Biometric Animal Databases from Field Photographs: Identification of Individual Zebra in the Wild

Mayank Lahiri
Dept. of Computer Science
University of Illinois at Chicago
mlahiri@gmail.com

Chayant Tantipathananandh
Dept. of Computer Science
University of Illinois at Chicago
ctanti2@uic.edu

Rosemary Warungu
The Ol'Pejeta Conservancy
Laikipia, Kenya
rosengima@yahoo.com

Daniel I. Rubenstein
Dept. of Ecology and Evolutionary Biology
Princeton University
dir@princeton.edu

Tanya Y. Berger-Wolf
Dept. of Computer Science
University of Illinois at Chicago
tanyabw@uic.edu

ABSTRACT

We describe an algorithmic and experimental approach to a fundamental problem in field ecology: computer-assisted *individual* animal identification. We use a database of noisy photographs taken in the wild to build a biometric database of individual animals differentiated by their coat markings. A new image of an unknown animal can then be queried by its coat markings against the database to determine if the animal has been observed and identified before. Our algorithm, called StripeCodes, efficiently extracts simple image features and uses a dynamic programming algorithm to compare images. We test its accuracy against two different classes of methods: Eigenface, which is based on algebraic techniques, and matching multi-scale histograms of differential image features, an approach from signal processing. StripeCodes performs better than all competing methods for our dataset, and scales well with database size.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Management Applications—*Image databases*; J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology*

General Terms

Ecology, biometrics, image databases, edit distance

1. INTRODUCTION

In wild animal populations, collecting behavioral data about a species often entails identifying individual animals between sightings taken at different places and times. This is a primitive operation in ecological analysis that underlies broader aspects of animal behavior research [24, 26]. Electronic tracking devices embedded in animals are one approach to identifying individual animals, but can be prohibitively expensive and difficult to design for field conditions, and involve considerable cost and risk for larger animals [10, 33]. Researchers are therefore left with no alternative other than to manually record data about individual animals in the field using methods such as manual visual identification from photographs or video [4, 24, 32], genetic markers in excrement [27], or *capture-recapture* techniques [20]. Advances in hardware and the corresponding drop in prices of digital cameras have increased the availability of digital photographs of wild animal sightings at high resolutions and qualities, making fully-automatic or computer-assisted animal identification an attractive approach.

We describe a technique for identifying individual animals from their coat markings in typically noisy field pictures (non-cooperative subjects, coat deformations, occlusion, and variations in exposure, scale, and perspective). Working with field ecologists, we collected a dataset under these conditions for automatic individual animal identification in two species of zebra in Kenya, to augment the efforts of professionally trained field assistants. The techniques we develop are applicable to animals with prominent morphological characteristics like stripes or large patches[1], and are intended to be part of a cost-effective, computer-assisted individual animal identification system. Our algorithm capitalizes on the high resolution of modern field pictures (typically 8 or more megapixels with commodity hardware) to offer excellent retrieval accuracy, transparency in terms of visual feedback on image matches (*i.e.,* the algorithm is not a black box), and extremely simple implementation.

We approach the problem by first extracting a set of discriminative mathematical (as opposed to biological) features from an image of an animal, tolerant to noise from variances in scale and exposure, occlusion, partial deformations, and mild shear. These features allow us to efficiently and robustly compare images in a database by their appearance. We then develop a distance measure between a pair of fea-

[1]Although we only present results for zebra, preliminary field tests suggest that the method can successfully be applied to giraffe.

ICMR '11, April 17-20, Trento, Italy

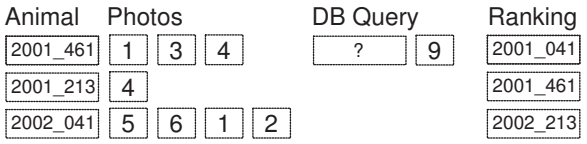


Figure 1: Structure of the image database: contents, query, and output.

ture sets taken from two images, and an efficient algorithm for computing it, that allows us to judge how different the coat markings depicted in two pictures are. A lower distance between images of two animals signifies a higher chance that the two animals are the same. This measure is used to determine whether an animal just photographed in the wild exists in a database of prior sightings.

The lack of availability of an open-access dataset, or source code, for individual animal identification seems to have been an impediment to progress in this area, and is likely the reason why prior studies have been unable to test competing classes of methods [5, 11, 22]. We attempt to rectify this by not only releasing our dataset and code, but by performing one of the first comparative studies of three different classes of techniques (PCA, edit distance, and differential image features). We have published our dataset on two species of zebras publicly, with many annotations (GPS coordinates, date and time, camera model, focal length of image, exposure, *etc.*).[2]

2. PROBLEM DEFINITION

Let D be a database of animal coat marking images $q_1, ..., q_N$. Each image is associated with an *animal identifier* a_i. Individual animal identification requires a ranking algorithm $\mathcal{A}$ that operates directly on a query image $Q \notin D$ to produce a ranking.

$$\mathcal{A} : D \times Q \rightarrow \mathcal{R}$$

where $\mathcal{R}$ is a permutation (ranking) of the original set of images $\mathcal{D}$ based on similarity to the query image $\mathcal{Q}$. The similarity should return an approximate distance between the coat markings of animals.

