Environ Monit Assess (2017) 189: 527
https://doi.org/10.1007/s10661-017-6206-x

Semi-automated camera trap image processing for the detection of ungulate fence crossing events

TLDR:
Extracted animal from background by:
- Downsampling: averaged 16 pixels into 1 and applied a 3x3 Gaussian with gamma adjustment
- Averaging pixels in nearby frames
- Thresholding: compare input with absolute difference, using combined RGB sum of 60 to create binary image, retaining animal pixels.
- Histogram rules: brown = animal

Michael Janzen · Kaitlyn Visser · Darcy Visscher · Ian MacLeod · Dragomir Vujnovic · Ksenija Vujnovic

Received: 11 March 2017 / Accepted: 27 August 2017 / Published online: 27 September 2017

Abstract Remote cameras are an increasingly important tool for ecological research. While remote camera traps collect field data with minimal human attention, the images they collect require post-processing and characterization before it can be ecologically and statistically analyzed, requiring the input of substantial time and money from researchers. The need for post-processing is due, in part, to a high incidence of non-target images. We developed a stand-alone semi-automated computer program to aid in image processing, categorization, and data reduction by employing background subtraction and histogram rules. Unlike previous work that uses video as input, our program uses still camera trap images. The program was developed for an ungulate fence crossing project and tested against an image dataset which had been previously processed by a human operator. Our program placed images into categories representing the confidence of a particular sequence of images containing a fence crossing event. This resulted in a reduction of 54.8% of images that required further human operator characterization while retaining 72.6% of the known fence crossing events. This program can provide researchers using remote camera data the ability to reduce the time and cost required for image post-processing and characterization. Further, we discuss how this procedure might be generalized to situations not specifically related to animal use of linear features.

Keywords Camera trap · Computer vision · Monitoring · Remote camera · Image processing

M. Janzen (✉) · K. Visser · D. Visscher · I. MacLeod
The King's University, Edmonton, Alberta, Canada
e-mail: Michael.Janzen@kingsu.ca

D. Visscher
e-mail: Darcy.Visscher@kingsu.ca

D. Vujnovic · K. Vujnovic
Alberta Parks, Edmonton, Alberta, Canada

Introduction

Monitoring wild animal populations may involve visual monitoring of animals or live trapping, which are labor intensive, expensive, impractical, and invasive (Sirén et al. 2016; Efford 2004; Jhala et al. 2009). An alternative to direct human observation is a camera trap use that captures a series of images in response to movement in the camera's field of view. While this approach removes the need for a human observer in the field, it results in a number of digital images that require post-processing by human operators for the identification and characterization of events of interest. A number of applications have been built to manage and characterize the large numbers of images within databases, often in project specific forms (e.g.,

Bubnicki et al. 2016; Hines et al. 2015; Krishnappa and Turner 2014; Tobler et al. 2015; Forrester et al. 2013); however, there are fewer options for automating the removal of non-target images. As a result, researchers must deal with an "analytical bottleneck" during post-processing with human operators (Spampinato et al. 2015). Despite this potential bottleneck in terms of post-capture image analysis, remote cameras have become ubiquitous for ecological research (Burton et al. 2015; Meek 2014).

Developing a semi-automated processing of obtained images has the potential to significantly reduce "analytical bottleneck." In particular, the ability to discriminate and remove images containing "noise" or images triggered due to movement attributed to the background (e.g., vegetation, precipitation, wind, or clouds) and those containing animals will result in a large reduction in processing and characterization time by human operators. Secondly, there is a need to further reduce the number of images being processed by discriminating between events of interest and non-target events. Crucial to both these objectives is the ability to remove images consisting of only background noise and identify images containing animals and events of interest. The primary technique to do this is background subtraction (Piccardi 2004).

Background subtraction techniques have been successful in video imaging studies used to monitor hummingbirds, beavers, and fish (Weinstein 2015; Swinnen et al. 2014; Ardekani et al. 2013; Goehner et al. 2015). In a camera trap setup, the images are necessarily spaced further apart temporally, making the background subtraction process more challenging. Other semi-automated approaches identify an individual or species, but may require human interaction to appropriately segment the image (Yu et al. 2013; Anderson et al. 2010; Osterrieder et al. 2015). Camera traps, however, have advantages over video recording making it suitable to solving different problems. In addition to being potentially less expensive and producing a higher resolution image than a video camera, a camera trap can also require less human monitoring. For example, since the camera trap activates in response to movement, the camera can be left for weeks before requiring a new memory card or a battery recharge. This ease of use and reduced field investment have made remote camera traps an important ecological tool in spite of the difficulties posed by automated image processing (Bubnicki et al. 2016).

Here, we describe a semi-automated computer program to pre-process remotely captured images into relevant and irrelevant sets for future human consideration. We developed our project in response to a specific problem instance, ungulate fence crossing events (Visscher et al. 2017), where a large number of the images contained approaching animals but no fence crossing event (in particular domestic bovids), non-target species, additional noise in the form of wind, and illumination changes resulting in image sequences that are not of interest. Our objectives were twofold: (1) to reduce the number of image sequences that required direct input from human operators in terms of characterization, and (2) the identification of crossing events. Further, we provide an assessment of achieving these goals by the program based on a known dataset characterized completely by a human operator.

Methods

Data collection

Twelve Reconyx motion and thermal-activated remote cameras were installed at the Wainwright Dunes Ecological Reserve, located approximately 250 km southeast of Edmonton in the Parkland Natural Region of Alberta. The 2821 ha reserve is located within an agricultural matrix and has an ongoing history of being used seasonally (June to October) as rangeland for free roaming cattle by a local grazing association. Sites were selected based on evidence that wildlife used these locations for moving into and out of the reserve. Cameras were located on perimeter fences to best capture animal crossings. Biologists from Alberta Parks maintained the cameras and downloaded images on a monthly basis from September 2011 until November 2014.

