Using Eigenvectors to Identify Animal Footprints

J. Brettle and T. Jewell-Alibhai

March 2, 2020

1 Summary

The detection and classification of animal footprints has a crucial application in wildlife monitoring, as one of the only truly noninvasive techniques [1]. Thus far, classification of footprints has been restricted to species classification by eye and relatively labor intensive individual animal classification through an algorithm that works off of human entered landmark points [2]. Automating a system to detect species and/or individuals would increase monitoring capability and reduce necessary training requirements to be able to monitor animals in a noninvasive way. Our system uses a combination of a relatively automated image processing system and principal component analysis to determine the species of an animal from its footprint. Despite having a high level of accuracy on the species level, the algorithm is not consistent at identifying individuals. Future iterations of this system could make the entire image processing process automated, and also increase the algorithm's ability to determine individual animals accurately.

2 Introduction

The concept of using footprints to monitor wild animals is a useful noninvasive method of keeping track of a population. Footprints are relatively easy to find in most places and taking images of footprints requires relatively little technology when compared to GPS or aerial tracking. This concept can hugely benefit reserves and parks looking to keep track of endangered species, but could also potentially be exploited by illegal poaching to identify animals.

The algorithm is designed to identify species (and potentially individuals) of animals by their footprints. This works using principal component analysis (PCA), by taking each image and converting it into a vector. When all the images have been compiled into a matrix, the system identifies the x principal eigenvectors, and uses these to calculate the closest match to each image in the training set using the "nearest neighbor" approach. The system can then identify what species the footprint comes from based on the species of the closest match.

This could be used to identify animal species in the wild, or to help track individual animals outside of captivity. The ability to track animals in this passive way (without chips, tags, or GPS) could be a lot healthier for the animals as well as allow more amateur trackers to help keep tabs on endangered species. However, if these photos are geotagged and published, poachers and others with malicious intent could more easily find and hurt the wildlife.

More broadly, we are exploring the ability of Principal Component Analysis based recognition technology to differentiate and categorize animals by their footprint. On a deeper level, we are determining the accuracy with which footprints can be linked back to the individual using this method.

3 Methods

Our method relies on a combination of manual and automated image processing and principal component analysis (PCA) to standardize footprint images and classify them into species and individual. The first step in image standardization is to crop each image such that the edges of the footprint touch the four edges of the cropped image. The images are then converted to grayscale and have their shortest side stretched to be

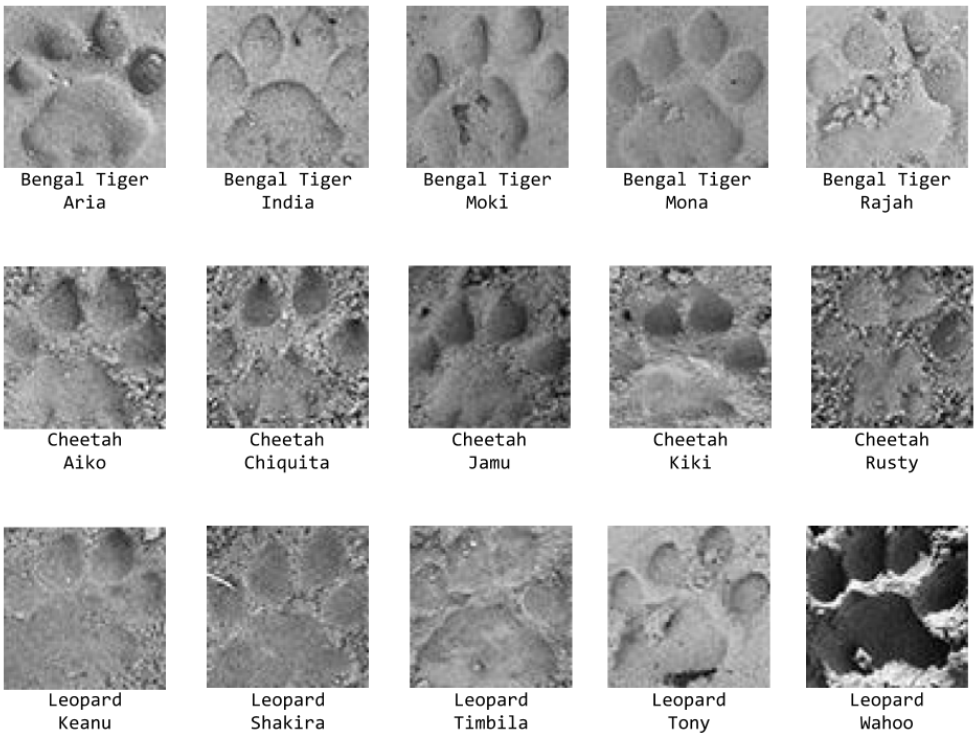

Figure 1: These are examples of the 64 by 64 pixel images (before mean-centering) that we fed into MATLAB. We used the footprints of three species of big cats: Bengal Tigers, Cheetahs, and Leopards, and multiple images of footprints for each of five captive individuals (named above) of each species.

the same length as their longer side. These square images are then exported at 64 by 64 pixel resolution and 8 bit-depth for use in MATLAB.