(Optimal ranking) For a query image q_i of a known animal a_i, an *optimal* ranking algorithm will always return q_j at the highest rank where $a_j = a_i$.

$$\mathcal{A}_{OPT}(D \times q_i) \rightarrow \langle q_j, ... \rangle \quad \text{if } a_i = a_j$$

In general, for a query image of animal a_i, we want to minimize the rank of a database image with the same animal identifier. The ranking algorithm therefore needs to model the similarity (or distance) between animal coat markings, and not necessarily images, which can vary substantially for the same animal.

A line of research in population biology examines the relationship between morphological characteristics of an animal and genetic traits or physiological processes [6, 9]. A natural question would be to ask whether the distance function we learn above can be used as an approximation of 'morphological distance' between individual animals. It should be noted that if the algorithm is truly a distance function on the bodily markings of individual animals (assuming one exists), then it is optimal by the definition above. However, if it is optimal by the definition above, then it is not necessarily a true distance function on the bodily markings of individual animals. This can be easily proved by noting that randomly permuting the lower ranks of the output of an optimal ranking function does not affect its optimality, even when it would destroy the true ordering by morphological distance. Since we cannot hope for an optimal algorithm, we instead aim to minimize the rank of the correct animal in the ranking. However, given the impossibility of any exact methods for characterizing the phenotypic markings of animals, a suboptimal but well-performing ranking function can serve as a useful, if imperfect, biological index of morphological similarity between animals.

[2]All code and data, as well as a GUI frontend, may be downloaded from `http://code.google.com/p/stripespotter/`

3. ANIMAL RECOGNITION

A number of ecological datasets on animal behavior in the wild are currently compiled using variations of the following basic workflow.

1. Scouts or camera traps photograph animals at various locations and times.
2. A trained field assistant codes each image using ad-hoc codes to describe various physical features.
3. A database of reference image codes is searched using the textual code for the observed animal.
4. If the database contains a match for the code, the field assistant verifies that the query and reference animals match by visually comparing images, after which the observation is recorded.
5. If the database does not contain a match, then it is presumed that the animal is new, and is added to the database with its code. If there was a human error in coding the photograph, then this is a *false positive* for a new animal (*i.e.,* rejecting a previously seen animal from the database).

Errors in the identification process can have serious consequences. False positives can cause over-counting of an endangered species. They can also cause missing animal association data, which as a form of sampling error can cause non-trivial biases in higher-order analysis, such as network analysis [8]. False negatives (*i.e.,* not identifying a new animal as one) also introduce errors in behavioral analysis, and can be triggered by changes in an animal's appearance due to growth, aging, pregnancy, and scars from fighting.

We simplify the workflow described above by attempting to eliminate the ad-hoc manual coding process, which is time-consuming and the primary source of error. In its place, we describe a ranking algorithm for an image database that operates directly on a query image. An effective ranking algorithm will rank the correct reference animal highly, which would reduce the false positive rate. On the other hand, drastic changes in an animal's appearance can make it impossible for even humans to tell if the animal has been observed before, so we focus on minimizing the rank of the correct animal if it exists in the database, as opposed to determining whether an animal exists in the database or not.

3.1 Image acquisition

Our procedure starts with an image of the *region of interest* (ROI) of an animal, manually cropped "as consistently

as possible," but subject to some variation.[3] Unlike the approach of Foster *et al.* [11] and Burghardt and Campbell [5], we do not place any constraints on where the ROI should be located. We do, however, assume that the animal being identified has coat markings of a small number of distinctive colors, and relatively large and prominent morphological features (*e.g.,* stripes in zebras and tigers, or large patches in a giraffe). Specifically, regions of each color should be accurately separable by a color segmentation algorithm (see Cheng *et al.* [7] for a review). We also assume that image rotation is not a significant factor, or that the image has been rotation-aligned in a consistent way. Coincidentally, a recent paper on unsupervised image region segmentation used zebra images as a test dataset [18]. We leave the incorporation of such methods to future research, since they introduce an additional source of error.

Assuming that the coat marking has C principal colors, the image is first filtered into k horizontal bands. Within each band, we retain a single summary row of pixels that is their average value in the column, yielding an image of $1/k^{th}$ the height of the original. This is done to accommodate small vertical shifts in the cropping process (horizontal shifts are accounted for by the matching algorithm). Each row of the resultant image is thresholded so that the pixel at each column contains a principal color of the animal's coat at that point. Since we deal with zebras, median thresholding segments the image into two colors. Figure 2 shows the feature extraction process visually.

3.2 StripeCodes: animal coat features

Starting with a C-color thresholded image, as shown in Figure 2(c), we read off k rows of C-ary values ($C = 2$ for zebras and giraffe) in run-length encoding [21]. This yields a sequence of size n of $(color, length)$ values, denoted:

$$X = \langle (c_1, l_1), ..., (c_n, l_n) \rangle, \quad c_i \in \{1...C\}, l_i > 0$$

To impose scale invariance, we perform the following transformation on X:

$$l_i = \frac{l_i}{l_{i-1}}, \quad i > 1$$

We express each l_i as a ratio of its length to that of the previous color block, and drop the first color block from the sequence. We call X a *StripeString*. As a sequence of ratios of lengths, by definition, it is preserved under affine transformations such as shear, or in practice, tolerant to small changes in perspective. Furthermore, it is also tolerant to occlusion; for example, if the left part of the image is clipped, the remainder of the resultant sequence will still match the suffix of the original sequence, with at most the first two ratios being altered.