Once the data had been downloaded, individual pictures were viewed and manually sorted, classified, and summarized using the Reconyx MapView Professional program by tagging the image with appropri-

ate metadata such as species, demographics, crossing events, methods of crossing, behaviors, and other noteworthy data. Some images containing domesticated animals, humans, or species of non-interest or no animals (i.e., background "noise") were removed. The remaining images were provided to us as a testing dataset to assess the efficacy of the semi-automated program.

Program description

As an overview, our program functioned in the following fashion, described below in more detail. Images were copied from a camera trap memory card and, for each site, images were manually marked to identify the fence region. Based on time between images (we used a default of 30 s), our program separated these input images into image sets for processing, each representing one event. For each image set, our program combined images into a background image which ideally contained no animal of interest. The background image was compared to each input image to compute a set of difference images, highlighting animal movement. Within these image sets, groups of connected pixels were combined into fragments. Fragments clearly not corresponding to animals were removed using color histogram based rules. Finally, the set of images was assigned an animal crossing confidence.

The background image was made using a series of image processing steps, which were customizable depending on instructions specified in a file (see Gonzalez and Woods 2007 for information on basic image processing techniques).

- **Fence marking** To determine if a crossing event occurred, we marked the fence location using two lines bordering the fence. The marking was applied to an average image, and used for all images taken from the same site since the fence is relatively stationary for all images from the camera trap. We used an implicit equations of lines to avoid division by zero of horizontal lines and since this approach easily compared image pixels to one of three regions defined by the two lines (see Fig. 1f). In cases where there was fence movement due to wind, the fence in the average image would appear fuzzy, but the lines were placed to contain the maximum extent of the fence.
- **Image set segmentation** The images contained many crossing and non-crossing events. We used time between images to segment these events, the default value being 30 s. This resulted in image sets between 3 and approximately 200 images in size, as the camera was set to record a set of three images every time it was activated.
- **Downsampling** The input images were down sampled by averaging 16 pixels into one pixel and a 3×3 Gaussian filter further smoothed each image. We applied a gamma adjustment to each pixel to equalize brightness levels between images, so all images had the same average intensity.
- **Temporal pixel averaging** Each pixel location across the images was averaged into a background image; however, this produced ghosting, where faint images of the animal were present in the background image as seen in Fig. 1d. To reduce problems with ghosting, our program computed the standard deviation of pixels in red, green, and blue channels. Only pixels within one standard deviation of the average color were included in the background image as shown in Fig. 1e.
- **Thresholding** We constructed our difference image set by comparing each input image with the background and kept the absolute difference value, thresholded using a default value of combined red, green, and blue sum of 60 to create a binary image for consideration. A value of 60 was selected as it retained pixels from animals in the output image, while it removed pixels resulting from noise such as slight differences in lighting between images. Median filtering the image removed some salt-and-pepper noise. The bottom row in Fig. 1 shows thresholded images after the next two steps are applied.
- **Dilation and erosion** Erosion and diffusion operations connected close fragments of pixels together while removing small fragments. A series of small nearby fragments may better be considered one fragment of the same animal. To

