A Kinect-based system for automatic recording of some pigeon behaviors

Damian M. Lyons · James S. MacDonald ·
Kelly M. Cunningham

Published online: 8 November 2014
© Psychonomic Society, Inc. 2014

Abstract Contact switches and touch screens are the state of the art for recording pigeons' pecking behavior. Recording other behavior, however, requires a different sensor for each behavior, and some behaviors cannot easily be recorded. We present a flexible and inexpensive image-based approach to detecting and counting pigeon behaviors that is based on the Kinect sensor from Microsoft. Although the system is as easy to set up and use as the standard approaches, it is more flexible because it can record behaviors in addition to key pecking. In this article, we show how both the fast, fine motion of key pecking and the gross body activity of feeding can be measured. Five pigeons were trained to peck at a lighted contact switch, a pigeon key, to obtain food reward. The timing of the pecks and the food reward signals were recorded in a log file using standard equipment. The Kinect-based system, called *BehaviorWatch*, also measured the pecking and feeding behavior and generated a different log file. For key pecking, BehaviorWatch had an average sensitivity of 95 % and a precision of 91 %, which were very similar to the pecking measurements from the standard equipment. For detecting feeding activity, BehaviorWatch had a sensitivity of 95 % and a precision of 97 %. These results allow us to demonstrate that an advantage of the Kinect-based approach is that it can also be reliably used to measure activity other than key pecking.

Keywords Kinect · Key pecking · Vision detecting behavior

D. M. Lyons · J. S. MacDonald · K. M. Cunningham
Fordham University, The Bronx, NY, USA

D. M. Lyons (✉)
Department of Computer & Information Science, Fordham
University, Building JMH Room 340, 441 E. Fordham Rd, The
Bronx, NY 10458, USA
e-mail: dlyons@cis.fordham.edu

Recording pigeons' key pecking using a contact switch, a *pigeon key*, is common in studying animal learning and behavior. Such measurements only record whether or not a peck occurred at a specific time and at a general location. More recently, touch sensitive screens, which can provide a precise location of the peck, have become increasingly popular. Additional information could be collected if a video were used, including:

1. the initiation time, speed of motion, head pose at start and end of pecks, pecks on the wall adjacent to key, and pecks initiated but not completed.
2. the global body and head pose: head position and orientation, body position, and foot position.
3. the subject behaviors relative to the stimulus area, turning around, flapping wings, and so forth.

In addition, the noncontact nature of the sensing reduces issues of mechanical wear and allows sensing at a wide variety of locations that do not have specific instrumentation, which provides flexibility for experimental design. Although touch screens may reduce the issue of mechanical wear, there is not enough experience to claim that other issues related to wear would not occur, such as visible damage to the screen. Additional issues with touch screens are the cost of the screen and that no commercially available software is specifically written for controlling learning experiments. Along with developing the programs using, for example, Visual Basic or MATLAB, investigators must integrate interface equipment for delivering reinforcers, providing feedback clicks for pecking etc. Finally, pigeon keys and touch screen can only detect pecks and various investigators have been interested in studying other naturally occurring behaviors, such as in a behavior systems approach (Silva & Timberlake, 2005), or reinforcing topographically different responses like treadle pressing (Wheatley & Engberg, 1978) or head bobbing (Ortega, Stoppa, Güntürkün, & Troje 2013).

Although it might seem that a video would provide data for counting behavior, it is extremely difficult to automate the extraction of this information using a standard video camera. Figure 1 illustrates the problem. When an image of a scene is viewed, through lens optics, on the imaging chip of a video camera, the depth information of the objects in the scene is lost. The intensity of each pixel in the image is created by light arriving along a single ray (e.g., the ray from the top of the head of the pigeon in Fig. 1), and there is no easy way to determine at which distance along that line the object rests. One approach is to use a carefully calibrated camera (Zhang, 2000). Careful measurements are made of the position and pose of the camera (called the *extrinsic parameters*), the lens optics, and the spatial relationship of the lens to the imaging chip (the *intrinsic parameters*). The quality of calibration of the extrinsic and intrinsic parameters heavily influences any measurements made using the camera. Loose animal hair or “pigeon dust” may affect these calibrations. Thus, calibrations may need to be repeated frequently.