After the images have been standardized, they are added to one of two folders: one for a 'test' data set which consists of images that are effectively unknowns and must be classified and another folder for 'training' images which are known and can be compared to the test images to classify the test images.

An initial function in MATLAB takes each folder in turn and vectorizes the images, turning them into a $n \times 1$ matrix where n is the total number of pixels in each image, so that each image is stored in a single column. Then the images are appended into a matrix which is of size $n \times m$ where m is the number of images in the folder. All the data is mean-centered, where the mean of the data is subtracted from each value so that the mean now lies at 0. The covariance matrix is then found, where:

if A is a the $n \times m$ matrix of vectorized images, the $n \times n$ covarience matrix C is calculated by

$$C = A^T A$$

This creates a symmetric matrix with comparisons between each pixel of each image. Once this is done, it is simple to find the eigenvectors, which represent the most significant variation trends in the pixels. The x number of eigenvalues with the largest respective eigenfaces are the principal components, which are then used to compare the test and training data sets. Each image vector is simplified (compressed) so that it is a representation of the principal components. Since there are x principal components, each image is now represented by x values. Our method uses from 6-10 principal components, so this simplifies each image immensely.

Once all the images are compressed in this way, they can be compared using the nearest neighbor algorithm, which calculates the closest image vector from the training set to any given image vector from the unknown set, using the principal components as the dimensions. Once the closest match has been found, the known data from the closest match is used to classify the unknown.

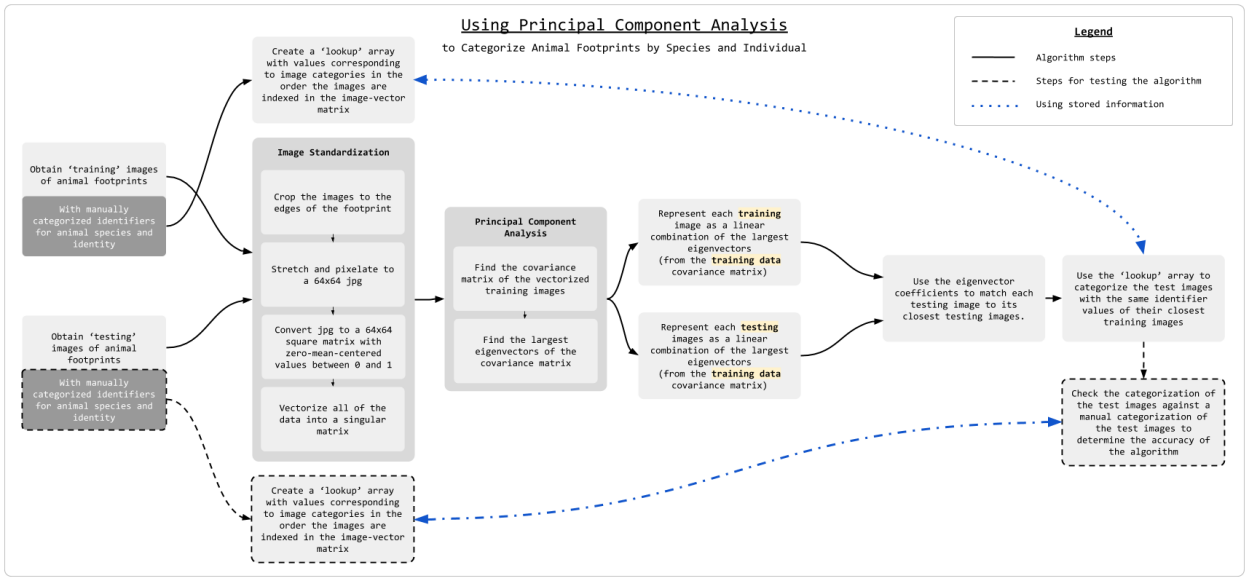

Figure 2: Our method can be briefly described through this block diagram. Within the grey boxes labelled `Image Standardization` and `Principle Component Analysis` are processes we performed on both the training and testing data sets. The dark grey blocks with white text represent the correct manual categorization data for the corresponding data sets, such as `Species: Cheetah, Individual Named: Aiko`. Dashed lines and block-frames represent steps completed when analyzing the algorithm (when one knows the correct categorization of footprints) but skipped if one were using this algorithm on unknown data.