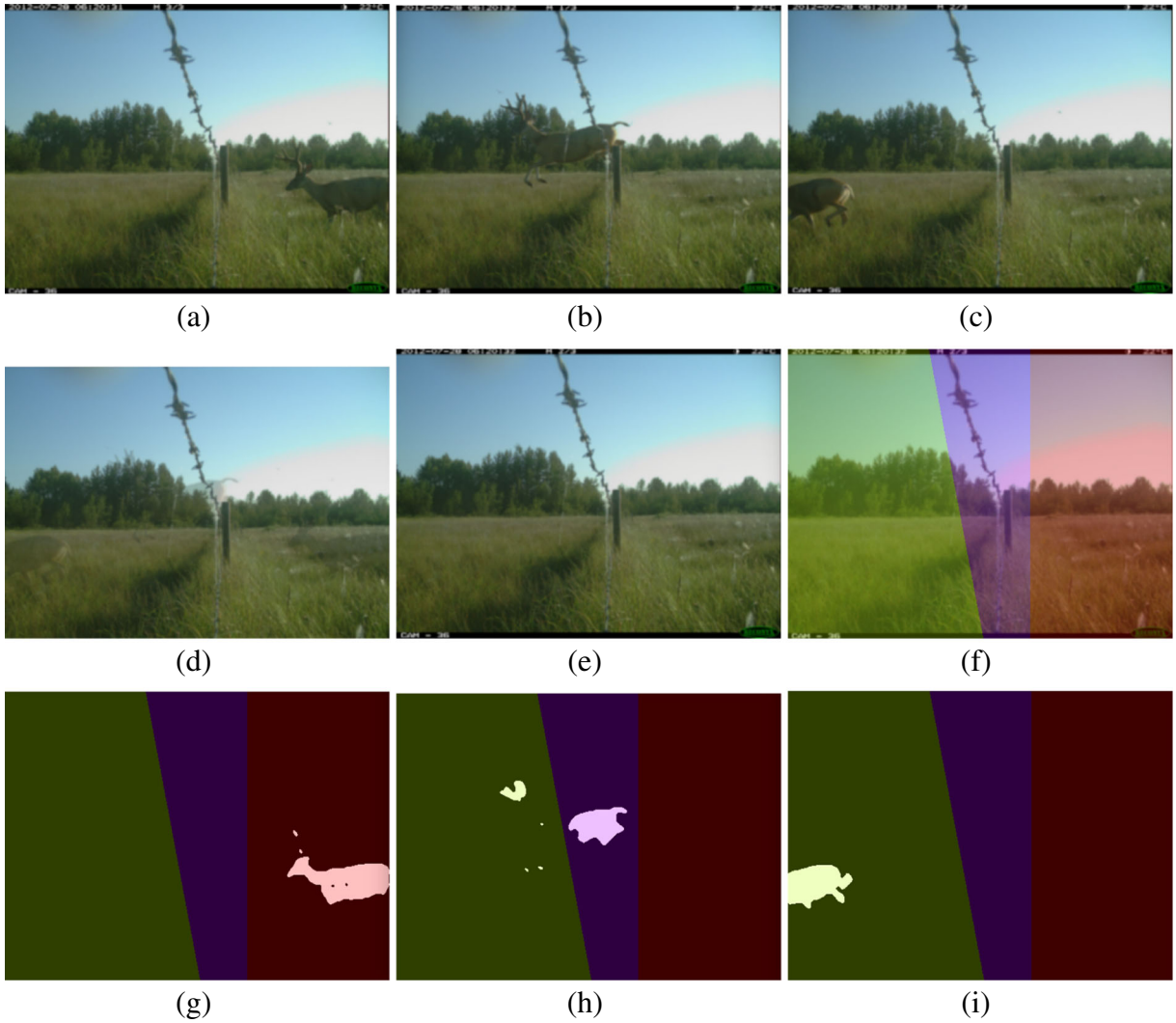

Fig. 1 Typical steps to identify animal using background subtraction. Top row shows input images. Middle row shows backgrounds with ghosting (**d**), without ghosting (**e**) and with user identified fence region (**f**). Bottom row shows difference images where each image corresponds to an image in the top row

handle this case, we dilated and eroded the resulting binary difference image, with a default value of three. While adjustable, this setting made any background pixels with three neighbors a foreground pixel, filling in gaps between foreground pixels. Thus, small nearby regions merged into one another during the dilation step, which an equivalent number of erosion steps returned the resulting fragment to approximately the original size, corresponding to the detected animal.

– **Histogram rules** For each remaining fragment, we applied rules to its color histogram from the input image to determine if the fragment corresponded to an animal, background, or was undetermined. We manually constructed our rules based on the ungulates of interest. For example, a fragment consisting primarily of white or green is likely cloud or grass respectively, while a brown fragment was typically retained, as it may correspond to an ungulate of interest. Animal

and undetermined fragments were retained for further processing, while background fragments were removed from consideration.

Each histogram rule was written as a Java class inheriting from a more general class to easily enable adding or removing histogram rules from our program. Thus each histogram rule received as input the distribution of intensity discritized into 16 bins. The rule would return two binary values indicating the fragment corresponded to an animal, background, or neither. This approach allowed us to program combinations of colors based on the percentages in each histogram bins. For example, a histogram rule could require that sum of the total amount of color be contained in some bins while other bins could contain no more than a certain amount of color. An example rule is described in more detail in the section below on program testing.

- **Animal determination** Finally, our program calculated how many pixels were located in each of the three regions surrounding the fence, based on the number of pixels in each difference image. Images were ignored where the pixels identified as difference pixels exceeded 65% of the region as this tended to correspond to an animal approaching the camera, which occluded the fence. A region was considered to contain the image of an animal if the difference pixels remaining after processing exceeded 0.5% of the image. The binary determination of an animal in each region determined which confidence category into which the image set was binned, described as follows:

1. **High crossing confidence**. An animal detected on one side of the fence was detected in the next one or two images on the other side of the fence.
2. **Likely crossing detected**. An image set contained only three images, one or two which contained an animal and one or two which did not. This likely corresponded to a fast moving animal crossing the fence at high speed. The animal may be anywhere in the image including the fence region.
3. **Unknown**. The image set was classified as unknown if it did not fall into one of the other four categories.
4. **Noise**. All images detected animals on both sides of the fence more likely corresponding to wind, clouds or snow than an animal crossing.
5. **Non-crossing event**. One or more animals were detected approaching, but not crossing, the fence. This can occur with ungulates but frequently occurred with images involving cattle contained within the fence.

- **File moving** After assigning a confidence to each image set, we assigned directories for each category, and the corresponding images were moved for manual processing. Consequently, likely crossings would be investigated (categories 1, 2, and 3) while likely non-crossings would be ignored (categories 4 and 5).

Our program was written using the Java programming language without using libraries external to the Java Development Kit since we wrote our own functions for image processing. The program executes on multiple architectures, but was designed to be automated on a Windows platform due to the use of batch files between executable components.

Program testing

The sequence of image processing operations used by our program is customizable, so we devised a typical sequence of instructions to automatically detect the presence of a crossing animal, and encoded the sequence into an xml instruction file. We used the same instruction file for all images, except we adapted the x and y coordinates for each site location to correspond to the fence location. The xml instructions corresponded to actions the user can manually test before applying the instructions to a set of images, but our program proceeded without user intervention.

Then, 480,542 input images were distributed to approximately 10 computers divided primarily by site location and secondarily by date. The computers processed the images for approximately 2 weeks, although some of this time was idle between assigned tasks. The length required to process an image varies depending on a the instructions applied, hardware configuration, and the image content processed. Using a Intel i7-4790 processor at 3.6 GHz and a SSD

storage device each image required approximately 2.6 s to process. Manual processing of image sets already completed by the computer occurred while the computer continued to process additional image sets. In total, the manual processing required approximately three months.