Nonetheless, some use has been made of automated video information extraction from a video camera: Pigeon behaviors, such as the “head-bobbing” and “foot-plant” components of courtship, have been monitored from motion capture data and automatic image recognition criteria identified using a conditional restricted Boltzmann machine by Zeiler, Taylor, Troje, and Hinton (2009). Image analysis has also been used to classify avian observations according to species (Song et al., 2008). However, it would be challenging to use image analysis to track the head and beak motions with sufficient accuracy and also be able to detect gross body motions within the full area of an experimental enclosure. Gomez-Martin, Partoune, Stephens, and Louis (2012) described a comprehensive computer vision package, Sensory Orientation Software (SOS), for automated measurements of animal posture and movement. However, they commented on the sensitivity to disturbances of camera pose during measurements. A more general and less expensive solution has recently been developed. The *Kinect*¹ sensor (Freedman, Shpunt, Machline, & Arieli, 2008) is a combination of camera and distance sensor that generates both visual image and a depth image. The Kinect is a structured infrared (IR) distance sensor combined with a camera in such a way that the distance and visual images are registered—that is, that the pixel coordinate system relationship between the two images is known. A visual image can be represented by an intensity map I , where $I(u, v)$ is the image intensity at row u and column v of the image. A depth image, obtained from the IR range sensor, can be represented by a map D , where $D(u, v)$ is the depth (i.e., the distance along the ray in Fig. 1 to the closest object) of the object responsible for the intensity reading $I(u, v)$. The Kinect depth image is 320 pixels wide by 240 pixels high, with a field of view of

Fig. 1 The problem of determining the depth of an object in a video image

57.8°. A point cloud, a set of 3-D points, can be generated from $D(u, v)$ by using the camera focal length to project u and v into x and y scene coordinates, with $z = D(u, v)$. The Kinect is an inexpensive and general-purpose sensor currently available for the consumer video-gaming field (Suma, Lange, Rizzo, Krum, & Bolas, 2011). Because of its popularity in the consumer game market, there is software support for the sensor both from Microsoft² and in the open-source community (OpenNI³ and OpenKinect⁴). The sensor is designed for use in indoor, unstructured settings and can easily be mounted to a large experimental enclosure. That software support includes software for generating and tracking human body features using a *skeletal model*: a 3-D stick-figure model that represents the locations of the subject’s torso and limbs. Such skeletal models have been a topic of research for some time (Moeslund & Granum, 2001). Extracting skeletal models from point clouds generated by distance sensors such as the Kinect has been described by Sharf, Lewiner, Shamir, and Kobbelt (2007) and Suma et al. (2011), among others.

In this article, we present a flexible method for using the *Kinect for Windows* sensor to extract 3-D body information, at video frame rates (i.e., at the same rate that images are taken in the video sequence), of a pigeon viewed within an experimental enclosure. The method is embodied in a program, *BehaviorWatch*. No special care need be taken, other than to approximately center the subject in the field of view and place the closest edge of the animal chamber no closer than 400 mm (300 mm, if the Kinect is equipped with a Nyko wide-angle lens; Draelos, 2012). The maximum distance to the farthest object should be less than 3.5 m. A simple skeletal model is used to represent key body locations. We go on to present an approach to estimate pecking behavior on the basis of the motion of these body measurements. Identifying “treadle pressing” or “head bobbing” could use a similar approach. Finally, we present a comparison of our approach with a standard contact-switch-based approach, with which we show that the approach produces measurements for key pecking that are very similar to the output of a standard contact-switch-based system.

¹ www.microsoft.com/en-us/kinectforwindows/

² Kinect for Windows. <http://kinectforwindows.org>

³ Open source drivers for Kinect. <http://en.wikipedia.org/wiki/OpenNI>

⁴ OpenKinect project. <http://openkinect.org>

Additionally, we show that we can detect feeding behavior also, and that the timing of this detection matches very closely with time when the signal is sent to provide a food reward.

The remainder of the article is structured as follows: The next section presents the method that we have developed for extracting 3-D body information, and the subsequent section describes the approach to estimating pecking behavior from these measurements. The fourth section presents the details and results of the comparison experiment. The final section discusses these results and the future potential for this method of noninvasive experimental measurement.

Extraction of 3-D body information

The Kinect is positioned with respect to the experimental enclosure so that the pigeon is centered in the image (Fig. 2). The stimulus/food presentation area is to the right in this image (the image is left–right reversed). Example visual images from the Kinect are shown in Fig. 3. The respective unprocessed depth images $D(u, v)$ are shown in Fig. 3, panels a, b, c, g, h, and i, as gray-level images, in which the shade of gray is proportional to depth.

Target identification is sometimes difficult in a visual image. For example, Fig. 3j shows a backward-facing pigeon in which the white color from the tail is difficult to separate from the light color of the front of the enclosure. However, Fig. 3g shows the depth image for this frame, in which the foreground pigeon image is clearly separated from its (different depth, different gray level) surroundings. Furthermore, the depth information helps to disambiguate the pose of the pigeon; for example, in Fig. 3a–c, the legs are clearly at different depths.

The first stage in processing is the identification of the region of the depth image (e.g., the pigeon in the center of Fig. 3a) that corresponds to the pigeon body region. Due to the enclosure, pigeon size, and the fixed positioning of the Kinect, this simple foreground identification algorithm works very robustly in practice: The depth of a square region in the center of the image is estimated by averaging the depth values. This is an estimate of the distance from the camera to the nearest contact with the pigeon.