An image of an animal is therefore represented by an ordered set (*e.g.*, top of animal to bottom) of StripeStrings, which we call a *StripeCode*. The space complexity of this representation for an image is proportional to k multiplied by the average number of color blocks in a row. Simply put, in zebras, it is proportional to the number of stripes in the region of interest.[4]

[3]In our study, instructions given to the user performing the cropping, *i.e.*, a field assistant, were to crop the image to a consistent area of the zebra's anatomy across pictures.

[4]Note that stripe density varies dramatically between two species of zebra: *Plains* and *Grevy's*.

3.3 Distance function

Assuming that exactly the same portion of the animal's body was consistently cropped as the region of interest across photographs, and that the animal was photographed at the exact position in its walking gait as a stored picture, we could directly compare two sets of StripeStrings to determine how similar they are. Since these are unreasonable assumptions, we develop an approximate distance function on two StripeStrings using a variation of the *edit distance* algorithm for two strings. This allows us to partially match two StripeStrings while taking into account factors such as translation (caused by inconsistent cropping of the ROI) and localized deformations (caused by physical deformations of the animal's coat during movement). In principle, our approach is inspired by Blum's original shape representation scheme [3], but the specific underlying methods we use to implement it are different. The distance between two StripeCodes is defined as the average of the distance between corresponding StripeStrings.[5]

We use the notation of Marzal and Vidal [17]. Given two StripeStrings $X = \langle x_1, ..., x_M \rangle$ and $Y = \langle y_1, ..., y_N \rangle$, where $x_i = (c_i, l_i)$ and $y_j = (c'_j, l'_j)$ are *color blocks*, we seek a transformation of X to Y that minimizes a cost function. The transformation is defined as a sequence S of *edit operations*, where each edit operation $S_i = (a, b)$ operates on a pair of color blocks or the null string λ.

$$S = \langle S_1 = (a_1, b_1), ..., S_P \rangle \qquad (1)$$
$$a_i \in \{X \cup \lambda\}, b_j \in \{Y \cup \lambda\}, S_i \neq (\lambda, \lambda)$$

The cost of a single edit operation $S_i = (a, b)$ is defined as $\gamma(a \to b)$, subject to the conditions above.

$$\gamma(a \to b) = \begin{cases} \mathcal{D} & \text{if } a = \lambda \text{ or } b = \lambda \\ \infty & \text{if } c_a \neq c_b \\ 1 - \frac{\min(l_a, l_b)}{\max(l_a, l_b)} & \text{if } c_a = c_b \end{cases} \qquad (2)$$

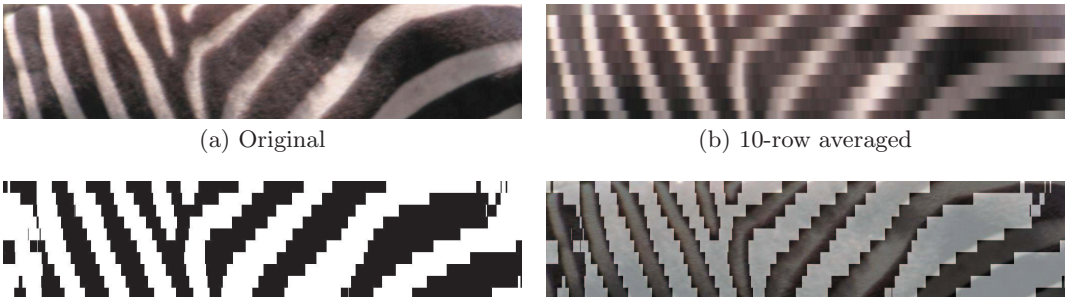
$\mathcal{D}$ is a constant insertion/deletion cost, and (c_a, l_a) and (c_b, l_b) are the color blocks corresponding to a and b if neither is the empty string. Note that $0 < \gamma(a \to b) \leq 1$ if $a, b \neq \lambda$ and $c_a = c_b$. Also note that the function is symmetric, *i.e.,* $\gamma(a \to b) = \gamma(b \to a)$, and that $\gamma(a \to a) = 0$ if $a, b \neq \lambda$.

The cost function above can be trivially extended to the entire edit sequence of length P, defined in Equation 1. This yields the total cost of aligning two StripeStrings using a given edit sequence as the sum of costs of each element in the edit sequence S. The *edit distance* between two StripeStrings is then the minimum cost achievable over all possible editing paths.

$$d(X, Y) = \min_{S = \langle S_1, ..., S_P \rangle} \sum_{i=1}^{P} \gamma(S_i) \qquad (3)$$

Marzal and Vidal [17] have noted that an edit distance computed using Equation 3 is not always appropriate for matching strings of different lengths because the final edit cost does not take the lengths of the strings into account; longer strings will generally have a higher cost associated with their matching. The authors propose the use of a *normalized edit distance* measure d_N, where the edit cost of two strings is divided by the length of the edit path (which ranges between

[5]Using the average was found to be preferable over other summarization functions, like the minimum.