The instruction to delete background fragments applied a series of histogram rules to each fragment it finds. The rules were based on our intuition, and tested against examples from our data set. As an example, in day time, images of ungulates tend to be within a particular shade of brown, and we considered colors differing enough from a typical ungulate to be background. When we encoded part of this into the program, we considered the peaks of the color of an animal in the red, green, and blue histograms. Each color was binned into 16 evenly sized color ranges labeled from 0 to 15. When implementing our rule, to determine if a fragment was background, we considered the percentage of color in red bins 4 through 6, green bins 6 and 7, and blue bins 3 through 5. If each grouping contained more than 25% of the pixels in the respective color histogram, then the fragment under consideration was deemed to be background. We developed other histogram rules applied in a similar fashion identifying animal or background as appropriate. Night images were identified by the camera's use of an infrared flash and histogram rules were disabled for such images since night images were captured in greyscale.

The last step, compute region counts, assigned a crossing confidence between one and five, as described in the previous section.

Program accuracy

We compared the results of our semi-automated approach for the 480,542 input images with results of manually processing the same images. We considered two metrics, removed image percentage (R.I.P.) and retained crossing percentage (R.C.P.). The fraction of removed images was determined from a count of the retained images divided by the number of input images, as shown in Eq. 1.

$$\text{R.I.P.} = \left(1 - \frac{\#\text{ of retained images}}{\#\text{ of input images}}\right) \times 100\% \qquad (1)$$

If an image set is determined to contain a crossing then all images in the set are included in the count of retained images.

The retained crossing is determined from the images manually identified to contain crossings. The manually identified crossing applies to an individual image, where the automated program marked sets as crossings. Consequently, the number of retained crossings is the number of manually identified crossing images that are in a set of images identified as a crossing set. A crossing image identified by the program is only considered a retained crossing if it was also manually identified as a crossing.

$$\text{R.C.P.} = \left(\frac{\#\text{ of retained crossing images}}{\#\text{ of manually identified crossings}}\right) \times 100\% \quad (2)$$

Consequently, the performance of the program improves as both the R.I.P. and R.C.P. increase.

Results

We found that, from the initial 480,542 images, the semi-automated program was able to reduce the total number of images that must be classified by hand by over half (R.I.P. of 54.8%, Table 1). Of the 10,197 crossing events detected manually, the R.C.P (images classified into category 1, 2, or 3 corresponding to those with the highest crossing probability) was 72.6%. The remaining crossing events were contained within the category 4 and 5 images (those with the lowest crossing probability), which were not retained for manual processing (Table 1). The efficacy of the program varied by site (Table 2). Across the 12 sites,

Table 1 Program classification of all images and classification of manual identified crossings

Category	Images count	Crossing count
1	175,793	6613
2	26,853	606
3	7645	180
4	6865	248
5	263,386	2550

Table 2 Crossing assignments by site of manually identified crossings

Site	Category					Percentage
	1	2	3	4	5	retained
1	933	29	32	89	352	69.3%
2	201	29	10	3	320	42.6%
3	419	74	26	2	197	72.3%
4	1378	105	32	44	342	79.7%
5	155	11	1	12	52	72.3%
6	32	16	4	2	54	48.1%
7	370	62	8	3	152	73.9%
8	179	54	9	17	278	45.1%
9	312	27	11	24	110	72.3%
10	296	51	6	31	73	77.2%
11	1357	126	27	9	412	78.2%
12	981	22	14	12	208	82.2%

the average R.C.P was 67.7%. Day and night time results have similar retained percentages, suggesting the method is adequate for night without further adjusting the input parameters. When processing night time images, 4756 of the 6404 crossings were retained for a R.C.P of 74.3%. Day time images were retained with a R.C.P. of 69.7%. The retained crossings by month are shown in Table 3. Averaging the monthly R.C.P. yields an R.C.P of 67.1%. Considering only the months June through September yeilds an average R.C.P. of 49.9% while averaging the remaining months yields an average R.C.P of 76.4%.

Table 3 Retained crossings by month

Month	Total	Retained	Percentage retained
January	1068	836	78.3%
February	442	305	69.0%
March	329	270	82.1%
April	1081	878	81.2%
May	1087	790	72.7%
June	402	195	48.5%
July	477	255	53.5%
August	351	164	46.7%
September	238	119	50.0%
October	1686	1285	76.2%
November	2103	1671	79.5%
December	933	631	67.6%

In total, the reduction in the number of images that required manual processing following identification from the semi-automated method resulted in a reduction in operator time by a factor of approximately six.

The sites corresponding to the lowest R.C.P are shown in Fig. 2 along with an image from a site with a high percentage retained crossing. As can be seen in the images, the low percentage of retained crossings can correspond to images where the fence is perpendicualar to the camera, or off center making marking fence regions difficult.

Discussion and conclusions

Our results show a reduction in the number of images (54.8%) requiring human processing while retaining the majority of crossings (72.6%). These numbers were satisfactory for our purpose and significantly decreased the resulting manual image processing time for a larger data set of 480,542 images. Adjusting the settings (e.g., thresholds) and sequence of image processing instructions can change the trade-off between the number of images removed and the number of retained crossings. Images from our testing show some sites with a reduced retention of crossings, keeping less than half. Causes for the low accuracy were attributed to site differences including camera placement and immediate landscape characteristics.