The image is then filtered by removing all pixels that represent a depth of more or less than 10 cm from the averaged depth of the center of the target region. This relatively wide depth window ensures that the full pigeon is seen, even if the region over which the center depth was estimated is only centered roughly on the pigeon. Finally, a target region is then constructed by removing any regions not connected to this central depth region using a common-connected-components (Bradski & Kaehler, 2008) approach. An example target region after this point in the computation is shown in Fig. 4a. This approach has the advantage of requiring no background imagery. However, a more robust target region extraction approach could be obtained at the cost of taking some imagery of the empty enclosure and implementing a background subtraction using the depth information. Such a method is sensitive to subsequent camera displacement during experimental measurements, and hence was not used for this article, but it will be evaluated in future work.

Once the target region has been extracted, the next stage in processing fits an anatomically appropriate skeletal model (Fig. 5b). Although skeletal models have been used extensively for human pose tracking (Straka, Hauswiesner, Ruther, & Bischof, 2011), there is little discussion in the literature of appropriate animal skeletal models. Sundar, Silver, Garvani, and Dickson (2003) used a medial skeleton graph for representing images of animals and objects for the purpose of identifying the shapes in images and retrieving samples from an image database. Gall et al. (2009) used a set of synchronized and calibrated cameras to identify a target in multiple views and to extract a volumetric medial skeleton. They showed examples for dogs as well as humans. Gomez-Martin et al.'s (2012) SOS package uses a single camera, leveraging a medial skeletal model constructed by region thinning as a postprocessing step, to identify body features in a wide range of animals from *Drosophila* larvae to fish. We used a simple skeletal model extracted primarily by region thinning of the target region (Fig. 4c).

Identification of pose

The target region is analyzed first to determine whether the pigeon faces to the left (backward), faces to the right

Fig. 2 (a) Experimental cabinet (at left) and Kinect (at right) with Plexiglas walls; (b) subject in the experimental cabinet. Key switches are in the depressed disks in the right (opaque) wall, the feeding opening is beneath them on the wall, and the Kinect shows in background

Fig. 3 (Panels d, e, f, j, k, and l) Visual images $I(u, v)$ from the Kinect of a subject in the experimental enclosure. (Panels a, b, c, g, h, and i) Associated depth images $D(u, v)$, rendered in gray levels

(forward), is turning (facing to or away from the camera), or is inclined (head down). This analysis is accomplished as follows. A smoothed, three-dimensional central region spine or medial curve $CX(x) = (y, z)$, $x = x_{mn} \dots x_{mx}$ [where $z = D(x, y)$] for the target region is extracted as the center of each column (x) of the foreground region. The width of the target region [difference of the largest row (y_{mx}) and the smallest row (y_{mn}) for each column (x)] and the top edge of the target region (y_{mx})

are extracted as $WX(x)$ and $HX(x)$, $x = x_{mn} \dots x_{mx}$, respectively. Finally, a line is fitted to the $CX(x)$ spine in the x - y (image width and height) plane as a linear approximation to the angle of pitch (up/down) of the pigeon. The angle α between the line and the horizontal (x) axis is measured as the pitch of the animal. A line is also fitted to the spine in the x - z (image width and depth) plane as a linear approximation to the angle of yaw (left/right) of the pigeon, measured as the angle β between the

Fig. 4 (a) Example foreground region; the white area corresponds to the pigeon's body, bounded by a dark line. (b–e) Various poses of the subject. In each case, the sparse, dotted curved line within the foreground region is the extracted medial axis in the horizontal/vertical plane, the straight line

marked “X–X” is the linear approximation to this curve, and at the bottom of the figure, marked “□–□,” is the linear approximation to the orientation of the subject in the horizontal/depth plane

Fig. 5 (a–i) *CX* (solid) and *CY* (dashed) medial curves in the *x–y* plane and raw (unfiltered) feature measurements (marked with rectangles) for a variety of poses. The features marked by “□”s are, from left to right: tail feature point (f_T), leg feature point (f_L), head feature point (f_H), and beak feature point (f_B). Feature measurements are only made in forward,

noninclined poses—for example, in panels (a) through (i). Images (m) through (p) show a backward pose, two side poses, and an inclined, feeding pose. Only the *CX* curve is shown in these panels, since that plays a role in determining pose

x–z line and the *z*-axis. Figure 4b through e show the medial spine and the pitch and yaw lines for several poses.

