Monitoring Creatures Great and Small: Computer Vision Systems for Looking at Grizzly Bears, Fish, and Grasshoppers

Greg Mori Maryam Moslemi Naeini Andrew Rova Payam Sabzmeydani
Jens Wawerla
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada

1 Introduction

For natural scientists, gathering data can be a labour-intensive and expensive process. As an example, traditional techniques for population-scale data collection are marking and recapturing individual animals or performing aerial counts. As an alternative or complement to these methods, the use of camera systems, which collect information largely in the absence of human operators, is increasing in popularity. However, cameras generate large amounts of data, which are typically sorted manually to collect the required data. As computer vision researchers, there is a great opportunity to aid natural scientists by automating parts of the video analysis process.

The Scientific Data Acquisition, Transportation and Storage (SDATS) project has the aim of researching methods of reducing the expense and manual labour involved in gathering scientific data in the field. SDATS projects include developing a system, deployed near the arctic circle, for recording videos and automatically detecting grizzly bears [3], stereo tracking of grasshoppers in cages and attempting to recognize their actions by analyzing their 3D movement [1], and developing a method for automatically determining the species of fish from images collected with an underwater video camera [2]. In this paper we summarize these three previous SDATS projects[1] and provide a discussion.

2 BearCam: Monitoring Grizzly Bears

The BearCam is a camera system deployed in Fall 2005 to monitor the behaviour of grizzly bears at the Ni'iinlii Njik (Fishing Branch) Park. The system aids biologists monitoring grizzly bear behaviour at a new ecotourist destination. The main objective of the biologists' study was to assess whether ecotourists negatively affected grizzly bear feeding behaviour at this salmon spawning stream. This camera system served two purposes: 1) to increase the observation area without additional observers or without reducing the researcher's time at the primary observation area, and 2) to record bear behaviour in an area of minimal ecotourist activity without requiring the researcher's physical presence, which would effectively render this no longer a minimal human activity area.

This park is a remote wilderness area in the Yukon, Canada, just below the arctic circle. The only man-made structures are a handful of wooden huts and electricity is provided by a small gas-powered generator. We developed a camera system for operating in these challenging conditions. We also developed a novel "motion shapelet" algorithm for automatically detecting bears in the video captured by this camera system.

2.1 Detecting Bears

Our system monitors the river site for 4 hours per day. Biologists require enormous amounts of time to manually search these videos for bear activity. For this reason, we developed an algorithm for automatically detecting the presence of bears in the recorded video.

In our bear detection algorithm, a set of informative mid-level features is automatically learned. These mid-level features, called *shapelet features*, are constructed from low-level features. Shapelet features which best discriminate between object and non-object classes are built. The AdaBoost algorithm is used as the core computational routine, using it once to build the shapelet features, and again to build a final classifier from these shapelets.

In our work, we use background differences and gradient responses as our lowest level features. We use the absolute value of gradient responses, computed in four different directions. The absolute value is used because the sign of the gradient is uninformative due to varying background colours. To reduce the influence of small spatial shifts in the detection window, we locally average each of these cues by convolving the responses with a box filter.

We define a *motion shapelet feature* as a weighted combination of low-level features. Each low-level feature consists of a location, a direction or motion, and a strength. Each motion shapelet feature will cover a

[1]We would like to acknowledge our natural scientist collaborators Lawrence Dill, Kristina Rothley, Shelley Marshall, and Greg Dutton.

(a) (b) (c) (d)

Figure 1: Illustration of learned motion shapelets. Column (a) shows low-level gradient features with positive parity, (b) those with negative parity. Column (c) shows motion features with positive parity, (d) those with negative parity.

small sub-window of the detection window, and its low-level features are chosen from that sub-window.

We consider k sub-windows $w_i \in \mathcal{W}$, $i = 1, \ldots, k$ inside our detection window. We build a separate mid-level motion shapelet feature for each sub-window w_i. To do this, we collect all the low-level features $f_t(x)$ that are inside that sub-window and consider decision stumps based on them as potential weak classifiers of an AdaBoost run: $h_t(x) = p_t f_t(x) < p_t \theta_t$ where $\theta_t \in (-\infty, \infty)$ is the classification threshold of the weak classifier and $p_t = \pm 1$ is a parity for the inequality sign.

After creating these motion shapelet features, a second run of AdaBoost over all of them is used to build a final classifier.

We visualize the results of the motion shapelet learning algorithm in Figure 1, which shows the sum of all the low-level features selected inside all the motion shapelet features over the entire detection window. The selected low-level features are separated in two groups according to their classification parity. This parity shows whether the selected feature is part of a positive (bear) or negative (non-bear) discriminating motion shapelet feature.

2.2 Results

We performed experiments comparing the use of different parameter settings and a comparison of shape and motion features in isolation and together. Space does not permit a full description of these experiments. We performed tests on whole video frames, a realistic test of the final system. We ran our detector in a "window-scanning" fashion, sliding it across the input frame and running our classifier at each location.