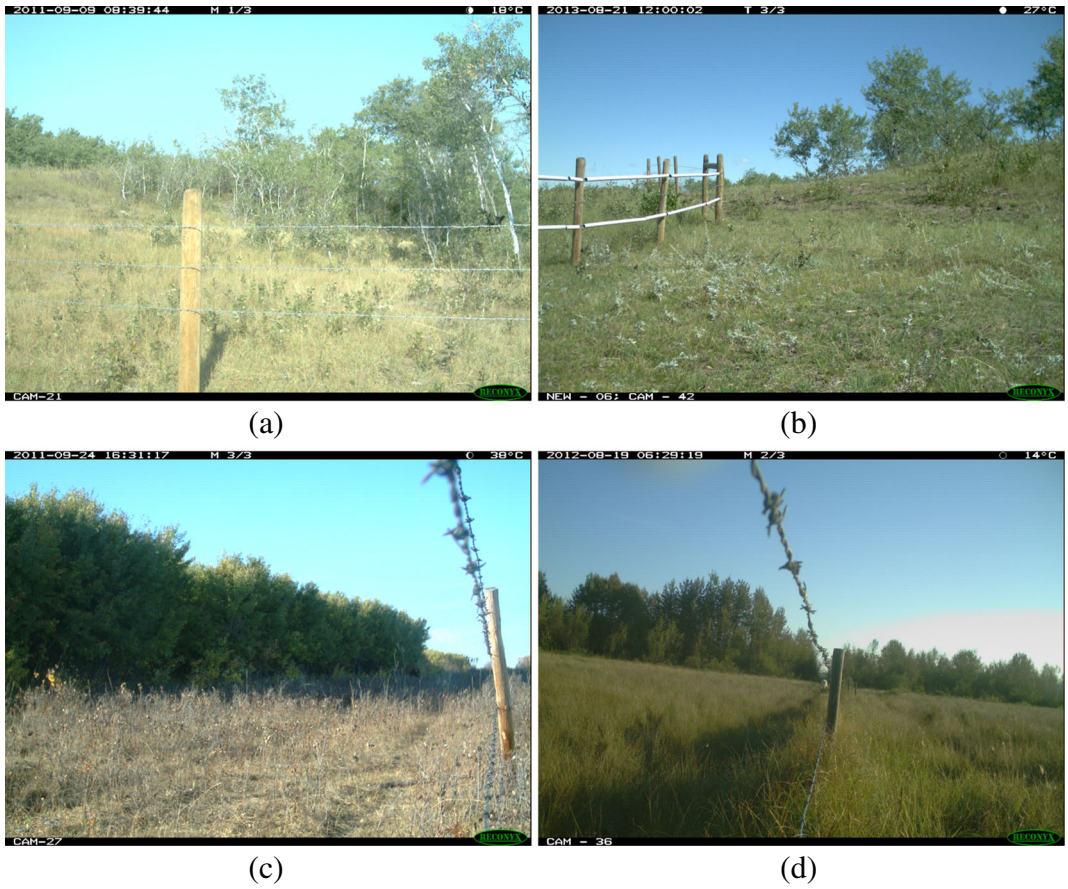

Fig. 2 Three images with poor camera placement (**a**, **b**, **c**) versus one image with a good placement (**d**)

In particular, issues with the input images included the presence of fog, fence location relative to the camera placement, and the camera toppling over during the recording process. In regards to camera placement, our method works best when the camera is placed on the fence resulting in clear left and right regions for animals to enter and leave. In some site locations, the camera was pointed at the fence such that the animal could approach from behind the fence, without passing through an entering or leaving region. Figure 2 shows three images from three site locations with poor camera placement and one image from a site with good camera placement.

While our program removes over half of the images, this number under represents our program's ability, as the images we received had already been pre-processed. The cameras had taken approximately three million images, and Alberta Parks staff had already removed some images they identified as having only cows that did not cross the fence, or images where they identified background noise. Despite the concentration of crossings in the image data we received, our program was able to further concentrate the number of crossings in the retained images demonstrating itself as a useful tool in the project. We plan to analyze our program's performance on raw camera trap data in the future. Another factor that increases the number of retained images is keeping an entire image sequence for a crossing event. Ideally, a crossing could be identified from only two images, one on each side

of the fence. Our program, however, keeps the entire sequence for manual processing, raising the number of images retained. For this project, keeping entire image sequences for a unique event was beneficial for manual processing, but future improvement work will identify the most relevant images to further improve the R.I.P.

While our approach is tailored to a specific problem, it can be generalized to camera-trap instances without a linear feature by adding an arbitrary point of crossings, or by only removing noise distributed across the entire image, such as wind and clouds. Currently, the software can execute without an actual fence simply by specifying three regions of interest. When an animal passes through these regions it will trigger a crossing event. There are many such cases where this is useful, such as animals crossing linear features (fences, roads, ditches, etc.). In cases where animals are attracted to a particular feature, such as a water source, the program could be run twice on the image sets, once to detect animals entering from the left, and the other to detect animals entering from the right. Multiple detections of the same image sets will not result in additional manual processing since the images retained have the same filename, and only one copy will be kept.

If users want to use the program to remove noise distributed across the entire image the user can set the program to retain images sets classified into categories one, two, three, and five. This will retain all detected animals while removing image sets that triggered the camera trap due noise such as wind, or lighting changes due to clouds. In future work, we plan to further consider situations without a linear feature to cross. For example, we will try to determine animal movement within the image to approximate the animal direction, or number of animals.

Research in other papers addresses similar challenges in our work, chiefly identifying target species from images while removing images sequences containing noise. Swinnen et al. (2014) tracked beavers using heat and motion activated video camera traps, while Weinstein (2015) tracked humming birds using video recorded at one frame per second. Swinnen et al. (2014) generate their background image using two methods. The first averaged pixels across time, similar to our method. The second compares pixels in subsequent frames and retains frames when the pixel difference is large. Similar to our work, Swinnen et al. (2014) acknowledge that their system may remove non-moving animals, but that these animals had to move to enter the image region. They mention their system can work with camera trap images but acknowledge the challenge of changing conditions such as vegetation and lighting changes.