First, BehaviorWatch determines whether the pigeon is facing forward or backward or is in the process of turning, using the information extracted above. A forward-versus-backward pose is determined by looking at the body width and height on each side of the vertical midline, leveraging the fact that the pigeon breast is typically wider than its tail. The midline is calculated simply as $x_{md} = (x_{mn} + x_{mx})/2$. The left and right heights and widths are calculated as:

$$h_l = \max_{x=x_{mn} \dots x_{md}} HX(x), h_r = \max_{x=x_{md} + 1 \dots x_{mx}} HX(x), \\ w_l = \max_{x=x_{mn} \dots x_{md}} WX(x), w_r = \max_{x=x_{md} + 1 \dots x_{mx}} WX(x).$$

Whether the animal is facing forward is then tested by evaluating the condition

$$\text{forward} = (h_r > h_l) \wedge (w_r > w_l).$$

Turning is determined by looking at the width of the pigeon profile with respect to an empirically determined threshold (smallest) width w_t (determined empirically):

$$\text{turning} = (x_{mx} - x_{mn}) < w_t.$$

If the pigeon is determined to be facing forward, then the pitch is used to determine whether the animal is inclined head

downward or not. When the slope of this line is negative and the pigeon is facing to the right, the pigeon is considered to be inclined:

$$\text{inclined} = \text{forward} \wedge (\alpha < 0).$$

Extraction of skeleton

Because the keys and feeder are to the right in the image, skeleton fitting only happens when the pigeon is facing forward and not inclined. The medial spine *CX* always has a significant downward bend in the region of the legs if the pigeon is viewed from the side, even when the pigeon is pecking the ground. The point is detected by measuring the change of slope along *CX*(*x*) and looking for a minimum point $\partial CX(x)/\partial x = 0$ and $\partial^2 CX(x)/\partial x^2 > 0$. The row and column indices of the minimum point, along with the depth value, are recorded as the *leg feature point* $f_L = (u_L, v_L, z_L)$. This location is shown in Fig. 5a–l as a rectangle just above the legs (only the *x* and *y* indices are plotted; the depth is not shown here).

The tail end and head end of the spine *C* can be distinguished by looking at the average silhouette width on either side of the leg feature point. The center point of the tail-end mass of the silhouette is recorded as the *tail feature point* f_T (shown in Fig. 5 as a rectangle in the tail region).

The head and beak are detected by looking at the medial spine CY , calculated as the centers of the rows (y). A maximum point $\partial CY(y)/\partial y = 0$ and $\partial^2 CY(y)/\partial y^2 > 0$ is identified on the head side of the leg feature point. This extreme point is labeled as the *head feature point* f_H . The point of the silhouette opposite the head feature point is labeled as the *beak feature point* f_B , and the line from head to beak is used to indicate the direction in which the head is oriented. (This assumes that the pigeon is looking toward the front.)

Filtering the skeleton

The values of the feature points in the set of feature points $\{f_L, f_T, f_H, f_B\}$ extracted from each image will be influenced by noise in the image and will tend to vary even with a stationary target. This noise is much less for skeletal-based approaches than for silhouette-based approaches (Gall et al., 2009) but is still an issue. In the still model, the skeleton feature points are extracted at frame rate from the range/image streams and filtered using a set of *Kalman* filters (Bar-Shalom & Li, 1993; Bradski & Kaehler, 2008), one filter per feature point and assuming zero-mean sensor and process Gaussian noise. The state of each body feature is the row, column, and depth locations and velocities (calculated by subtracting locations in consecutive frames), and each point is modeled as stationary with a very small process and measurement noise.

Figure 6 shows the filtered skeleton and feature points from several successive frames at the start of a peck motion by the pigeon toward the stimulus presentation area of the enclosure. Each feature point is indicated by the center of a square marker. Note that the filter parameters do not model this fast forward motion very well and demonstrate considerable lag in following it (see, e.g., the beak feature locations in Fig. 6c and g).

For this reason, we add a parallel set of Kalman filters to model this fast peck motion. The filters differ in their predictive model. The regular filters predict no motion of the feature points, whereas the peck filters predict a small feature velocity toward the right. The peck motion filters also have a larger process noise than the stationary filters, but are otherwise the

same. Thus, the peck filter will follow fast motions to the right better than the regular, still filters, but they will also react more readily to noise.

Detecting peck motions

We combined both the still and peck filter models in an interacting multiple model (IMM) framework (Bar-Shalom & Li, 1993; Blom & Bar-Shalom, 1988). An IMM approach was used by Farmer, Hsu, and Jain (2002) for rapid model switching and behavior detection for an airbag suppression system, to distinguish human-initiated versus crash-driven body motions. Although nonpeck and peck motions are both initiated by the pigeon, the difference in velocity of these kinds of motion is similar to the one successfully addressed by Farmer et al. This approach has the advantage that the model-switching parameters and filter prediction can be also used to classify the start of a pecking motion and the direction of pecking.