For training, we marked 451 windows containing bears from 6 different clips of video as our positive set, and extracted non-bear windows from the same videos. A test set from a separate set of 4 video clips was created. For the purposes of this experiment, an image is considered a true positive if it contains at least one bear, and a false positive if it contains no bears. 405 frames containing at least one bear were labeled as our positive set, while 16000 frames not containing bears comprise the negative set.

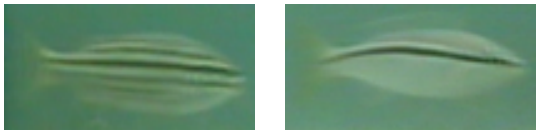

Figure 2: The two types of fish to be classified: Striped Trumpeter (l), Western Butterfish (r)

We marked a region of interest (ROI) within the video, in order to ignore areas where bears cannot appear. We ran our detector over each image, and kept the single highest-valued response over all detection windows within the ROI as the response for each image.

This method proves effective at automatically detecting the occurrence of bears For example, we can detect 76% of the frames containing bears at 0.001 false positive images per image examined, or 88% at 0.01. Such a detection rate would be useful for building an interactive system, where a user could be guided to frames in the video where bears are likely to be present.

3 Fish Species Recognition

Quantifying the number of fish in a local body of water is of interest for applications such as guiding fisheries management, evaluating the ecological impact of dams, and managing commercial fish farms. Going beyond an aggregate count of aquatic animals, information about the distribution of specific species of fish can assist biologists studying issues such as food availability and predator-prey relationships. Applications like these motivate the development of methods for collecting biological data underwater.

In our work, we only address the species recognition problem, and assume that a pre-processing step has obtained bounding boxes containing individual fish. Fig. 2 shows examples of the two species of fish we attempt to discriminate between.

We approach this task as a deformable template matching problem followed by the application of a classifier based on texture features. Aligning the images before classifying by appearance provides a demonstrable increase in performance.

3.1 Deformable Template Matching

For each of the two classes, we repeat the following steps. First, a representative template image is chosen. A set of interest points, sampled from Canny edge detection results, is chosen on the template. These interest points are connected into a minimum spanning tree.

Correspondence between a new test image and a template is performed by finding a configuration which

minimizes a cost function defined using these interest points and tree. If a tree has n vertices $\{v_1, \ldots, v_n\}$ and an edge $(v_i, v_j) \in E$ for each pair of connected vertices, then a configuration $L = (l_1, \ldots, l_n)$ gives an instance of an object, where each l_i specifies the location of part v_i. Then, the best match of a template to a test image is

$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right) \quad (1)$$

where $m_i(l_i)$ is a matching cost between features in the model and query image at location l_i, and $d_{ij}(l_i, l_j)$ is the amount that model edge (v_i, v_j) is changed when vertex v_i is located at l_i and v_j is placed at l_j. In our method, $m_i(l_i)$ is a matching cost based on *shape contexts* and $d_{ij}(l_i, l_j) = \|(l_j - l_i) - (l_j^m - l_i^m)\|_2$, with l_k^m denoting the location of vertex v_k in the model tree.

3.2 Results

Once the query images have been transformed into estimated alignment with the template they are processed to extract texture properties. First, each image is convolved with a 3-pixel-tall vertical central difference kernel. The motivation for vertical derivative filtering is that after successful warping, the vertical direction captures the most image information. Next, the filter response is half-wave rectified to avoid cancellation during subsequent spatial aggregation. Each half-wave component of the filter response is summed into 7-pixel square sections. Finally, all of the combined filter responses are concatenated into a feature vector as input to an SVM.

For each species of fish being classified 160 images were manually cropped from frames of underwater video. In these 320 images, the fish appear at different angular poses although all of their heads face the right. All images were converted to grayscale, deinterlaced and resized to 50 $\times$ 100 pixels, empirically chosen based on the size of the majority of fish in the images. Half of these data were used for training, and half for testing.

Experiments were performed with linear and polynomial SVM kernels. Linear kernels obtained 84% (no warp) and 90% (warping) classification, polynomial 81% (no warp) and 86% (warping). With both, warping the fish into alignment with a template improved classification performance.

4 Clustering Grasshopper Actions

Our biologist partners are interested in analyzing the behavior of grasshoppers with a particular focus on predator-prey relationships. An adult grasshopper is placed in an enclosure with one of its natural predators, a juvenile spider. Though the spider is immature, and hence harmless, the grasshopper will still exhibit behaviour similar to that it would in the presence of a real threat. The actions of the grasshopper in response to this threat can be studied under a range of environment variables – temperature, illumination level, presence of food source, etc.

Figure 3: Experimental setup for grasshopper monitoring. A "time-lapse" visualization of a portion of video is used – only one grasshopper is present in the enclosure. A stereo pair of cameras is used, only one image is shown here.

As is commonly the case, many hours of data need to be collected in such a study, and an automated system for collecting these data would be very useful. We have developed a novel application of computer vision methods to this problem in order to track grasshoppers and form clusters of similar "actions."