Weinstein (2015) used running video as input and retained video sequences with motion. As with our work, pixels were classified as foreground or background and thresholded foreground pixels form fragments, which he called blobs. Frames containing a large enough fragment were retained while others were discarded. As with our input images, video conditions in Weinstein's work (2015) can change including changes caused by rain, wind, and sunlight.

Our work adapts and improves the techniques from these two papers in several ways. The largest difference is our use of camera trap images rather than video sequences. While a camera trap is cheaper and more convenient to maintain, the differences between subsequent images are larger than subsequent video frames. Thus comparing only subsequent frames as used by Swinnen et al. (2014) is not directly applicable to camera trap images. Additionally, with video each unique event is recorded in a large number of frames. Averaging a large number of temporally nearby frames to form a background image reduces the effects of lighting changes and ghosting in the background image. With the potential of few input images per unique event, our program reduces the effects of ghosting by selecting pixels within one standard deviation of the pixels in the same location from different images. Lighting changes were handled by gamma adjusting the images to have an equivalent image intensity for all images in an image sequence. Another adaptation in our work selects animals exhibiting a particular behavior. While the two papers described above attempted to keep all frames with the target species, we were only interested in animals that crossed a linear feature. Thus, we defined three regions around the fence to identify movement between these regions. This allowed us to remove animals that approached, but did not cross the fence. The techniques described in the

previous work do not account for target animals that should not be retained for manual processing, as their experiment did not identify linear features of interest. As a last improvement to prior work, our system removes fragments based on their color histogram in addition to requiring a fragment to contain a sufficient number of pixels. Weinstein (2015) only considered fragment size while Swinnen et al. (2014) did not consider color to identify background or target species, possibly since the videos were primarily captured at night using infrared.

In terms of results, the program from Swinnen et al. can remove 53 to 76% of non-target recordings with a 5 to 20% loss of images containing the target species. These values, however, are considered under ideal circumstances. Additionally, beavers are large and slow moving, where ungulates in our program may be fast and far from the camera making detecting ungulates more challenging. Under ideal circumstances, our system had a loss of images containing target species of approximately 18% (March numbers from Table 3). Without adjusting the sequence of processing instructions, our system removes an average of 54.8% with a loss of images containing target species at 27.4%. With optimal parameters, the system from Weinstein (2015) returned 5% of hand counted frames. As mentioned above, both systems we compare against use video input where our system uses a sequence of still images, and consequently neither of the results are directly comparable. In particular, Weinstein (2015) uses running video, and likely contained a large number of frames without a target animal present, making result comparisons less applicable since many frames likely have no movement present.

In conclusion, we present a software system that assists in detecting ungulate crossing events from camera trap images. Use of a camera trap is less expensive and requires lower maintenance than using video as input, thus being more valuable for monitoring in remote sites. It requires improvements, however, from previous systems to suitably process the still images. Our system functions with little user input, other than identifying the fence location at each site images were captured. We use a modified version of background subtraction to determine ungulate crossings of a linear feature, and a collection of color histogram rules to remove non-target fragments arising from noise. Our system can be applied to other camera trap settings with a linear feature, or can be adapted to settings without a linear feature by specifying an arbitrary crossing point in the image. Use of our system improved the throughput of manual categorization by a factor of approximately six, while retaining 72.6% of the crossing events.

Acknowledgments We thank Alberta Parks for funding for this project and for helping with field setup and data collection and the King's University and the King's Centre for Visualization in Science for funding and support. This work, in part, comes from the undergraduate research projects of K. Visser and I. MacLeod. We also thank Dr. Jose Alexander Elvir and an anonymous reviewer for their comments.

Appendix I

Software availability: http://cs.kingsu.ca/mjanzen/CameraTrapSoftware.html.

Example instructions for running the program on a Windows computer are also available from this website.

Appendix II

Additional sequences of images showing a false negative event, Fig. 3, and false positive event, Fig. 4. In Fig. 3, the camera trap takes its first picture after the moose has already entered the field of view, and the moose overlaps in images causing the overlap to appear as part of the background image. In Fig. 4, movement of the fence or camera causes an image change to be detected, which is retained in one of the three images. This causes the program to classify the image set into category two, implying a fast moving animal detected in only one of the three images.