The covariance information from each model's Kalman filter for beak and head location is used to calculate the model likelihood (Farmer et al., 2002) and combined with a Markov switching matrix, S . The switching matrix controls the selection of which of the two models suits the observed data better. This matrix $S = \begin{bmatrix} 0.9 & 0.1 & 0.050 & 0.95 \end{bmatrix}$ was chosen to prefer the peck model slightly; this enhances fast peck switching (fast detection of peck), and the greater error for the peck model will quickly cause a transfer back again should the switch not be supported (i.e., should the data not support it). These probabilities are chosen purely on the grounds of fast classification of a peck choice and do not reflect any behavior of the pigeon. It does mean, however, that short-duration "false" peck classifications may be seen and should be ignored.

Figure 7 shows a 5-s example of the fast switching between models when a peck is initiated, with pecks occurring at the vertical dashed lines. The solid line is the probability of the IMM filter peck model. Whenever this probability is greater than the dashed line (the still model), a peck is indicated. The

Fig. 6 Several successive frames, (a–g), in a peck sequence, each annotated with skeleton model (superimposed lines) and features (marked “□”). To save space, the pigeon's tail is omitted from each frame

Fig. 7 Example of the still (dotted line) and peck (solid line) interacting multiple model filter probabilities over 5 s. Dotted vertical lines indicate manually labeled key-peck times. Notice the transitory peck

classifications around 1.6 s and 3.4 s, due to the S matrix's predisposition to select the peck model, and then to reject it when no further evidence is present

figure shows that the switching corresponds well with the pecks. It also shows (at 1.6 s and 3.4 s) the previously mentioned transitory peck classifications. The classifications are transitory because the model is quickly rejected if the data do not continue to support it.

Although this model allows us to detect when a pigeon is pecking, it is not sufficient to detect key pecking. A key peck should only be registered if a peck motion occurs with its endpoint on the key switch. For all of our experiments, the location of the key switch was manually identified a priori. When a peck was detected with its beak feature position within the key zone, a peck was recorded. However, we are developing a user interface that will allow a user to identify pecking target zones on the live image (Fig. 8) in order to make the tool more useful to researchers.

Detecting feeding

Whereas pecking is a fast, fine motion, feeding is a gross motion in which the pigeon inclines its body and inserts its head into the food opening in the experimental cabinet. When the pigeon is facing to the right and the slope of the pitch line (Fig. 4e) is negative, the pigeon is considered to be inclined. It is possible for the pigeon to be inclined and to peck at a

number of locations on the floor and on the wall. A feeding action is classified only when the target is inclined and close enough to the feeding opening to have its head inserted. The range of foot positions to quantify this closeness was identified a priori (Fig. 8).

To determine the accuracy of this automated approach to measuring peck and feeding activity, we trained a set of pigeons to peck an illuminated pigeon key. The experiment and procedure are described in the next section. The purpose of this experiment was to evaluate whether the still and peck algorithms would reliably sense key pecks by pigeons. There are well-established methods for humans to train these behaviors.

Method

Subjects The subjects were five adult pigeons with extensive training in other chambers.

Apparatus The pigeons were trained in one operant chamber with transparent walls, 20-mm-diameter pecking key centered on one wall that could be transilluminated by a green color, a grain feeder with a 2-in. by 2-in. opening 5 in. below the pecking key and a 24-V DC house light located outside the top

Fig. 8 Manual identification of important regions via a graphical user interface. The operator draws a box on the image location (shown on the left depth image for each pair) for (a) the key position and (b) feeding foot position

panel and centered on it. Mounted on a tripod outside the chamber, 40 cm from the closest chamber wall was a Kinect. A “Kinect for Windows” (43° vertical by 57° horizontal field of view, 30 fps) operated in “near” range (40 cm to 3.5 m) recorded visual and IR images. Depth and video sequences were collected and stored using Kinect for Windows SDK 1.5 and Kinect Studio under Windows 7. A program written in C++ using the Microsoft Kinect SDK was then run to extract a sequence of depth image files from the Kinect Studio video file. The sequence could also have been generated directly from the Kinect, without Kinect Studio. However, using Kinect Studio allows the experimenter to review any portion of the video sequence. The algorithm described in the article was implemented in a C++ program, called BehaviorWatch, using OpenCV (Barski & Kaehler, 2008), under Windows, which accepted the sequence of depth image files as input. BehaviorWatch also runs under Ubuntu/Linux, and the experimental results reported here were obtained under Ubuntu on a Dell Latitude D630 laptop (dual core, 1.8 GHz). The program is deterministic, and it produces identical results under Windows and Linux. Recording of responses and delivering of reinforcers were controlled by the MED-PC IV software package and a hardware interface from the company MED Associates and a PC-based computer.

Procedure Because the pigeons were trained to peck in touch screen chambers, they were first trained to eat from the feeder and then to peck by autoshaping (Brown & Jenkins, 1968). Once pecking, they were trained on CRF for one session and then trained on a variable-interval (VI) 20-s schedule for several sessions. Whenever a reinforcer occurred, the house light and key light when out as the feeder was raised and the lamp inside the feeder illuminated it. Then, in one 5-min session, pigeons pecked on a VI 20-s schedule controlled by the MED-PC IV program as the Kinect recorded its behavior. The MED-PC IV log file from this session recorded session onset, the start of the breaking of the pigeon key by a peck, and reinforcer onset at 0.01-s precision. The Kinect imagery was then run through the BehaviorWatch program, which generated a separate log file recording the peck and feeding events and their timestamps. Additionally, a human observer manually recorded the start and end times of feeding behavior in a separate log file.

