenance overhead (e.g., when a project needs to be moved to a different storage location or a project resides on a shared external disk drive). Aardwolf can be configured to use more powerful relational database systems like MySQL and PostgreSQL, if necessary.

Using a relational database for metadata storage means that the Aardwolf database can be queried using the full power of SQL. Although harnessing the full power of SQL requires some programming knowledge, Aardwolf provides an easy interface to create reports and extract data. Reports give a user insight into the state of the project, which can be used to make time estimates or other changes in a project.

Data can be exported into a Comma-Separated Value (CSV) file, which can then be imported into any statistical software package. The columns in Aardwolf's CSV output can be classified into three groups:

Identification group, which contains a unique row number (photo_id) in the SQLite database and the names of the filesystem location (fs_location), camera (camera_name), folder (folder_name) and photograph (photo_name) which can be used to construct a file path for the photograph.

Photograph metadata group, which contains unmodified metadata from a photograph's EXIF data (e.g., photo_time). Currently, Aardwolf only extracts and stores the creation date and time because the full list of EXIF tags is exhaustive and camera makers may add their own custom EXIF tags which are not standardized. Users who require additional EXIF tags must modify Aardwolf's source code or use an intermediate program to annotate Aardwolf's CSV output.

Aardwolf metadata group, which contains one column for each tag group and tag combination. The column name uses an underscore to separate tag group name from tag name. For example, a Behavior_Browsing column contains values for tag group 'Behavior' and tag 'Browsing'.

Aardwolf data can also be extracted directly from the SQLite database using the SQLite interface or by using a graphical query builder (e.g., sqliteman) if more control over the output format is needed.

3.6. Limitations

Aardwolf was built to handle large camera studies and was successfully used for a study with over a million photographs. Table 1 shows sample query running times for commonly used Aardwolf operations for various database sizes. SQLite supports billions of rows (Owens and Allen, 2006) and database sizes that can be several terabytes. However, the current usability of Aardwolf is limited by the main memory size of a user's computer, which for a computer with 4 gigabytes of main memory is around 10 million photographs. Filtering, search and data extraction are the three most time-consuming queries because they need to go through every single row on the SQLite tables they operate on. The performance of filtering and search can be improved significantly by using SQLite extensions such as virtual tables, full-text search capability and secondary indexes that can be utilized to lower query running times for search and filter queries. The implementation of data extraction can be improved to not require the entire database to be in memory. These improvements in the future may help Aardwolf support hundreds of millions of photographs. Additionally, Aardwolf can be reconfigured to use more powerful database management systems like MySQL or PostgreSQL to support an even greater number of photographs.

3.7. Installation

The latest builds of Aardwolf can be downloaded as installable packages for Microsoft Windows and Mac OS operating systems (http://sourceforge.net/p/aardwolf/). The complete source code for the project can be downloaded from http://github.com/yathin/aardwolf. All enhancements and bug fixes are merged into the online source code along with each binary version release. All required dependencies are packaged within the binary packages, which means users can just install the Aardwolf software as any other package and not worry about missing dependencies.

4. Summary

We believe that Aardwolf will help in minimalistic management of large amounts of camera trap data, especially in projects that have a small number of personnel. As an open source and multi-platform solution that is fast, flexible and extendable, future development may help in even faster and easier management and annotation of data.

The Aardwolf program can further be ported to have a web interface which could help it achieve even greater scale and annotation efficiency. For example, a web interface could facilitate crowd sourcing the annotation task and support remote upload and download of photographs and data.

Acknowledgments

We would like to thank Salena Thong, Katie Dean and Ronald Ho who put in many hours annotating 1.2 million photographs, and whose feedback helped refine the Aardwolf program in the process. We would also like to thank Craig Tambling, Colin Tucker and Zoe Barandongo for excellent constructive feedback on the Aardwolf program when we were preparing it for publication. We also thank the staff at the Etosha Ecological Institute in Etosha National Park, Namibia, for permission to conduct the research that lead to the Aardwolf program and for logistical support. We thank Nils Chr. Stenseth and the Centre for Ecological and Evolutionary Synthesis (CEES), Department of Biosciences, University of Oslo for hosting us during the development of this software. Funding was provided by the CEES, NSFOISE-1103054 (to WCT) and NIHGM083863 (to Wayne Getz).