(a) Original (b) 10-row averaged

(c) Median threshold by row (d) Difference between (a) and (c)

Figure 2: Feature extraction process. Typical image processing errors can be seen in the top right and leftmost portions of the difference image.

M and $M + N$ for $M \leq N$).

$$d_N(X, Y) = \min_{S=\langle S_1,\ldots,S_P\rangle} \frac{\sum_{i=1}^{P} \gamma(S_i)}{P} \tag{4}$$

However, minimizing the distance in Equation 4 is not necessarily the same as minimizing the un-normalized cost in Equation 3 and then dividing by the length of the resultant editing path, a procedure known as *post-normalization* [17]. Instead, we used an iterative, unbiased fractional programming algorithm described by Vidal *et al.* [29] to directly compute the normalized edit distance. In our experiments, however, we found that editing paths and costs rarely varied significantly between post-normalization and the unbiased algorithm. Since the unbiased algorithm generally has a larger constant factor, we use post-normalization as an approximation in practice.

A fractional programming algorithm to compute Equation 4 uses an inner loop that computes Equation 3 along the way using the standard dynamic programming method for edit distance [31]. We create a dynamic programming matrix $Z_{(M+1)\times(N+1)}$, which is filled in as follows.

$$\begin{aligned} Z[1,n] &= (n-1)\mathcal{D}, \quad \forall n \\ Z[m,1] &= (m-1)\mathcal{D}, \quad \forall m \\ Z[m,n] &= \min \begin{cases} Z[m-1,n] & +\gamma(\lambda, (c'_n, l'_n)), \\ Z[m,n-1] & +\gamma((c_m, l_m), \lambda), \\ Z[m-1,n-1] & +\gamma((c_m, l_m), (c'_n, l'_n)) \end{cases} \end{aligned} \tag{5}$$

The matrix is filled row by row and the cost of the optimal alignment is contained in cell $Z[M + 1, N + 1]$. By storing which case of Equation 5 was selected at each cell, backtracking from cell $Z[M, N]$ returns the optimal edit path. The cost function γ from Definition 2 defines a substitution matrix for an edit distance computation. More formally, for any given pair of StripeStrings, the combination of γ and $\mathcal{D}$ define the StripeCode substitution matrix if the (otherwise numeric) l_i values are treated as symbols.

Figure 3 shows the edit path taken to align two StripeStrings in our dataset, with $\mathcal{D} = 0.6$. Horizontal and vertical bars in the edit path represent insertions and deletions that incur cost $\mathcal{D}$, and diagonal bars represent matches. Notice that different colors are never matched because of the high penalty associated with such an action.

Finally, the complexity of the distance algorithm has some unique features in the animal identification domain. If M and N are the number of color blocks in a pair of StripeStrings, the post-normalized edit distance can be computed using dynamic programming in $O(M \cdot N)$ time and $O(M \cdot N)$ space. The fractional programming formulation for computing the normalized edit distance is an iterative procedure that runs the unnormalized edit distance algorithm for (usually) a very small number of iterations. However, M and N are both bounded by the number of distinctive color patches (stripes, in our case) in the ROI, and can be effectively considered constant for a given animal species. The efficiency of the algorithm is therefore heavily reliant on a particular implementation.

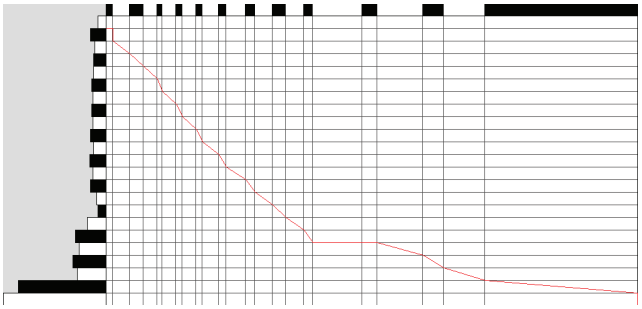
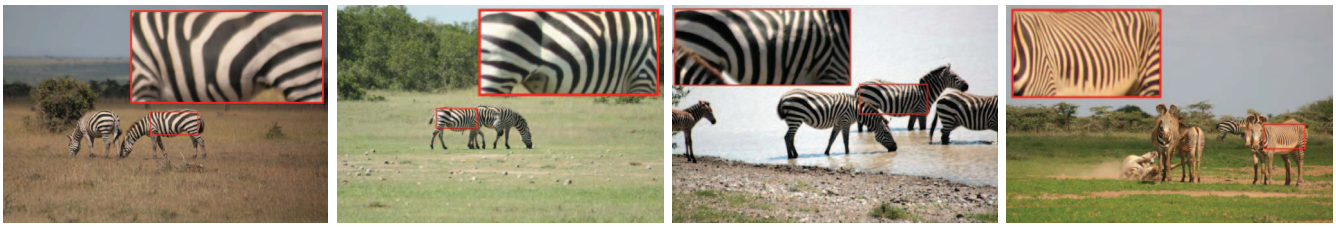


Figure 3: An example of aligning two StripeStrings using dynamic programming.