Results

Each event in the Med-PC IV log file is tagged with its time in milliseconds relative to the start of the session. The first event in the session is the switching of the enclosure light, an easily detectable visual event. The BehaviorWatch log file also tags each event with a time in milliseconds. The time stamp from

the MED-PC log file was manually synchronized with that of the BehaviorWatch log file using the switching on of the overhead light in the experimental enclosure to establish a common start time for both event streams. The timestamps for peck events from the BehaviorWatch log file and from the MED-PC IV logfile were compared. If a MED-PC peck was found within 0.5 s of a BehaviorWatch peck, it was recorded as a true positive (TP); if no such peck was found, then it was recorded as a false positive (FP). If no BehaviorWatch peck was detected within 0.5 s of a MED-PC peck, it was recorded as a false negative (FN). Feeding was compared in a similar way, except that the comparison was with the log file generated by the human observer.

Table 1 shows the duration in seconds for each sequence, the total number of depth frames stored in that duration, the average frame rate (the overall average was 25.3 fps), and the average time taken to analyze an image, as measured by timing the BehaviorWatch software. Although the duration of each sequence was 300 s, the number of frames in each sequence varied due to variations in the frame rate of the Kinect and video frame storage.

The average run time over all sequences was 0.021 s per frame, or a rate of approximately 48 fps, much faster than the approximately 25 fps at which the image data were collected.

Figure 9 shows an example section of a comparison of the log files from MED-PC and from BehaviorWatch. The event codes for peck and feed/reward are plotted against the timestamp for that event in the log file. Notice that where MED-PC reports a reward event, BehaviorWatch reports the observation of the pigeon in inclined pose.

When both log files reported a peck within their time window, this was recorded as a true positive (TP). When BehaviorWatch detected a peck but the MED-PC log file did not, this was recorded as a false positive (FP). When the MED-PC log file detected a peck and BehaviorWatch did not, this was detected as a false negative (FN). Finally, when the MED-PC log file did not detect a peck and neither did BehaviorWatch, this was recorded as a true negative (TN). This approach treated the MED-PC data as the standard; we will return to this assumption in the Discussion section.

Table 1 Session duration, number of frames, average frames per second, and average BehaviorWatch processing time per frame for each of the five subjects

Subject	Duration (s)	Num. Frames	Av. fps	Avg. Time
S1	300	7,651	25.5	0.022
S2	300	7,788	25.96	0.021
S3	300	6,407	21.36	0.020
S4	300	7,637	24.46	0.020
S5	300	8,815	29.38	0.020

Fig. 9 BehaviorWatch and MED-PC log-file event codes (Peck vs. Feed/Reward), plotted against time for Subject 1

Table 2 shows, for each subject, classifications for its key-peck activity and feeding activity (respectively) in terms of TP, FP, and FN counts. The TN numbers are not used, since they are so large (approximately equaling the number of frames) that they swamp the other measurements and hide information.

Sensitivity is defined as the ability to correctly recognize positive results, calculated as $\text{sensitivity} = \text{TP}/(\text{TP} + \text{FN})$. *Specificity* is the ability to correctly recognize negative results, calculated as $\text{specificity} = \text{TN}/(\text{TN} + \text{FP})$. However, the large TN numbers make this measurement always close to 1 and ineffective for evaluating performance. *Precision* measures positive predictive value and is defined as $\text{precision} = \text{TP}/(\text{TP} + \text{FP})$. The F1 score is a single overall score that reflects the accuracy of classification, calculated as $2\text{TP}/(2\text{TP} + \text{FN} + \text{FP})$.

The algorithm showed an average sensitivity for key-peck detection of 95 %, a precision of 91 %, and an F1 score of 92. For feeding detection, it showed an average sensitivity of 95 %, a precision of 97 %, and an F1 score of 95 (see Table 3).

Discussion

This article has described a method to use a Kinect sensor to measure pigeon key pecks and feeding activity. The sensor is relatively inexpensive (selling for US\$250 or less) and it does not require much in the way of calibration or careful setup—just a view of the experimental chamber in which the pigeon is roughly centered. A comparison of the accuracy of this method to a standard key was also presented, showing that the

method had an average sensitivity of 95 % and a precision of 91 % for peck detection when the MED-PC data were regarded as the ground truth, and sensitivity of 95 % and precision of 97 % for feeding detection when observer-scored feeding was the ground truth.