4 Detailed Findings

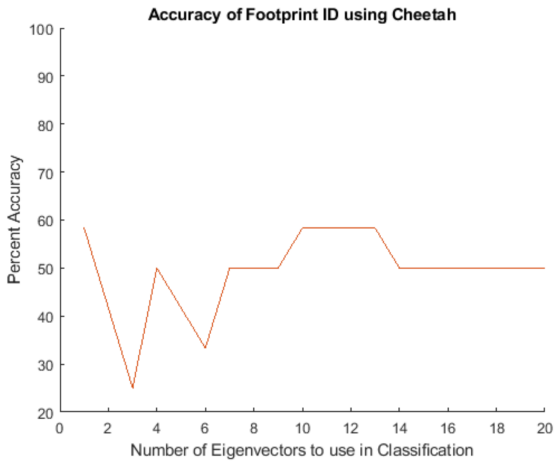

Figure 3: When identifying individual cheetahs within a cheetah-only data set, 10 to 12 eigenvectors seemed to be the ideal number to use for PCA. Information and related data collected about different groups of animals helped decide the number of eigenvectors used in our subsequent simulations.

After image sorting and standardization, our combined data sets included a total of around 100 images from 3 different animals: Bengal Tiger, Cheetah, and Leopard. Each of these species included 5 individuals

with between 3 and 8 images per individual. When we used around a third of the data as test and the rest as training data (chosen arbitrarily, but making sure there were at least 2 prints of each animal left in the training set) the species accuracy was around 90 percent and the individual accuracy was close to 50 percent. The more prints in the training set, (and consequently less in the test) the higher the individual accuracy. At one fifth of the footprints as test images (20:80), the species accuracy remained around 90 percent but the individual accuracy jumped to 80 percent. This is indicates that individual accuracy is heavily reliant on number of images to compare to.

This is not a surprising result, and is actually relatively high considering that it can be challenging to visually differentiate the footprint. The lack of individual accuracy could be attributed to the relatively small number of footprints per individual animal, as there were only about 100 images spread among 15 individuals. With a larger data set, the accuracy of determining individuals could potentially be much higher. This system, in its current iteration, is probably not a huge improvement on current and existing systems as it still requires some human interaction to process the images. However, it demonstrates the ability of PCA to identify species and potentially individual animals and provides an encouraging outlook on the ability of this method to supplant other manual and computer-aided animal-footprint identification methods.

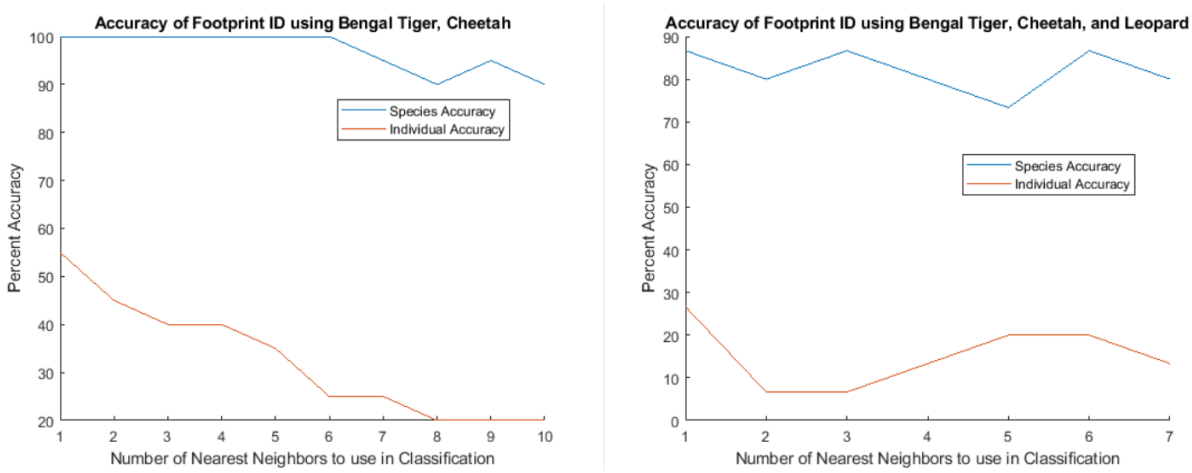

Figure 4: These two graphs of accuracy with a varying number of nearest neighbors considered indicates two primary things. 1) Pre-sorting by species before calculating the relevant eigenvectors seems to dramatically increase individual identification accuracy (a jump from 28 to 55 percent just by removing one species). This supposition is further supported by the 60 percent accuracy shown for individual identification for only one species in Figure 3. 2) Taking the mode of an increasing number of nearest neighbors (as opposed to simply using the categorization of the first nearest neighbor) seemed to dramatically reduce accuracy rather than make the program more consistent.

