Application of object tracking in video recordings to the observation of mice in the wild

By: Matina Kalcounis-Rueppell, Thomas Parrish, and Sebastian Pauli

Kalcounis-Rueppell M., Parrish T., Pauli S. (2013) Application of Object Tracking in Video Recordings to the Observation of Mice in the Wild. In: Rychtář J., Gupta S., Shivaji R., Chhetri M. (eds) Topics from the 8th Annual UNCG Regional Mathematics and Statistics Conference. Springer Proceedings in Mathematics & Statistics, vol 64. Springer, New York, NY.

Made available courtesy of Springer: https://doi.org/10.1007/978-1-4614-9332-7_11

Abstract:

We give an overview of methods used to track moving objects in video and describe how information about animal behavior can be extracted from tracking data. We discuss how computer-aided observation can be used to identify and pre-select potentially interesting video sequences from large amounts of video data for further observation, as well as directly analyze extracted data. We examine how this analysis can be used to study animal behavior. As an example, we examine thermal video recorded from free-living, nocturnal, wild mice in the genus *Peromyscus*.

Keywords: Foreground Object | Video Analysis | Animal Behavior | Background Movement | Computer Vision Technique

Article:

1 Introduction

Methods for the automating the processing of digital video have been a topic of research since the mid-1980s [1]. These techniques have been used extensively in traffic surveillance and security. In the past decade, automated analysis of video has become increasingly popular in the study of animal behavior, both in the laboratory and in the wild. For example, the individual and social behaviors of fruit flies in a planar arena in a laboratory setting have been quantified using data obtained with computer vision methods [4].

As part of a larger study examining vocal communication among wild deer mice (*Peromyscus* species) [6] infrared video was collected over 131 nights from dusk until dawn. The video was taken from a camera suspended in the tree canopy above the free-living mice on the forest floor. The video was recorded nonstop, regardless of the level of mouse activity. Thus, the volume of video recordings obtained in this study is a challenge to manually process. Computer vision techniques, however, allow us to detect and record the trajectories of moving objects from the video data without human intervention. In the initial phase of the project, mouse trajectories were extracted from short clips of the video recordings with the goal of analyzing the speed of mice [13] and data extracted from the video was validated by a human observer [2]. As the result of this experience we are now able to process the approximately 1,500 h of video and extract biologically meaningful data.

In this paper we report on the methods we used to track the movement of mice in video material and describe how we obtained biologically relevant information from the tracking data, namely measures of mouse activity. The results of our analysis are subject of a forthcoming publication by the authors.

1.1 Notation

We will use the following notation in our discussion of video and image data. We represent an image as an $m \times n$ matrix $F \in C^{m \times n}$, where C denotes a color space. We denote the (x, y) entry in F by $F_{x,y}$ and refer to it as a picture element, or *pixel*.

Common examples of color spaces are black and white ($C_0 = \{0, 1\}$), grayscale (C_g = {0,…,255}), and true color (C_t ={(R,G,B)|R,G,B∈{0,…,255}}). For ease of presentation we will limit most of our discussions to grayscale images and video. It can be easily generalized to other color spaces.

A video V is a sequence of images,

$$V = F_1, F_2, \ldots, F_n \tag{1}$$

where $n \in \mathrm{N}$ is the number of images in the video. Each image is called a frame, and those frames are displayed at a constant frame rate, which is typically 24, 25, or 30 frames per second.

2 Foreground Isolation

One of the most fundamental applications of automated video processing is the identification and tracking of moving objects. The most common tracking method is referred to as *blob tracking*. This process involves isolating foreground from background information by means of background subtraction, identifying foreground *connected components*, or collections of adjoined pixels, and tracking those over time.

For our purposes, we consider each pixel of a video image to belong to either the foreground or background. We define the background to be the set of static, or predominantly unchanging pixels, and the foreground to be the set of all other pixels.

The foreground isolation functions return a black and white image M called the foreground mask. A pixel of value $M_{x,y} = 0$, or black, represents a background pixel, and a pixel of value $M_{x,y} = 1$, or white, corresponds to a foreground pixel. We call elements of the foreground objects, and their corresponding foreground mask elements blobs.