Tracking this insect is difficult due to its small size and its colour variation (dorsal / ventral). In addition, the insect makes occasional very swift jumps which are difficult to track. In consultation with our biologist partners, three walls of the enclosure were painted pink, with a glass face left for viewing. This overcame these difficulties, and a simple image differencing scheme is used to detect and track the grasshopper. Figure 3 depicts this experimental setup.

4.1 Approach

Our biologist partners are particularly interested in movement rates and location preferences of the grasshopper under the different experimental conditions. Since we require 3D information for these data, we use a calibrated stereo pair of cameras to track the location of the grasshopper in 3D.

Summary statistics on location or movement rates can be computed given these 3D tracks. However, these do not tell the whole story – for example a few sudden jumps over the course of an experiment would not be obviously different from a slow meandering around the enclosure. The biologists also wish to obtain a finer-grained analysis of the grasshoppers' actions.

To this end, we attempted to automatically group

segments of these tracks into similar types of motions, or perform clustering into primitive action types. These clusters of similar motion could then be presented to a user who could use them to gain insight into the behaviour of the grasshopper over the experiment.

We construct a motion feature from windows of contiguous frames within the 3D track of the grasshopper. A Gaussian filter is used to remove the noise in the tracker output. Then for each non-overlapping window of size W of 3D position of the object we compute the difference between x_t (location of grasshopper in 3D at time t) and $x_{t+\delta_t}$ for each of the frames in this window. So our feature vector V_t for window of size W of 3D coordinates sequence in time will be $V_t = \{|x_k - x_{k+\delta_t}| : k = t, t+1, \ldots, t+W\}$.

We then perform spectral clustering on these W dimensional feature vectors. For our application, there will be thousands of windows of frames, so constructing, storing, and computing the eigenvectors of the matrix needed for spectral clustering will be intractable. To overcome this limitation, we apply the Nystrom extension which provides a method for extrapolating eigenvectors computed on a portion to the entire matrix.

4.2 Results

We tested the presented algorithm on 16500 frames of a video of one grasshopper. Stereo tracking, motion feature computation, and spectral clustering of windows of frames are all performed. We performed experiments with different numbers of clusters. In order to verify the accuracy of our clustering, we manually marked ground truth labelling of these frames into 3 classes of distinct actions – standing still, walking and jumping.

The *cluster purity*, or percentage of frames which belong to the dominant type in each cluster, is relatively high – near 90% for all numbers of clusters in the range 3-8. The "jumping" action proved the most difficult, due to its rarity. However, with a sufficient number of clusters (more than 5) approximately 80% of "jumping" frames are placed in a "jumping" cluster.

We have also experimented with a preliminary method for visualizing the entire video based on these clustering results, grouping together snippets of video which are placed in the same cluster.

5 Discussion

In this paper we have described three applications of computer vision to biological data collection tasks in different domains. Our experience on these projects has convinced us that a symbiotic relationship between computer vision researchers and natural scientist partners exists.

A vast number of research projects in the natural sciences require monotonous manual labour for data collection. Gathering large amounts of data can be expensive and error-prone due to its tedium. Providing these scientists with data that has been automatically processed by computer vision algorithms would be of great benefit. It would be possible to conduct larger-scale studies and free up researchers to focus on other tasks.

For computer vision researchers, the natural science domain provides challenging real-world datasets to test the capabilities of vision algorithms. These datasets are often collected outside the laboratory and are subject to real-world lighting and other imaging difficulties. Further, the ground-truth definitions, and the decisions on which objects to track or recognize are in the hands of natural scientists. This separation of performance measurement from algorithm design is advantageous.

We also believe that progress in the "looking at animals" domain will advance the state of the art in the "looking at people" domain. Many of the same problems exist in the two domains. For example, in the fish species counting problem we made attempts to automate the fish detection problem. However, these attempts were confounded by shifting lighting conditions underwater due to waves, large amounts of sediment and debris floating in the water, low colour contrast between the fish species and the water/sand, large variations in the image sizes of fish, and focus problems with the camera. All of these problems are present in some form in the surveillance of people.

The bear detection algorithm similarly suffered from these difficulties. The false positives for bear detection are often found in textured areas, such as those on the banks of the river, or regions of the water, which contain large amounts of gradient information. In addition, areas of motion, such as birds or ripples in the water, also lead to false positive detections.

In some cases, such as the grasshopper action problem, it is possible to instrument the environment so as not to affect the behaviour of the subject. Even in this setting though, problems remain. Our attempt at automatically clustering the motions of the grasshopper were an initial foray into attempting to obtain some primitive "action" vocabulary. This is also an open problem in the study of human movement.

References

[1] M. M. Naeini, G. Dutton, K. Rothley, and G. Mori. Action recognition of insects using spectral clustering. In *IAPR Conf. on Machine Vision Applications*, 2007.

[2] A. Rova, G. Mori, and L. M. Dill. One fish, two fish, butterfish, trumpeter: Recognizing fish in underwater video. In *IAPR Conf. on Machine Vision Applications*, 2007.

[3] J. Wawerla, S. Marshall, G. Mori, K. Rothley, and P. Sabzmeydani. Bearcam: Automated wildlife monitoring at the arctic circle. *J. of Machine Vision Applications*, 2008. (to appear).