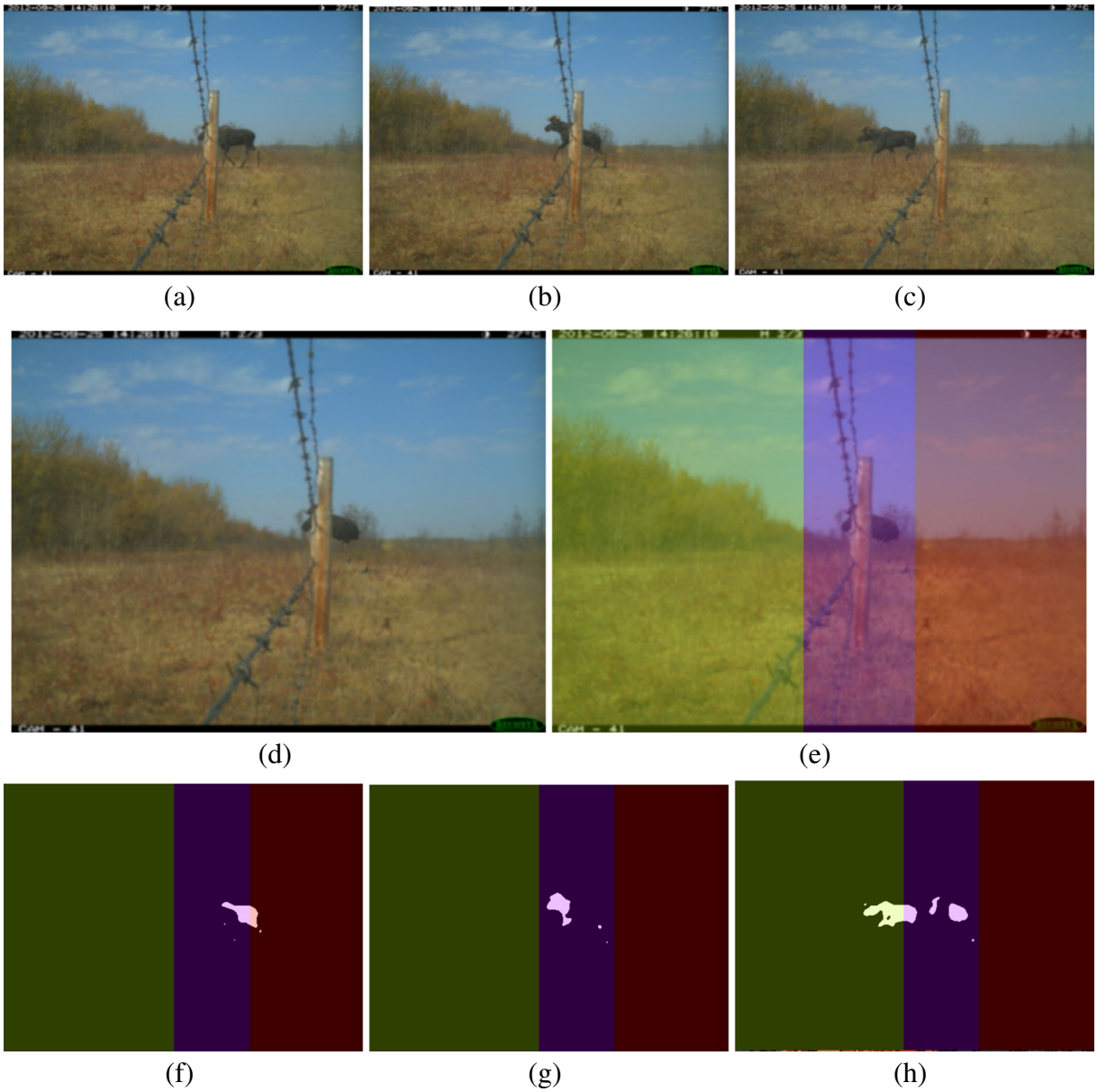

Fig. 3 An sequence of processing steps resulting in a false negative where the moose is not suitably detected

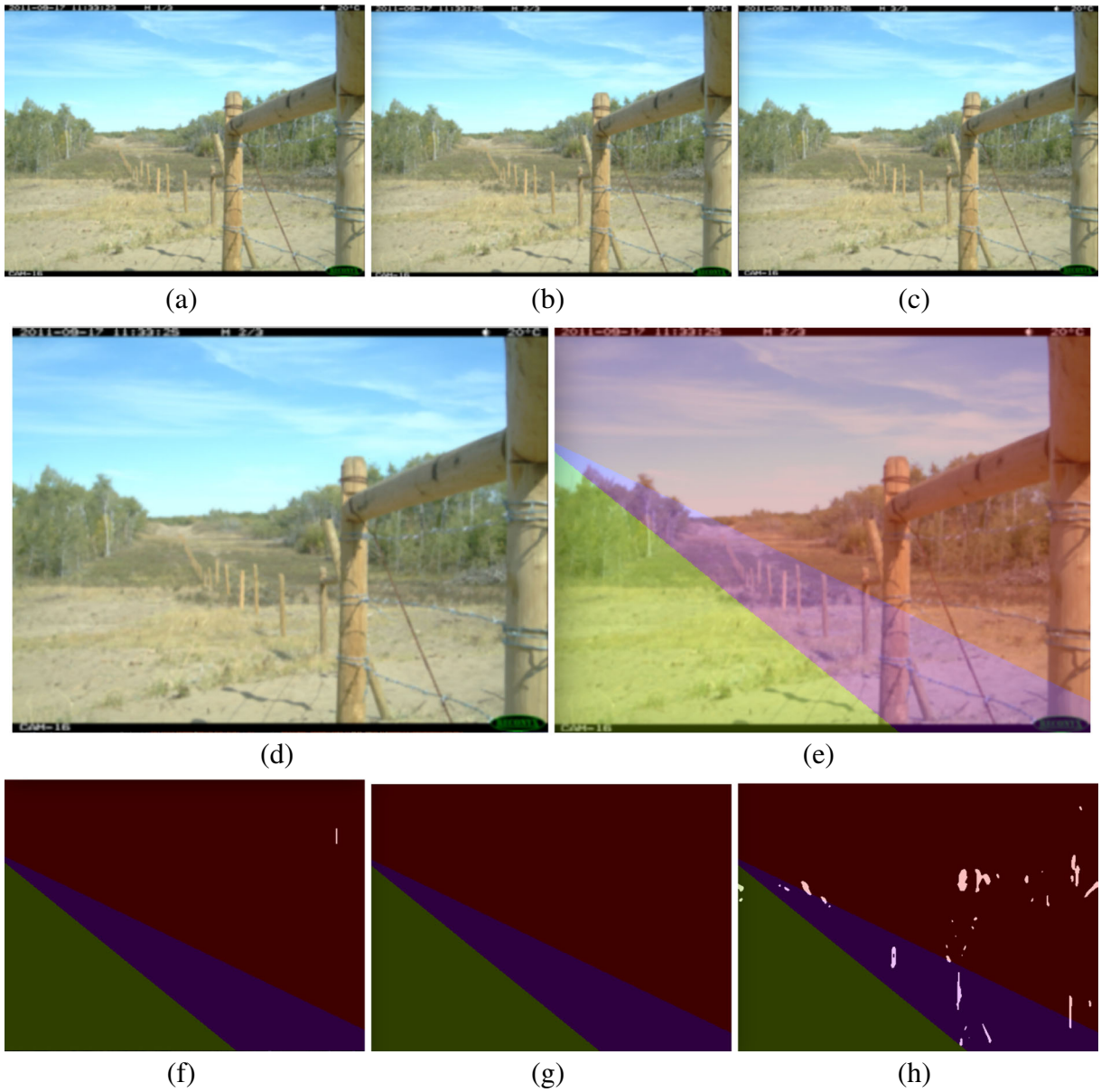

Fig. 4 A sequence of processing steps resulting in a false positive, classifying the image set as a fast moving animal