2.1 Background Subtraction

If the pixels corresponding to the background are known, then the foreground can be extracted by taking the absolute *difference* $s(F, B)$ of a frame F and a reference background image B, where

$$s : C^{m\times n} \times C^{m\times n} \to C^{m\times n}, s(F,B) = G \text{ where } G_{x,y} = \left|F_{x,y} - B_{x,y}\right| \qquad (2)$$

Clearly if $s(F, B)_{x,y} = 0$, then $F_{x,y}$ belongs to the background. Because we want to allow some fluctuation in the background pixels a threshold function is used to decide whether a pixel belongs to the foreground or background:

$$t : C_g^{m\times n} \times C_g \to C_0^{m\times n}, t(F,c) = G \text{ where } G_{x,y} = \begin{cases} 0 \text{ if } F_{x,y} < c \\ 1 \quad \text{else.} \end{cases} \qquad (3)$$

For each frame F, if $F' = s(F, B)$, then the foreground mask can be given by $t(F', c)$, where c is typically greater than 200 for grayscale images.

There are various methods for determining the background image, which can be static or updated with every frame, for example:

First Frame Method. If the first frame of the video only consists of background, the first frame can be used as the background image. This yields the fastest background subtraction method.

Average Frame Method. The average of all frames of the video is used as a background image. This can work even if objects are present in the foreground of all frames, as long as those objects move frequently. Because the entire video must be processed prior to tracking, this method does not allow video processing in real time.

Running Average of Frames Method. Using the running (weighted) average of all previous frames as the background image yields better results, particularly when there are frequent subtle changes in lighting. Typically, the background B is initialized to the first frame F_0, and after processing each subsequent frame F, B is updated to $w_\alpha(F, B)$, where

$$w_\alpha : C^{m\times n} \times C^{m\times n} \to C^{m\times n}, \left(w_\alpha(F,B)\right)_{x,y} = \left\lfloor \alpha F_{x,y} + (1-\alpha)B_{x,y} \right\rfloor \qquad (4)$$

for some $\alpha \in (0, 1)$.

2.2 Dilation and Erosion

Often, a foreground pixel is similar in intensity or color to the corresponding background pixel. In this case, the foreground pixel is likely to be improperly classified as a background pixel. This can result in hole within a connected component, or two distinct connected components that represent the same object.

To prevent such errors, a series of morphological operations can be applied namely *dilation* and *erosion*. Dilation increases the size of blobs, merging blobs that represent the same object and removing holes. Erosion reduces the size of blobs and smoothes the edges. These operations are often combined with foreground isolation techniques.

In each operation, the value of a pixel $F_{x,y}$ is set to either the lightest or darkest pixel value in the neighborhood specified by a *kernel*. The kernel can be described as a set of relative coordinates $K⊂Z×Z$. The neighborhood of $F_{x,y}$ specified by K consists of the pixels with coordinates in $\{(x+i,y+j)|(i,j)∈K\}$. The dilation of an image F using the Kernel K is

$$d_K : C^{m\times n} \to C^{m\times n}, d_K(F) = G \text{ with } G_{x,y} = \max\{F_{x+i,y+j} | (i.j) \in K\} \qquad (5)$$

The erosion of F using the Kernel K is

$$c_K : C^{m\times n} \to C^{m\times n}, c_K(F) = G \text{ with } G_{x,y} = \min\{F_{x+i,y+j} | (i.j) \in K\} \qquad (6)$$

It is common to choose a simple kernel, such as $K = \{(i,j) \mid i,j \in \{-1,0,1\} \}$.

Typically, a series of dilation and erosion operations are applied to the foreground mask in the form of *open* and *close* operations, where opening is the dilation of an erosion, and closing is the erosion of a dilation. Both opening and closing will result in blobs very close to their original size.

2.3 An Advanced Method

More often than not, however, videos of interest will not contain a stationary background. In such cases, it is necessary to seek more intelligent methods of distinguishing foreground pixels from background pixels. The method chosen for our application, developed by Liyuan Li, Weimin Huang, Irene Y.H. Gu, and Qi Tian, uses a Bayes decision rule to classify objects as foreground and background [9]. It is designed to accommodate two types of changes in background state: gradual changes, such as changes in natural lighting, and rapid changes, such as a camera rotation or tree branch movement. Stationary background pixels are classified by their color features, while moving background elements are classified by their color co-occurrence features. The algorithm consists of four steps: detection of state changes, classification of state changes, foreground object identification, and background learning and maintenance. For each frame, the following steps are executed:

1. Generate background model
2. Perform simple background subtraction to remove pixels of insignificant change
3. Classify each remaining pixel as stationary or moving
4. If stationary, compare pixel value with learned color states and use a Bayes rule to determine probability of being foreground
5. If a pixel is classified as moving, compare color co-occurrence, along with color to the set of learned states, and use Bayes rule to determine probability of being foreground
6. Assign pixel to foreground or background accordingly

7. Perform a pair of dilate–erode and erode–dilate operations to remove artifacts and connect blobs
8. Update the set of learned color states and color co-occurrence states
9. Update the reference background image

3 Component Identification and Labeling

In order to identify specific elements of an image, it is important to identify the connected components, which exist as sets of *neighboring* pixels. In this application, two pixels are considered neighbors if the distance between them is less than or equal to $\sqrt{2}$ pixels.

One way to identify objects is to use component *contours* as the primary identifying feature of each object. An object's contour is its set of edge pixels.

A simple method of identifying and labeling components in an image $F \in C^{m \times n}$ involves generating an associated label image, $L \in \mathrm{N}^{\mathrm{m} \times \mathrm{n}}$, with each pixel $L_{x,y}$ consisting of the label corresponding to the pixel $F_{x,y}$. An extremely efficient method, proposed by Fu Chang, Chun-Jen Chen, and Chi-Jen Lu can be used for this task [7].

In the algorithm they present, an image F is processed left to right, and top to bottom. When an external contour pixel is encountered, the entire contour is traced and, for each pixel $F_{x,y}$ in the contour, we set $L_{x,y} = l$, where $l \in \mathrm{N}$ is unique to this contour. Once the contour has been traced, foreground pixels inside the contour are also labeled l. If an *internal* contour point is reached, the internal contour is again traced, and labeled l. When a new external contour pixel is found, it is labeled $l + 1$, and the tracing process repeats. Each set of pixels of the same label is referred to as a blob.

4 Blob Tracking

In each frame, blobs are labeled by order of detection, making it difficult to ensure a unique label preservation between frames. Because of this, a blob will often have many labels over time, some of which may correspond to labels assigned to other blobs. It is then necessary to check each successive frame and ensure that for any given blob, its label in the current frame corresponds to its label in the previous frame. There are a number of methods to accomplish this. One simple approach is to calculate a set of identifying features, such as size, location, location of centroid, orientation, intensity or color for each blob. After labels are assigned in each frame, the features of each blob are compared to those of every blob in the previous frame that is within a set distance, and labels are re-assigned accordingly. The set of features for each blob can then be output as track information, sorted by blob label.

4.1 Tracking Data

Because video frames are processed sequentially, blob data generated by the tracker are returned in sequential order. After each frame, the tracker returns data for each blob, consisting of the unique label of the blob (not to be confused with the labels of the components in the frame), its position, its size, and the number of frames the blob has been present. Additional information, such as bounding boxes, histogram information (of use in color video), velocity and acceleration vectors, can also be extracted. However, because it would require inference, rather than direct observation, to generate velocity and acceleration data, introducing uncertainty, these data were not produced. In addition, because the thermal videos are in greyscale, color information was ignored.

5 Object Tracking in the Mouse Videos

We describe the video material with which we worked, how the tracking was done, and discuss some challenges we encountered and some decisions we needed to make to obtain as much usable data as possible.

The videos were recorded during research where audio, video, and telemetry data were used to analyze the ultrasonic vocalizations of two species of free-living mice, *Peromyscus californicus* and *P. boylii*. The fieldwork took place over 131 nights at the Hastings Natural History Reservation in upper Carmel Valley, California, USA, during the winters of 2008 and 2009. A detailed description of the methods, with example data representing audio, video, and telemetry, can be found in [6].

5.1 The Mouse Videos

A thermal-imaging camera was suspended by a simple pulley system in the tree canopy approximately 10 m above the ground, allowing continued recording of active mice in the field of view, through the night. The camera used was a Flir Photon 320 with a resolution of 320 by 240 pixels at 30 frames per second in grayscale. The video was recorded with a JVC Everio GZ-MG 555 hard disk camcorder connected to the camera with a composite video cable at an upscaled resolution of 720 by 480 pixels. In the following all pixel measures refer to pixels in the recorded video.