5 Recommendations

5.1 For use:

Identifying animal species from their footprints can be accomplished with PCA – a human with enough training can also do it – but identifying individuals is far harder. This can primarily be attributed to the fact that there are clear differences between footprints of species (as you would expect) but the differences between individuals of the same species are much subtler.

We know that it is possible for a combination of human and computer systems to identify individuals, as Wildtrack is able to do this with over 95 percent accuracy [3]. However, this system is both labor intensive and requires extensive training to use the system. If a combination of automated image processing and PCA could identify individuals, the work of identifying and monitoring endangered animals could be simplified

```
result = 3x6 cell array
    {'img_12345678'}    {'img_91011121'}    {'img_31415161'}    {'img_71819202'}    {img_122232425'}    {'img_26272829'}
    {'Cheetah     '}    {'Bengal Tiger'}    {'Leopard     '}    {'Bengal Tiger'}    {'Cheetah     '}    {'Leopard     '}
    {'Rusty       '}    {'India       '}    {'Timbila     '}    {'India       '}    {'Jamu        '}    {'Timbila     '}
```

Figure 5: While a more suitable user interface should be implemented for field use, the MATLAB cell-array output can display the raw identification the program performs. The first row represents the name of the image being compared, and the second two rows represent the algorithm's categorization based on its calculation of the nearest neighbor.

and sped up drastically.

The primary steps to automation are 1) making the image processing automatic, by finding a way to create a bounding box around the footprint automatically and cropping it, and 2) improving the current PCA method to successfully identify individuals with high accuracy.

5.2 For improvement:

We propose that improving our standardization procedure for paw-print images could improve the accuracy of individual-print recognition. As the MATLAB algorithm does not consider the possible rotation of the footprints in its comparison, we would rotate each image to a particular landmark direction. For example, prints of the above general shape (that of a big cat) would be rotated such that the second toe centered at the top of of the crop. This should ensure that multiple footprints of the same individual are oriented the same way in each image.

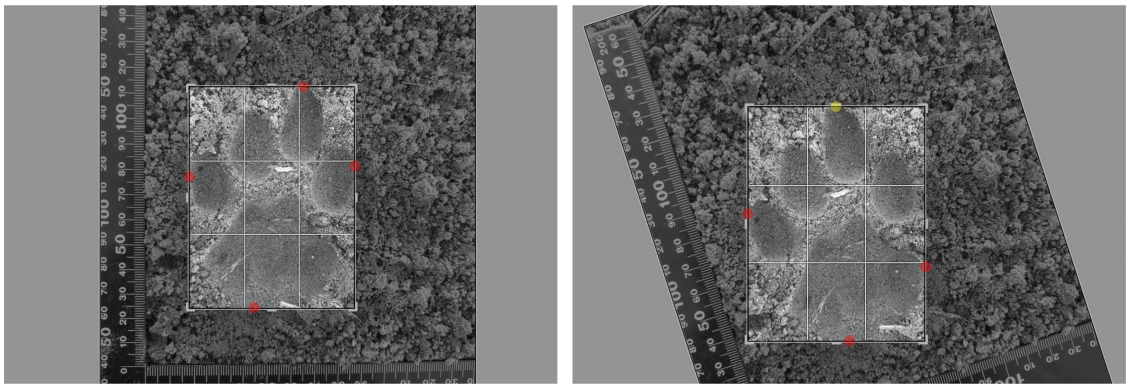

Figure 6: The red and yellow dots in the cropping images above mark the locations where the edges of the footprint touch the edges of the crop. The image on the left is an example of the current cropping method - an arbitrarily upright view of the print, and the image on the right is an example of the proposed cropping method, with the second toe centered on the top edge of the crop (represented by the yellow dot).