4. RELATED WORK

The need for individual animal identification in ecology and field biology has been long recognized, as has the tedious and error-prone nature of the task when performed manually [14]. There are three broad categories of methods: those designed for humans to follow manually [24, 32], semi-automatic methods developed with a specific species in mind [1, 12, 15], and semi-automatic methods that can be applied to a class of species that share similar morphological characteristics [5, 22]. Our approach falls in the last category.

Intuition would suggest that obvious candidates for animal recognition algorithms are human biometric identification techniques like fingerprint and face recognition algorithms. A major difference from human biometric identification is that wild animals are almost never cooperative, resulting in large pose and occlusion variances in the images



(a) Regular photo (b) Overlapping animals (c) Occlusion, shadows (d) Skew, species variation

Figure 4: Typical images from our dataset, with the cropped ROI (inset). All four images are of different animals. Image 4(d) is *E. grevyi*, an endangered species of zebra with higher stripe density.

of even a single animal. Furthermore, the coat patterns of an animal can be altered drastically by attacks from other animals, disease, or pregnancy. For zebras, Foster *et al.* [11] describe why fingerprint recognition algorithms are inappropriate. Human face recognition algorithms, however, are more generic, and we test one such method here. *Eigenface* was developed by Turk and Pentland [28], and uses principal component analysis and covariance between image matrices to recognize faces.

Ravela and Gamble [22] use multi-scale histograms of differential image features to identify individual salamanders. In an initial preprocessing stage, the image is treated as a three-dimensional intensity map and filtered with invariant combinations of specific Gaussian derivative operators [23], at multiple scales. The resultant image features are then binned into a histogram. Comparing two images in a database scan is reduced to computing an inner product between two such histogram vectors, which is a very efficient process. However, the use of multi-scale histograms is essentially a black box, with little meaningful feedback available to the user. This could be a disadvantage when developing a computer-assisted identification system, since the user does not get any feedback beyond a similarity score.

Other more sophisticated approaches include that of Burghardt and Campbell [5], who use successive video frames to map a two-dimensional image onto a three-dimensional model of the animal's body. Our algorithm is much simpler, and does not require either video or a three-dimensional model of the animal's body. Foster *et al.* [11] present partial details of a zebra identification system, but require the user to manually select six pre-defined points on each image, corresponding to specific parts of the zebra's body. In comparison, our method simply requires the user to draw a box around any (presumably distinctive) part of the animal's body, and is not specifically tailored to specific features of zebra stripes. Other approaches exploit features of specific species of animals. Curve-extraction and matching, for example, has been used to identify individual dolphins [12] and elephants [1].

In principle, a multitude of approaches in pattern recognition could be applied to the individual animal recognition problem. Shape matching is an obvious candidate for animals with large, prominent coat markings [19]. While these might ultimately be the best performing approaches, we demonstrate in this paper that a simple approach based on dynamic programming is both transparent (*i.e.,* yields visual feedback for a human operator), and performs on par with the multi-scale histograms described by Ravela and Gamble [22]. Other sophisticated object recognition approaches include shape contexts [2] and shock graphs [25].

5. DATASET

We collected our dataset over a period of seven days at the Ol'Pejeta Conservancy in Laikipia, Kenya, using typical field procedures in ecological data collection. The broader research goal was to collect accurate individual identifications, and therefore accurate association data, for network analysis of two different zebra populations in the area [26]. We used cheap, off-the-shelf digital SLR cameras and 300mm zoom lenses. Each day, we made a semi-random circuit through the 90,000 acre nature conservancy, which contains several hundred wild Plains zebras, and fewer than 20 endangered Grevy's zebras (some of which are included in our dataset). Two people were stationed on top of the vehicle to take pictures while the driver circled around individual groups of zebras, so as to capture both flanks of the animal.

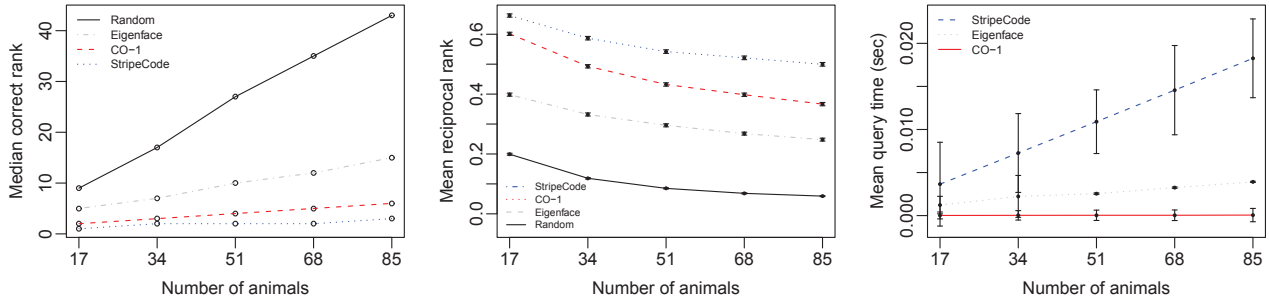
We collected as many pictures as possible of each flank of an animal in different positions in its natural walking gait. As a result, a number of pictures are quite similar. A professionally trained field assistant identified the images based on a database of prior sightings stretching back almost ten years. All but a few zebras were reliably identified. The manual identification method involves assigning each zebra an ad-hoc code based on the pattern of stripes along its shoulder. These (textual) codes are then fed into a stock photo organizing program that searches metadata for a similar code string. The final match is made by direct visual observation by a professional.[6] We determined after manual identification that several animals were observed on multiple days, which makes our dataset more representative of intended usage.