Although using the standard method for collecting key pecks is reasonable (the only other alternative would be to manually validate key pecks, and the key switch is standard piece of experimental technology), it is not unassailable. Our method may *also* detect key pecks that were begun but were not completed, because they were too weak; that stopped short of the key or were too quick to register; or that were to the side of the key (Blough, 1977). Because the key is recessed from the surface of the wall, some pecks can fall short of the key but do enter the recessed area.

The strength of the approach that we present is its ability to be extended to produce additional information. One such piece of information is the “intent to peck.” The algorithm detects a peck as soon as it starts and does not need to wait until the peck hits the stimulus. However, for this work, the recognition of a peck was constrained to be both the recognition of a peck motion and the end of the peck motion lying within the a-priori-defined key area (and see Fig. 8). This information cannot be detected at all using the standard key switch approach. Detecting intent to peck could be useful in training with few or no errors. When a peck to the incorrect stimulus is about to occur, conditions could change to prevent that incorrect response; for example, the stimulus could be moved or replaced with a different stimulus.

Table 2 Frequency of true positives (TP), false positives (FP), and false negatives (FN) by BehaviorWatch versus MED-PC IV

	Peck			Feed		
	TP	FP	FN	TP	FP	FN
Subject						
S1	264	23	54	18	0	1
S2	555	0	2	11	0	1
S3	291	26	9	19	0	1
S4	487	57	26	11	1	1
S5	405	42	0	26	5	0

Table 3 Individual and mean sensitivity, precision, and F1 scores

	Peck			Feed		
	Sensitivity	Precision	F1	Sensitivity	Precision	F1
Subject						
S1	83.0	91.0	87.0	95.0	100.0	97.0
S2	99.0	92.0	96.0	92.0	100.0	96.0
S3	97.0	91.0	94.0	95.0	100.0	97.0
S4	95.0	90.0	92.0	92.0	100.0	96.0
S5	100.0	91.0	91.0	100.0	84.0	91.0
Avg	95.0	91.0	92.0	95.0	97.0	95.0

Although only the peck and feeding data from the algorithm were evaluated in this article, the other body feature points can yield useful information, too:

1. From the skeletal information, we could determine which direction a subject or its head was facing at any time.
2. From gross body motions, we could determine when a subject was frustrated or losing attention (pigeon wing flapping, turning around, etc.).
3. From gross body motions, we could determine when the subject is stepping on a treadle.

The software incarnation of the algorithm used for this article is not a turn-key system, and it requires multiple steps and data copying. In future work, we will to rewrite the software to work directly from the Kinect data and to generate the time-stamped log file. The present software produces the body skeletal data for each frame as well as the start and stop of peck motions. Future work could include extending this vocabulary to include the information about subject direction and state described in the previous paragraph.

The present approach to data acquisition uses the Kinect to collect response data without a transducer for the response. The Kinect may also be suitable for real-time control of experiments. Once the Kinect detects a response, further coding could determine whether to reward it, and as time passed or responses were detected, whether the stimulus conditions should change. This might be especially attractive when presenting visual images on a computer monitor. Currently, no turn-key application is available for setting up a monitor with integrated touch capability or a touch screen overlay, detecting pecks, arranging contingencies, and recording data. Thus, each investigator must troubleshoot setting up a system and continually check that the touch screen has not been damaged by the pigeon pecking. A Kinect may simplify this process. With the Kinect, a simple monitor is all that is required. Because there is no mechanical transducer, the recording of responses should be stable over time. Because the Kinect does not require that a subject physically touch the screen with a minimal force, it can detect pecks that touch the screen very lightly or that do not actually make contact with the screen, but stop just short in the area above the designated peck zone. The Kinect would also be useful for species with soft beaks, since these subjects can damage their beaks by physically pecking touch monitors. At 30 frames per second, a frame occurs every 33.3 ms. Using a comparatively slow processor, the present code takes about 21 ms (or 63 % of the frame interval) to detect whether a peck has occurred, leaving over 12 ms (or 37 % of the frame interval) for additional code to arrange contingencies, deliver rewards, and record data. The major concern with this software is that pecks that occur just after a frame-captured behavior may be missed, since it will be 33 ms until the next frame and peck detection. This probably did not

occur too often, given the high sensitivity scores. In fact, the false negatives may have been these missed pecks. The biggest concern is the delay to reinforcement, which would be an average 16.5 ms longer than otherwise. Many current operant control systems operate with a 10-ms time step, but not too long ago the time steps were 50 ms. Looking back to relay equipment, it is not clear what the delays were then. Thus, although this delay with a slow processor is longer than in most current systems, it is not so long as to pose a problem for most investigators. One way to deal with this would be to use new technology, a device that records at 60 frames per second (see below). With this device, the maximum delay between a peck and its detection would be 17 ms and the average delay, 8.7 ms, which is likely to be within an acceptable range for most investigators.