References

Anderson, C.J., Lobo, N.D.V., Roth, J.D., & Waterman, J.M. (2010). Computer-aided photo-identification system with an application to polar bears based on whisker spot patterns. *Journal of Mammalogy*, *91*, 1350–1359.

Ardekani, R., Greenwood, A.K., Peichel, C.L., & Tavaré, S. (2013). Automated quantification of the schooling behaviour of sticklebacks. *EURASIP Journal on Image and Video Processing*, *2013*, 1–8.

Bubnicki, J.W., Churski, M., & Kuijper, D.P. (2016). *TRAPPER: an open source web-based application to manage camera trapping projects*. Methods in Ecology and Evolution.

Burton, A.C., Neilson, E., Moreira, D., Ladle, A., Steenweg, R., Fisher, J.T., Bayne, E., & Boutin, S. (2015). REVIEW: Wildlife camera trapping: a review and recommendations for linking surveys to ecological processes. *Journal of Applied Ecology*, *52*, 675–685.

Efford, M. (2004). Density estimation in live-trapping studies. *Oikos*, *106*, 598–610.

Forrester, T., McShea, W.J., Keys, R., Costello, R., Baker, M., & Parsons, A. (2013). *emammal–citizen science camera trapping as a solution for broad-scale, long-term monitoring of wildlife populations*. Sustainable Pathways: Learning from the Past and Shaping the Future.

Goehner, K., Desell, T., Eckroad, R., Mohsenian, L., Burr, P., Caswell, N., Andes, A., & Ellis-Felege, S. (2015). A comparison of background subtraction algorithms for detecting avian nesting events in uncontrolled outdoor video. In *2015 IEEE 11th international conference on e-science (e-Science)* (pp. 187–195). IEEE.

Gonzalez, R.C., & Woods, R.E. (2007). *Digital image processing*. New Jersey: Prentice Hall.

Hines, G., Swanson, A., Kosmala, M., & Lintott, C.J. (2015). Aggregating user input in ecology citizen science projects. In *Proceedings of the twenty-seventh innovative applications of artificial intelligence conference* (pp. 3975–3980). AAAI.

Jhala, Y.V., Mukherjee, S., Shah, N., Chauhan, K.S., Dave, C.V., Meena, V., & Banerjee, K. (2009). Home range and habitat preference of female lions (Panthera leo persica) in Gir forests, India. *Biodiversity and Conservation*, *18*, 3383–3394.

Krishnappa, Y.S., & Turner, W.C. (2014). Software for minimalistic data management in large camera trap studies. *Ecological Informatics*, *24*, 11–16.

Meek, P., Ballard, G., Claridge, A., Kays, R., Moseby, K., OBrien, T., OConnell, A., Sanderson, J., Swann, D., Tobler, M. et al. (2014). Recommended guiding principles for reporting on camera trapping research. *Biodiversity and Conservation*, *23*, 2321–2343.

Osterrieder, S.K., Kent, C.S., Anderson, C.J., Parnum, I.M., & Robinson, R.W. (2015). Whisker spot patterns: a noninvasive method of individual identification of Australian sea lions (Neophoca cinerea). *Journal of Mammalogy*, *96*, 988–997.

Piccardi, M. (2004). Background subtraction techniques: a review. In *2004 IEEE international conference on systems, man and cybernetics*, (Vol. 4, pp. 3099–3104). IEEE.

Sirén, A.P., Pekins, P.J., Abdu, P.L., & Ducey, M.J. (2016). Identification and density estimation of american martens (Martes americana) using a novel camera-trap method. *Diversity*, *8*, 3.

Spampinato, C., Farinella, G.M., Boom, B., Mezaris, V., Betke, M., & Fisher, R.B. (2015). Special issue on animal and insect behaviour understanding in image sequences. *EURASIP Journal on Image and Video Processing*, *2015*, 1.

Swinnen, K.R., Reijniers, J., Breno, M., & Leirs, H. (2014). A novel method to reduce time investment when processing videos from camera trap studies. *PloS one*, *9*, e98,881.

Tobler, M.W., Zúñiga Hartley, A., Carrillo-Percastegui, S.E., & Powell, G.V. (2015). Spatiotemporal hierarchical modelling of species richness and occupancy using camera trap data. *Journal of Applied Ecology*, *52*, 413–421.

Visscher, D.R., MacLeod, I., Vujnovic, K., Vujnovic, D., & DeWitt, P.D. (2017). Human induced behavioural shifts in refuge use by elk in an agricultural matrix. *Wildlife Society Bulletin*, *41*, 162–169.

Weinstein, B.G. (2015). Motionmeerkat: integrating motion video detection and ecological monitoring. *Methods in Ecology and Evolution*, *6*, 357–362.

Yu, X., Wang, J., Kays, R., Jansen, P.A., Wang, T., & Huang, T. (2013). Automated identification of animal species in camera trap images. *EURASIP Journal on Image and Video Processing*, *2013*, 52–53.