Figure 4 shows typical pictures, and manually cropped ROIs, from our dataset. In particular, they illustrate the difficulty of automatic animal segmentation and the problems associated with perspective skews.

6. EXPERIMENTAL RESULTS

We measured the accuracy of three animal recognition algorithms: Eigenface [28], the CO-1 algorithm based on multi-resolution histograms of differential image features [22], and the method we propose in this paper: StripeCodes. As the number of unique animals in the database grows, the accuracy of any algorithm is likely to suffer. We therefore chose the number of unique animals in the database as the primary independent variable for our analysis. Furthermore, analysis is restricted to animals that are already contained in the database, since it is difficult to define accuracy (and

[6]This is only tractable manually because a trained professional can distinguish a *mis*match very quickly.



(a) Median rank, $ipa = 1$. (b) MRR, $ipa = 1$. (c) Mean query time, $ipa = 1$.

Figure 5: Performance of various algorithms.

efficacy in general) when an animal has never been observed before.

6.1 Testing Methodology

Assume that the dataset consists of a total of I images of A different animals ($I > A$). In our dataset, almost all animals have images of both their flanks, but we treat each flank as a different animal. This is because zebra stripe formation is a chaotic process which is not fully understood [13], so there is no basis for assuming symmetry. Our evaluation methodology has two independent parameters: *ipa*, which specifies the images per animal contained in the database, and N, the number of unique animals in the database. Keeping a fixed number of the most recent images of each animal in a production database is desirable not only for efficiency in database scans, but also as a form of temporal smoothing to account for natural changes in the animal's appearance.

For fixed values of *ipa* and N, the following procedure samples uniformly from all possible database and query pairs, and returns the primary evaluation measure: the rank of the correct animal.

1. Choose $N \leq A$ animals at random.
2. For each animal a, choose *ipa* images randomly and add to the database.
3. From *all* the remaining images of the N animals, choose an image randomly as the query image.
4. Rank each animal in the database by the minimum distance of a database image to the query image. Return the rank of the correct animal.

We use three statistics over many random iterations of the loop above to evaluate algorithm performance at each value of N and *ipa*: the *mean reciprocal rank* (MRR), popular in text information retrieval [30], the median rank of the correct animal, and the average query time of our implementation. Note that this puts the Eigenface method at somewhat of a disadvantage, since we used a Matlab implementation of it, instead of the native C++ implementations of other methods.

6.2 Results

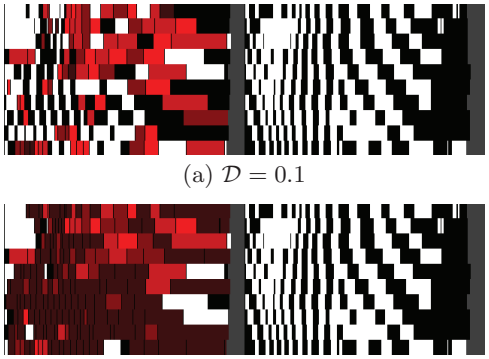
We set N to be successive fifths of the dataset size, and report results at an *ipa* value of 1. We used 5,000 random iterations for each pair of *ipa* and N values, for all algorithms. The baseline used was a random (uniform) permutation of the ranks of the animals in the database. For the $CO-1$ algorithm, we used histograms at 4 different scales, proceeding in half-octaves as in the original study [22], and 10 bins for each feature. All runs were performed on an AMD Athlon x2 processor with 2 GB of RAM running Ubuntu Linux.

6.2.1 Accuracy

Figure 5 shows the performance of all algorithms for $ipa = 1$. Note that a lower median correct rank indicates better performance, but a higher MRR is considered better. An optimal algorithm would always return the correct animal at rank 1, so its MRR would be 1, with sub-optimal algorithms having an MRR strictly less than 1. The MRR places a greater emphasis on searches where the correct animal is closer to the top of the result list. This is in contrast to, for example, the mean rank of the correct animal, which is significantly affected by outlier queries (*i.e.*, where the correct animal is closer to the bottom of the result list). The median correct rank is more indicative of performance that might be perceived by a human.

The StripeCode algorithm, while being the simplest to implement, also outperforms all the other methods. Its median correct rank stays consistently low as the database grows. Even with 85 animals in the database, its median correct rank is less than 5. The MRR curve as the database grows is also consistently higher than the closest competitor. The poor performance of Eigenface is not surprising, given that it was originally developed for spatially aligned, exposure-controlled images of human faces.

The actual query time of our implementation is, however, higher than competing methods because of the relatively expensive edit distance computation. Although it grows linearly, the growth is much faster than the CO-1 algorithm. This is not surprising, because a distance computation between two images in the CO-1 algorithm is reduced to (after preprocessing) computing an inner product between two relatively small numerical vectors, a highly efficient procedure on modern processors. StripeCodes are inherently slower, since they share the same foundation as sequence alignment algorithms used in bioinformatics. A number of modern technologies such as GPU computing have resulted in impressive performance gains for edit distance-like computations [16]. Our implementation, in contrast, is quite plain and could be greatly improved by implementing these modern computational techniques. Furthermore, the database



(a) $\mathcal{D} = 0.1$

(b) $\mathcal{D} = 0.5$

Figure 6: Comparing stripes from two images of the same zebra. Red blocks in the left image are matched to blocks in the right image, with a brighter color indicating a lower cost. Black and white blocks in the left image represent insertions and deletions.

search itself is trivially parallelizable. We also note that the one-time feature extraction from an image is not included in Figure 5. The average time to extract a StripeCode from an image was less than a second, whereas the CO-1 algorithm took an average of 8.1 seconds.