The Kinect operates at 30 fps, but our sequences averaged just over 25 fps when stored using Windows 7. From a closer inspection of the frame times, it is evident that periodic administrative tasks in Windows 7 cause occasional delays in getting, processing, and storing frames, slowing the actual average frame rate. To the degree possible, stopping those tasks would increase the frame storage rate. At that rate, a peck takes about five or six frames, but the motion in each frame is relatively large. One alternative would be to use a faster sensor. The DepthSense 311 camera from SoftKinect is also a combination IR and visible camera used for depth sensing. The DepthSense camera is slightly more expensive, but it operates at 60 fps. We have carried out initial experiments to integrate this sensor with our software, and future work will include leveraging the faster frame rate for better precision of timing the peck.

References

- Bar-Shalom, Y., & Li, X.-R. (1993). *Estimation and tracking: Principles, techniques and software*. Norwood: Artech House.
- Blom, H., & Bar-Shalom, Y. (1988). The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33, 780–783.
- Blough, D. S. (1977). Photoelectric recording of pigeon-peck responses to computer-driven visual displays. *Behavior Research Methods & Instrumentation*, 9, 259–262.
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV Library*. San Francisco: O'Reilly.
- Brown, P. L., & Jenkins, H. M. (1968). Auto-shaping of the pigeon's key-peck. *Journal of the Experimental Analysis of Behavior*, 11, 1–8. doi:10.1901/jeab.1968.11-1
- Draeos, M. (2012). *The Kinect up close: Modifications for short-range depth imaging*. Raleigh: Unpublished MS Thesis, Dept. of Electrical Engineering, North Carolina State University.
- Farmer, M. E., Hsu, R.-L., & Jain, A. K. (2002). *Interacting Multiple Model (IMM) Kalman filters for robust high speed human motion tracking*. Quebec: Paper presented at the 16th International Conference on Pattern Recognition (ICPR).

- Freedman, B., Shpunt, A., Machline, M., & Arieli, Y. (2008). Depth mapping using projected patterns. Patent No. US 2010/0118123A1 (PrimeSense Patent for Kinect).
- Gall, J., Stoll, C., Aguiar, E., Theobalt, C., Rosenhahn, B., & Seidel, H.-P. (2009). *Motion capture using joint skeleton tracking and surface estimation*. Miami: Paper presented at CVPR 2009.
- Gomez-Martin, A., Partoune, N., Stephens, G. J., & Louis, M. (2012). Automated tracking of animal posture and movement during exploration and sensory orientation behaviors. *PLoS ONE*, 7, e41642. doi: [10.1371/journal.pone.0041642](https://doi.org/10.1371/journal.pone.0041642)
- Moeslund, T., & Granum, E. (2001). A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81, 231–268.
- Ortega, J., Stoppa, L., Güntürkün, K., & Troje, N. F. (2013). Vision during head bobbing: Are pigeons capable of shape discrimination during the thrust phase? *Experimental Brain Research*, 199, 313–321.
- Sharf, A., Lewiner, T., Shamir, A., & Kobbelt, L. (2007). *On-the-fly curve-skeleton computation for 3D shapes*. Prague: Paper presented at EUROGRAPHICS.
- Silva, K. M., & Timberlake, W. (2005). A behavior systems view of the organization of multiple responses during a partially or continuously reinforced interfood clock. *Learning & Behavior*, 33, 99–110.
- Song, D., Qin, N., Xu, Y., Kim, C., Luneau, D., & Goldberg, K. (2008). *Systems and algorithms for an automatonous observatory assisting the search for the Ivory-Billed Woodbecker*. Washington DC: Paper presented at the IEEE International Conference on Automation Science and Engineering (CASE).
- Straka, M., Hauswiesner, S., Ruther, M., & Bischof, H. (2011). *Skeletal graph based human pose estimation in real-time*. Dundee: Paper presented at the 2011 British Machine Vision Conference.
- Suma, E., Lange, B., Rizzo, A., Krum, D., & Bolas, M. (2011). FFAST: The flexible action and articulated skeleton toolkit. In *IEEE Virtual Reality 2011* (pp. 247–248). Piscataway: IEEE Press.
- Sundar, H., Silver, D., Garvani, N., & Dickson, S. (2003). Skeletal-based shape matching and retrieval. In *Proceedings of SMI'03, the Shape Modeling International 2003* (pp. 130–140). Piscataway: IEEE Press.
- Wheatley, K. L., & Engberg, K. A. (1978). Choice performance in several concurrent key-peck treadle-press reinforcement schedules. *Journal of the Experimental Analysis of Behavior*, 28, 181–190.
- Zeiler, M., Taylor, Z., Troje, N., & Hinton, G. (2009). *Modeling pigeon behaviour using a Conditional Restricted Boltzmann Machine*. Brugge: Paper presented at the 17th European Symposium on Neural Networks.
- Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 1330–1334.