5.2 Our Implementation

Previously available animal tracking software was primarily designed for the analysis of animal behavior in a laboratory setting [5, 8], with animals moving in front of a stable background. This specialization makes them less suitable for processing videos of animals in natural environments, where lighting changes and background movement occur frequently. Moreover, many relevant behaviors will be seen in natural environments without a stable background.

For this reason, we wrote a tracking program based on the C++ libraries OpenCV [12] and cvBlob [10], which are freely available under a BSD licence and the LGPL, respectively. OpenCV provided implementations of the algorithms needed for the foreground identification (where we used the advance method described in Sect. 2) and the image clean-up steps.

Because of the small size of the mice (about 40 square pixels in the upscaled resolution, 10 square pixels at camera resolution) we use two dilation and no erosion steps in the image cleanup after foreground identification.

The foreground isolation and clean-up steps are illustrated in Fig. 1.

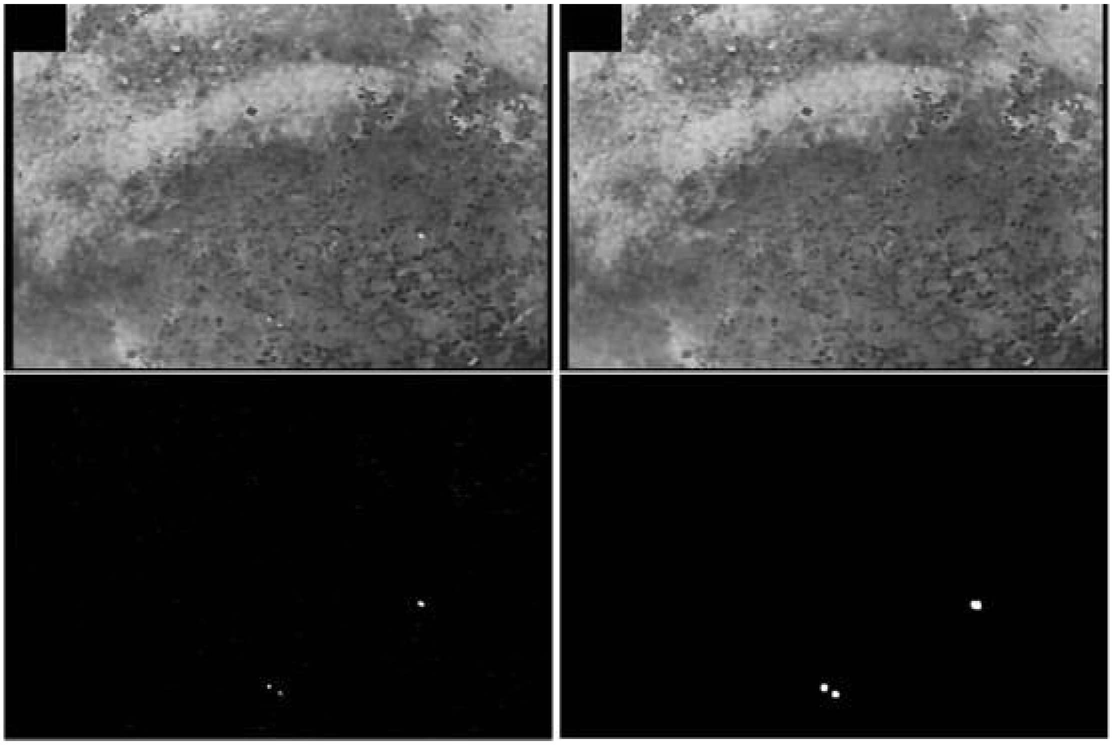

Fig. 1. The four images show a still from an infrared video, background image, the foreground mask, and the foreground mask after dilating twice. The two blobs on the left are partially concealed mice, the blob on the right is another mouse

The library cvBlob offered the functionality needed for the blob tracking step, including an implementation of the block labeling algorithm described in Sect. 3. We found that the simple blob tracking methods implemented in cvBlob were sufficient for our application.

For OpenCV and cvBlob installation instructions, see the web sites given in the references. For an introduction to OpenCV, see the OpenCV book [3]. The functions for the post processing were written using the Python-based computer algebra system Sage. The blob tracking program outputs tracking information in the form of a raw text, which is imported into Sage and processed.