6.2.2 *Transparency*

Unlike the two other algorithms we tested, StripeCodes can supply human-interpretable visual feedback on why two images match. In a computer-assisted system, which comprises the majority of individual animal identification use cases, this is essential in helping a human operator make the final decision about a match. To visualize a match between two images, we can plot the edit paths of each pair of StripeStrings as a heat map. This is illustrated in Figure 6, where a reference image is displayed side-by-side with another image of the same animal. Each color block in the reference image retains its original color if it was omitted as part of the optimal dynamic programming path. If it was not deleted, the block is colored red according to the cost incurred for the optimal path. We used k-means clustering on costs to segment them into four discrete shades of red, with brighter colors representing better matches.

To the best of our knowledge, ours is the only method where such visual feedback is available to the user. It can also be used to demonstrate the effect of the $\mathcal{D}$ cost on the optimal editing path, as we do in Figure 6. For a low value of $\mathcal{D} = 0.1$, insertions and deletions are favored over matching color blocks of different lengths. At a higher $\mathcal{D}$ value, insertions and deletions become more expensive, and blocks are matched even if they incur a relatively higher cost. We found values in the range $0.4 \leq \mathcal{D} \leq 0.6$ to be effective for our dataset.

6.2.3 *Follow-up test*

In order to account for natural aging and appearance changes in animals, we ran a follow-up test in the same area 13 months after the original dataset was collected. We tested the StripeCodes algorithm against the CO-1 algorithm on 83 new Grevy's zebra photos taken over a 3 day period while driving fixed routes. Note that the Grevy's zebra, shown in Figure 4(d), has a higher stripe density than the more common Plains zebra and is therefore more susceptible to noise. Testing with an ROI located on the flank of the zebra, and using a paired t-test, we found that StripeCodes outperformed CO-1 with $t = 3.85$ and $df = 82$ for $p < 0.0002$. The means on an ordinary t-test were 9.2 ± 1.3 (StripeCodes) and 15.3 ± 1.7 (CO-1).

7. CONCLUSION

We have developed a similarity algorithm for comparing animal coat markings across noisy images, designed to be a part of a computer-assisted system for individual animal identification. Our *StripeCode* algorithm is based on a novel feature extraction and matching method that capitalizes on the high resolution of modern digital cameras, and offers a number of benefits over other approaches:

- It offers state of the art retrieval performance, while being extremely simple to implement, and is tolerant of scale, exposure, occlusion, and mild perspective skews. Since it can be seen as a variant of sequence alignment algorithms used in bioinformatics, implementations of it can directly benefit from numerous algorithmic and hardware optimizations developed for algorithms like the Smith-Waterman algorithm [16].

- It is applicable to any animal with prominent coat markings consisting of relatively large morphological features and a small number of distinctive colors. Some examples are zebras, tigers, giraffe, and kudu.

- It offers visual feedback to a human user, essential as part of a computer-assisted system, on why two images of animal coat markings are similar.

We look forward to augmenting our open-access datasets in the future with libraries of other species, such as giraffe, as well as repeat sightings of the animals currently included. By releasing our code and data publicly, we hope to offer a common framework for the comparison of future algorithms.

8. ACKNOWLEDGEMENTS

This work is part of a project performed in the joint Princeton-UIC Computational Population Biology Course in Spring 2010[7], with co-instructors Tanya Berger-Wolf (University of Illinois at Chicago), Daniel Rubenstein and Iain Couzin (Princeton University), who were instrumental in several parts of this research. We thank the Kenya Ministry of Education, Science and Technology (research permit MOST 13/001/29C 80Vol.11 to D.I. Rubenstein), the staff at Mpala Resarch Centre, Kenya and fellow graduate students at EEB-Princeton University and CS at University of Illinois at Chicago. Funding was provided by Department of Ecology and Evolutionary Biology of Princeton University, generous contributions by Bill Unger (for the UIC students in the course), UIC College of Engineering, Department of Computer Science at UIC, UIC Graduate Research and Provost's Awards (Lahiri), NSF III-CXT 0705311 (Rubenstein) and IIS-CTX-0705822 and NSF IIS-CAREER-0747369 (Berger-Wolf). We thank Anthony Roy for assistance with processing our dataset. The follow-up analysis was run by Victoria Zero and the students of Princeton's 'Natural History of Mammals' class.

[7] http://compbio.cs.uic.edu/∼tanya/teaching/KenyaCourse.html

9. REFERENCES

[1] A. Ardovini, L. Cinque, F. Della Rocca, and E. Sangineto. A semi-automatic approach to photo identification of wild elephants. *Pattern Recognition and Image Analysis*, pages 225–232, 2007.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 509–522, 2002.