6 References

Wildtrack.org

[1] wildtrack.org/why-non-invasive-2

[2] wildtrack.org/how-fit-works

[3] wildtrack.org/10-reasons-to-use-fit

7 Appendix

Find the code and images at github.com/julesbrettle/footprint-id-project.

```
% Identifying Footprints Using Principle Component Analysis
% J. Brettle and T. Jewell-Alibhai
% 2020-03-02

clear all;

% function dir2matrix takes an input directory and locates all
% images with a .jpg extension. It then takes the images, vectorizes
% them and then grayscales them for use with PCA, adding them to a
% matrix.
[train, train_validate] = dir2matrix('footprint_images');
[test, test_validate] = dir2matrix('footprint_images\Test');
numeigs = 12; %number of eigenvectors to use in PCA

[NN_S, NN_ID] = nearestNeighbors(numeigs,train,test,train_validate,test_validate);

% rowMode() converts vertical cell array to horizontal
NN_S = rowMode(NN_S);
NN_ID = rowMode(NN_ID);
NN_R = [NN_S'; NN_ID'];

result = vertcat(test_validate,NN_R)


function [mydata, myvalidate] = dir2matrix(Dir)
    % function dir2matrix takes an input directory and locates all
    % images with a .jpg extension. It then takes the images, vectorizes
    % them and then grayscales them for use with PCA, adding them to a
    % matrix.
    jpegFiles = dir(fullfile(Dir,'*.jpg'));
    numfiles = length(jpegFiles);
    width = size(imread(fullfile(Dir,jpegFiles(1).name)),1);
    height = size(imread(fullfile(Dir,jpegFiles(1).name)),2);
    mydata = zeros(width*height, numfiles);

    for k = 1:numfiles
        F = fullfile(Dir,jpegFiles(k).name);
        temp1 = rgb2gray(double(imread(F))./255);
        %imagesc(temp1);
        temp2 = imresize(temp1,[width*height,1]);
        mydata(:,k) = temp2;
    end

    myvalidate = cell(2,numfiles);

    for z = 1:numfiles
        if(length(strfind(jpegFiles(z).name,'C_')) > 0)
            myvalidate(1,z) = {'Cheetah'};
        elseif(length(strfind(jpegFiles(z).name,'BT_')) > 0)
            myvalidate(1,z) = {'Bengal Tiger'};
        elseif(length(strfind(jpegFiles(z).name,'WR_')) > 0)
            myvalidate(1,z) = {'White Rhino'};
        elseif(length(strfind(jpegFiles(z).name,'P_')) > 0)
            myvalidate(1,z) = {'Puma'};
        elseif(length(strfind(jpegFiles(z).name,'L_')) > 0)
            myvalidate(1,z) = {'Leopard'};
        end

        temp1 = string(extractBetween(jpegFiles(z).name,"_"," ("));
        myvalidate(2,z) = {temp1(1)};
    end
end

function [NN_S, NN_ID] = nearestNeighbors(numPrincipalComponents, train, test, trainV, testV)
```

```
65
66      ATrain = train';
67      ATest = test';
68
69      % mean center the train and test data
70      ATrain = ATrain - mean(ATrain);
71      ATest = ATest - mean(ATest);
72
73      % get covariance matrix of the training data
74      covar = ATrain'*ATrain;
75
76      % if you wan to just get the eigenvectors with largest eigenvalues
77      % you can use eigs. Delta (diagonalized eigenvalues) is not used here.
78      [Q, Delta] = eigs(covar, numPrincipalComponents);
79
80      % project into face space
81      faceSpaceTrain = Q'*ATrain';
82      faceSpaceTest = Q'*ATest';
83
84      % Use nearest neighbor search (you can code this manually if you'd like)
85      NN = knnsearch(faceSpaceTrain', faceSpaceTest', 'K',1);
86      NN_S = cell(size(NN));
87      NN_ID = cell(size(NN));
88      for x = 1:size(NN,1)
89          NN_S(x,1:size(NN,2)) = trainV(1,NN(x,:));
90      end
91      for y = 1:size(NN,1)
92          NN_ID(y,1:size(NN,2)) = cellstr(string(trainV(2,NN(y,:))));
93      end
94  end
95
96  function a = rowMode(matrix)
97      % rowMode() converts vertical cell array to horizontal
98      a = cell([size(matrix,1) 1]);
99      for x = 1:size(matrix,1)
100         y = unique(matrix(x,:));
101         n = zeros(length(y), 1);
102         for iy = 1:length(y)
103             n(iy) = length(find(strcmp(y{iy}, matrix(x,:))));
104         end
105         [~, itemp] = max(n);
106         a(x,:) = y(itemp);
107     end
108     a;
109 end
```