A shell script calling the video processing and post processing was written, allowing several hundred videos to be processed in one batch.

5.3 Data Filtering

Although the program is able to disregard most noise, some noise may be categorized as legitimate foreground information. However, these false tracks typically have very short durations. For this reason, we have chosen to ignore tracks of extremely short duration, which we classify as tracks less than ten frames long, or one third of a second. It is also the case that a warm wind will occasionally heat up a stationary background element, such as a rock or mouse trap, for a time longer than ten frames. To account for these false tracks, we discard any track for which there is no movement.

5.4 Blob Classification

Once the tracks are filtered, blobs are categorized based on size and speed. For mice, we calculated an expected size based on known biological size ranges, which we converted to a pixel area based on the dimensions of each focal area. Because these dimensions varied across focal areas, we used a separate range for each area. In addition, we found that bats and birds traveled significantly faster than mice. Any object that traveled faster than three pixels per frame was considered to be a flying vertebrate.

6 Analysis of Tracks

We used the tracking information in two ways. In the first application, which we refer to as *computer-aided observation*, data were searched for information that targets specific events of interest to human investigators, who then analyzed these events.

In the second application, which we refer to as *automated analysis*, the computer directly computes data, which can then be used for the (statistical) analysis of behavior.

6.1 Computer-Aided Observation

Computer-aided observation is useful for finding specific events which require qualitative analysis. An example of such an application may be to have the computer extract all times in a video when several objects exist in concurrence. The investigator could then watch the video, in order to determine if the objects (animals) influence each other’s behavior.

A script was written to report all times when objects of specific size ranges appear in videos. These size ranges were used for two purposes. We used them to find predators such as cougars (*Puma concolor*), bobcat (*Lynx rufus*), and foxes (*Urocyon cinereoargenteus*), by searching for large blobs, which had an area greater than 500 square pixels. The times when large blobs were present were used as a queues for manual observation, so that these blobs could be classified and behaviors analyzed.

We also returned all times when objects in the expected size range of mice (80–120 square pixels after dilation, depending on focal area) existed for a period of at least 5 s. From this list, we selected a random sample of videos and times and observed the videos. In all cases, we found that the blobs in our expected size ranges corresponded to mice.

6.2 Automated Analysis

Although computer-aided observation is a valuable tool, it is desirable for the computer to do as much analysis as possible. While the analysis of complex events and interactions is difficult, some data lend itself to easy analysis. Examples of such data include analysis of size distributions, speed of travel, and location preference (i.e., objects do have a tendency to be found in one region more often than another). Our primary application of automated analysis was to analyze levels of mouse activity.

6.3 Measuring Mouse Activity

Often mice exit and reenter the field of view, or become temporarily masked under dense vegetation. Because of the uncertainty introduced by these events, a decision was made to use only observed data, and to not interpolate missing data. In addition, accurate identification of individuals is difficult due to a lack of identifying features in thermal video. As such, measures of activity that do not require the identification of individual mice were chosen. In this way we avoid introducing unnecessary error.

Assume that a track is active from frame number *m* to frame number *n*. Let (x_t, y_t) be the position of a blob, at frame number *t*. Because of the high sampling rate of the position of the blob at 30 times per second

$$d = \sum_{t=m}^{n} \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \tag{7}$$

is a good approximation of the length of the track.

To measure the activity of mice on a given night, we use two values:

1. the total observed distance *D* travelled by all mice throughout the night; that is, the sum of the lengths of all observed tracks and
2. the average speed *S* of all mice throughout the night; that is, S=D/TS=D/T where *T* is the sum of the lengths in time of all observed tracks.

These measures make it possible to investigate the change in mouse activity under various biotic and abiotic environmental influences. This investigation is subject of a forthcoming publication of the authors.

7 Conclusion

Automated tracking is remarkably useful. With a limited understanding of computer vision techniques and moderate computer programming experience, it is possible to construct an automated video processing program suitable for analyzing some types of animal behavior. The results obtained from these types of programs, e.g. tracking information, help us to answer numerous biological questions and save researchers a great deal of time. Useful information can often be obtained from even poor quality video.

Some caveats exist, however. For example, it is difficult to distinguish amongst individuals in grayscale video. Also, it is difficult to extract accurate tracking data from videos containing large amounts of background movement, which is often a result of wind when a camera is setup with a hanging-pulley system. An easy solution would be to anchor the camera in such a way so that swaying in windy conditions would be prevented.