[3] H. Blum and R. N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10(3):167–180, 1978.

[4] K. Bradfield. Photographic identification of individual Archey's frogs, *Leiopelma archeyi*, from natural markings. Doc. Science Internal Series 191, Dept. of Conservation, New Zealand, 2004.

[5] T. Burghardt and N. Campbell. Individual animal identification using visual biometrics on deformable coat patterns. In *Proceedings of the 5th International Conference on Computer Vision Systems, Berlin, Germany.. Accessed*, volume 9. Citeseer, 2007.

[6] T. Caro and S. Durant. Use of quantitative analyses of pelage characteristics to reveal family resemblances in genetically monomorphic cheetahs. *Journal of Heredity*, 82(1):8, 1991.

[7] H. Cheng, X. Jiang, Y. Sun, and J. Wang. Color image segmentation: advances and prospects. *Pattern Recognition*, 34(12):2259–2281, 2001.

[8] E. Costenbader and T. W. Valente. The stability of centrality measures when networks are sampled. *Social Networks*, 25(4):283 – 307, 2003.

[9] M. Daniels, D. Balharry, D. Hirst, A. Kitchener, and R. Aspinall. Morphological and pelage characteristics of wild living cats in scotland: implications for defining the 'wildcat'. *Journal of Zoology*, 244(2):231–247, 1998.

[10] S. Elbin and J. Burger. In my experience: Implantable microchips for individual identification in wild and captive populations. *Wildlife Society Bulletin*, pages 677–683, 1994.

[11] G. Foster, H. Krijger, and S. Bangay. Zebra fingerprints: towards a computer-aided identification system for individual zebra. *African Journal of Ecology*, 45(2):225–227, 2007.

[12] C. Gope, N. Kehtarnavaz, G. Hillman, and B. Wursig. An affine invariant curve matching method for photo-identification of marine mammals. *Pattern Recognition*, 38(1):125–132, 2005.

[13] B. Jonathan. A unity underlying the different zebra striping patterns. *J. Zoology*, 183(4):527–539, 1977.

[14] M. Kelly. Computer-aided photograph matching in studies using individual identification: an example from Serengeti cheetahs. *Journal of Mammalogy*, 82(2):440–449, 2001.

[15] A. Kreho, N. Kehtarnavaz, B. Araabi, G. Hillman, B. Wursig, and D. Weller. Assisting manual dolphin identification by computer extraction of dorsal ratio. *Annals of biomedical engineering*, 27(6):830–838, 1999.

[16] S. Manavski and G. Valle. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. *BMC bioinformatics*, 9(Suppl 2):S10, 2008.

[17] A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15:926–932, 1993.

[18] R. O'Callaghan and D. Bull. Combined morphological-spectral unsupervised image segmentation. *IEEE Trans. on Image Processing*, 14(1):49–62, 2005.

[19] E. Petrakis, A. Diplaros, and E. Milios. Matching and retrieval of distorted and occluded shapes using dynamic programming. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 1501–1516, 2002.

[20] K. Pollock, J. Nichols, C. Brownie, and J. Hines. Statistical inference for capture-recapture experiments. *Wildlife Monographs*, pages 3–97, 1990.

[21] D. Pountain. Run-length encoding. *Byte*, 12(6):317–319, 1987.

[22] S. Ravela and L. Gamble. On recognizing individual salamanders. In *Proceedings of Asian Conference on Computer Vision, Ki-Sang Hong and Zhengyou Zhang, Ed. Jeju, Korea*, pages 742–747, 2004.

[23] B. M. Romeny. Gaussian derivatives. In M. Viergever, editor, *Front-End Vision and Multi-Scale Image Analysis*, volume 27, pages 53–69. Springer Netherlands, 2003. Gaussian derivatives.

[24] B. Shorrocks and D. P. Croft. Necks and networks: a preliminary study of population structure in the reticulated giraffe (*giraffa camelopardalis reticulata*). *African Journal of Ecology*, 47(3):374–381, 2009.

[25] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999.

[26] S. Sundaresan, I. Fischhoff, J. Dushoff, and D. Rubenstein. Network metrics reveal differences in social organization between two fission–fusion species, Grevy's zebra and onager. *Oecologia*, 151(1):140–149, 2007.

[27] P. Taberlet and G. Luikart. Non-invasive genetic sampling and individual identification. *Biological Journal of the Linnean Society*, 68(1-2):41–55, 1999.

[28] M. Turk and A. Pentland. Face recognition using Eigenfaces. In *Proc. CVPR '91*, pages 586–591. IEEE, 2002.

[29] E. Vidal, A. Marzal, and P. Aibar. Fast computation of normalized edit distances. *IEEE Trans. Pat. Analysis and Machine Intelligence*, 17(9):899–902, 2002.

[30] E. Voorhees. The TREC question answering track. *Natural Language Engineering*, 7(04):361–378, 2001.

[31] R. Wagner and M. Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

[32] B. Wursig and T. Jefferson. Methods of photo-identification for small cetaceans. *Reports of the International Whaling Commission*, 12:43–52, 1990.

[33] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi. Hardware design experiences in ZebraNet. In *Proc. 2nd Intl. Conf. on Embedded networked sensor systems*, pages 227–238. ACM, 2004.