Table 1
Sample running times for Aardwolf queries. Machine specifications: 2.6 GHz Intel dual core i5, 8 GB 1600 MHz DDR RAM, 500 GB flash drive. Query running times are given in milliseconds (ms), seconds (s) and minutes (m). The database sizes for N photographs were N = 1000: 1 MB, N = 100,000: 30 MB, N = 1,000,000: 300 MB, N = 10,000,000: 3 GB.

Query	Definition of N	Query running time per N				Notes
		N = 1000	N = 100,000	N = 1 M	N = 10 M	
List cameras	Number of cameras	1 ms	30 ms	350 ms	3 s, 500 ms	In practice number of cameras <100
List folders	Number of folders	1 ms	30 ms	250 ms	2 s, 500 ms	In practice, number of folders <1000
List folders after applying filters	Number of photographs	1 ms	20 ms	3 s, 300 ms	8 s, 500 ms	Base case of listing 1 folder with N photographs
List folders after applying filters	Number of folders and number of photographs	1 ms	200 ms	8 s	1 m, 10 s	Query times increase with number of folders and number of photographs
List photographs in a folder	Number of photographs in a single folder	1 ms	30 ms	250 ms	3 s, 200 ms	In practice, folders contain <10,000 photographs in Aardwolf because each folder corresponds to a single download event from a camera storage card
List photographs in a folder after applying filters	Number of photographs in a single folder	1 ms	150 ms	1 s, 450 ms	19 s, 500 ms	
Search tags in photographs	Number of photographs in project	1 ms	250 ms	2 s, 300 ms	32 s, 500 ms	
View data	Number of photographs in project	15 ms	1 s, 400 ms	15 s, 500 ms	3 m, 20 s	
Export data to a CSV file	Number of photographs in project	1 s	3 s, 500 ms	30 s, 500 ms	5 m, 35 s	This operation depends primarily on write speed for target device of the CSV file. These numbers are from a machine with a flash drive, which is typically several times faster than a traditional hard drive.

References

Bolger, D.T., Morrison, T.A., Vance, B., Lee, D., Farid, H., 2012. A computer-assisted system for photographic mark-recapture analysis. Methods Ecol. Evol. 3, 813–822.

Fegraus, E.H., Lin, K., Ahumada, J.A., Baru, C., Chandra, S., Youn, C., 2011. Data acquisition and management software for camera trap data: a case study from the TEAM Network. Ecol. Inform. 6, 345–353.

Harris, G., Thompson, R., Childs, J.L., Sanderson, J.G., 2010. Automatic storage and analysis of camera trap data. Bull. Ecol. Soc. Am. 91, 352–360.

Jennelle, C.S., Samuel, M.D., Nolden, C.A., Berkley, E.A., 2009. Deer carcass decomposition and potential scavenger exposure to chronic wasting disease. J. Wildl. Manag. 73, 655–662.

Lu, D., Weng, Q., 2007. A survey of image classification methods and techniques for improving classification performance. Int. J. Remote Sens. 28, 823–870.

O'Connell, A.F., Nichols, J.D., Karanth, K.U., 2011. Camera Traps in Animal Ecology: Methods and Analyses. Springer, Tokyo.

Owens, M., Allen, G., 2006. The Definitive Guide to SQLite, vol. 1. Apress, Berkeley.

Prasad, S., Pittet, A., Sukumar, R., 2010. Who really ate the fruit? A novel approach to camera trapping for quantifying frugivory by ruminants. Ecol. Res. 25, 225–231.

Skinner, John D., Chimimba, Christian T. (Eds.), 2005. The mammals of the southern African sub-region. Cambridge University Press.

Sundaresan, S.R., Riginos, C., Abelson, E.S., 2011. Management and analysis of camera trap data: alternative approaches (response to Harris et al., 2010). Bull. Ecol. Soc. Am. 92, 188–195.

Tobler, M., 2007. Camera Base Version 1.3. Botanical Research Institute of Texas http://www.atrium-biodiversity.org/tools/camerabase.

VerCauteren, K.C., Burke, P.W., Phillips, G.E., Fischer, J.W., Seward, N.W., Wunder, B.A., Lavelle, M.J., 2007. Elk use of wallows and potential chronic wasting disease transmission. J. Wildl. Dis. 43, 784–788.