We believe that automated video processing provides a meaningful alternative to traditional methods of studying animal behavior, especially that of a nocturnal, secretive species. Past behavioral studies have resorted to methods such as trapping [11], sand transects [14], or test arenas [15]. With proper setup, remotely recorded video, along with automated video processing techniques, can provide information not traditionally available. This information includes data such as speed, distance traveled, frequency of travel, and number of animals in a given space at a given time. This type of information in a natural setting provides crucial information to better understand the evolution and maintenance of behaviors in natural contexts. The use of thermal imaging allows for the collection of these types of data on secretive and nocturnal rodents. Moreover, automated video processing presents a means to efficiently analyze the behaviors present in such videos, although it is equally capable of analyzing behavior in traditional video.

Acknowledgments

The work on this project was supported by National Science Foundation (Grants IOB-0641530, IOB-1132419, DMS-0850465 and DBI-0926288). We would like to thank the Office of Undergraduate Research of UNCG and in particular its directors Mary Crowe and Jan Rychtar. We thank Shan Suthaharan for bringing the group for the initial research project together and David Schuchart for the tracking program that he wrote for the initial project [13]. Thanks also go to Christian Bankester for his work on video analysis [2], Caitlin Bailey, Luis Hernandez, all the students who worked in the field collecting data, and the Hastings Natural History Reserve for all of their support of our field work.

References

1. Andersson, R.: Real-time gray-scale video processing using a moment-generating chip. IEEE J. Robotic Autom. **1**, 79–85 (1985)

2. Bankester, C., Pauli, S., Kalcounis-Rueppell, M.: Automated processing of large amounts of thermal video data from free-living nocturnal rodents. http://www.uncg.edu/mat/faculty/pauli/mouse/reu2011.html (2011)

3. Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, Sebastopol CA, USA (2008)

4. Branson, K., Robie, A.A., Bender, J., Perona, P., Dickinson, M.H.: High-throughput ethomics in large groups of Drosophila. Nat. Methods **6**, 451– 457 (2009). http://dx.doi.org/10.1038/nmeth.1328

5. Branson, K., et al.: CTRAX – the caltech multiple walking fly tracker. http://ctrax.sourceforge.net. Accessed 2012

6. Briggs, J.R., Kalcounis-Rueppell, M.C.: Similar acoustic structure and behavioural context of vocalizations produced by male and female California mice in the wild. Anim. Behav. **82**, 1263–1273 (2011). http://www.sciencedirect.com/science/article/pii/S0003347211003836

7. Chang, F., Chen, C.-J., Lu, C.-J.: A linear-time component-labeling algorithm using contour tracing technique. Comput. Vis. Image Und. **93**(2), 206–220 (2004)

8. EthoVision XT. http://www.noldus.com/animal-behavior-research/products/ethovision-xt. Accessed 2012

9. Li, L., Huang, W., Gu, I.Y.H., Tian, Q.: Foreground object detection from videos containing complex background. ACM MM (2003)

10. Liñán, C.C.: cvBlob – Blob library for OpenCV. http://cvblob.googlecode.com. Accessed 2012

11. Marten, G.G.: Time patterns of peromyscus activity and their correlations with weather. J. Mammal. **54**(1), 169–188 (1973). http://www.jstor.org/discover/10.2307/1378878?uid=2%26uid=4%26sid=21101483719073

12. OpenCV – Open Source Computer Vision. http://opencv.willowgarage.com. Accessed 2012

13. Schuchart, D., Pauli, S., Suthaharan, S., Kalcounis-Rueppell, M.: Measuring behaviors of Peromyscus mice from remotely recorded thermal video using a blob tracking algorithm. http://www.uncg.edu/mat/faculty/pauli/mouse/mathbio2010.html (2011)

14. Vickery, W.L., Bider, J.R.: The influence of weather on rodent activity. J. Mammal. **62**(1), 140–145 (1981). http://www.jstor.org/discover/10.2307/1380484?uid=2%26uid=4%26sid=21101483719073

15. Wolfe, J.L., Tan Summerlin, C.: The influence of lunar light on nocturnal activity of the old-field mouse. Anim. Behav. **37**(Part 3), 410–414 (1989). http://www.sciencedirect.com/science/article/pii/0003347289900882