Automatic Identification of Individual African Leopards in Unlabeled Camera Trap Images

Cheng Guo, *Student Member, IEEE*, Agnieszka Miguel, *Senior Member, IEEE*, and Anthony A. Maciejewski, *Fellow, IEEE*

***Abstract*— This article describes an algorithm to solve the real-world animal identification problem, i.e., determine the unknown number of K individual animals in a dataset of N unlabeled camera-trap images of African leopards, provided by *Panthera*. To determine the leopards' IDs, we propose an effective automated algorithm, that consists of segmenting leopard bodies from images, scoring similarity between image pairs, and clustering followed by verification. To perform clustering, we employ a modified ternary search that uses a novel adaptive k-medoids++ clustering algorithm. The best clustering is determined using an expanded definition of the silhouette score. A new post-clustering verification procedure is used to further improve the quality of a clustering. The algorithm was evaluated using the *Panthera* dataset that consists of 677 individual leopards taken from 1555 images, and resulted in a clustering with an adjusted mutual information score of 0.958 as compared to 0.864 using a baseline k-medoids++ clustering algorithm.**

***Note to Practitioners*—We proposed an effective automated algorithm to solve the real-world animal identification problem: identifying K unknown individual animals in N images of a given species, with most animals only represented by a single image. This algorithm is different from other methods that assume all images in a dataset are from known individuals and thus regard the animal ID problem as a retrieval identification task. Our approach consists of a new adaptive k-medoids++ clustering algorithm and a novel post-clustering verification procedure. The clustering is performed based on the degree of similarity between all image pairs in the dataset with the result validated using an expanded definition of the silhouette score. The accuracy of our algorithm was demonstrated on a real-world image dataset of African leopards, a small dataset with a relatively large ratio of K/N, provided by *Panthera*. Code has been made available at: https://github.com/obaiga/Automatic-individual-animal-identification.**

***Index Terms*— Computer vision for automation, robotics and automation in life sciences, object detection, segmentation and categorization, automated animal identification, camera-trap images, clustering.**

Manuscript received 11 August 2023; revised 27 December 2023 and 16 February 2024; accepted 11 March 2024. Date of publication 26 March 2024; date of current version 7 February 2025. This article was recommended for publication by Associate Editor Z. Liu and Editor L. Zhang upon evaluation of the reviewers' comments. *(Corresponding author: Cheng Guo.)*

Cheng Guo and Anthony A. Maciejewski are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523 USA (e-mail: Cheng.Guo@colostate.edu; aam@colostate.edu).

Agnieszka Miguel is with the Department of Electrical and Computer Engineering, Seattle University, Seattle, WA 98122 USA (e-mail: amiguel@seattleu.edu).

Digital Object Identifier 10.1109/TASE.2024.3379553

I. Introduction

FOR wild animals that are elusive and seldom seen, camera traps offer a low-cost and non-invasive method to obtain a large number of images that may contain animal information. Camera traps consist of numerous cameras at fixed locations throughout suspected habitats. The cameras are automatically activated by motion or infrared sensors, so that they may be triggered by moving animals, swaying vegetation, or sudden changes in the weather. Therefore, captured images not only contain a variety of animals, but frequently consist only of rocks and vegetation. At present, deep learning is an effective solution to classifying different species in a set of images [1], however, automatically identifying the individual animals from the same species, referred to as animal identification (or animal ID), is one of the current challenges in ecology [2].

Currently, animal ID is treated as an extension of the object or face recognition problem [3], [4], [5]. It is important to note that most works [6], [7], [8] assume that the individual animal dataset for a given species is a closed set. Thus the animal ID problem is regarded as a retrieval identification problem (Re-ID), i.e., given a new image, assign it an animal ID from the known dataset. This implies that there is at least one image for each individual animal label in the dataset. Recent works [9], [10] consider the case where new images may not correspond to known individuals, i.e., the individual animal dataset is an open set. Unfortunately, these techniques still do not address the "real-world problem" where a labeled dataset does not exist and even the *number* of classes is unknown. In addition, it is likely that, in a given camera trap dataset, there will be many animals for which only a single image is available.

In this paper, we consider the problem of automatically identifying an unknown number of K individual animals in a given dataset of N unlabeled camera trap images of African leopards, provided by *Panthera* [11]. Several factors result in N being relatively small, and K/N being relatively large. These factors include the behavior of African leopards, the location of camera traps, and that only one image is taken when a trap is activated (due to the limitations of the flash technology). In addition, the images have a wide variation in viewpoints, exposures, occlusions, and quality [12]. To formally state the problem under consideration: Given a set of images $\mathcal{I}$ with the following properties (1) $|\mathcal{I}| = N$ is small and (2) every image $I_x \in \mathcal{I}$ is known to contain exactly one leopard from an unknown number of K leopards whose images

do not exist outside of $\mathcal{I}$; determine K and the subsets C_i, where $1 \leq i \leq K$, that consist of unique individuals and $\bigcup_{i=1}^{K} C_i = \mathcal{I}$.

The main contribution of this paper is the design of an effective algorithm to solve the real-world animal identification problem as distinct from the animal Re-ID problem. That is, given an unlabeled image dataset of African leopards, the algorithm groups the images into subsets of individual leopards, and thus determines the number of distinct individuals in the dataset. Because traditional unsupervised techniques result in unacceptably low accuracy when applied to animal ID problems [13], we (1) develop a novel adaptive k-medoids++ clustering algorithm, (2) design a novel post-clustering verification procedure to further improve accuracy, and (3) define the silhouette score of a single-item cluster to measure and enhance the clustering performance.

The remainder of this paper is organized as follows. Section II reviews related work on the animal identification problem and provides background on existing techniques used by our algorithm. The next section explains the details of our new algorithm for automatically identifying individual African leopards in unlabeled images. The evaluation of the algorithm is given in Section IV. Finally, Section V concludes this article.

II. Background

A. Related Literature on Animal ID

There are several computer vision techniques that are frequently used to solve the animal identification problem, e.g., segmentation, feature extraction, and classification [3]. The goal of segmentation is to separate the animal from the background. Segmentation techniques range from manual extraction [12], [14] and classical image processing algorithms [15], [16], to deep learning methods [7]. The choice of feature extraction technique frequently depends on the animal being identified. For animals with unique identifiable markings, such as leopards, zebras, and tigers, traditional feature extraction methods can be used, e.g., Local Binary Pattern (LBP) [15] or Scale-Invariant Feature Transform (SIFT) [12], [14]. For animals without distinctive identifiable markings, such as bears, wolves, and sheep, deep learning methods are more frequently employed, e.g., Convolutional Neural Networks (CNN) [6]. Similarly, the type of classification model employed is generally determined by the size of the animal image dataset, more specifically, the quantity of images corresponding to individual animals. If there are only a small number of images for some individual animals, then it is common to employ a matching score mechanism between image pairs [12], [17]. The magnitude of this matching score determines if the image pair represents the same individual. For cases where there are a large number of images for each individual, machine learning techniques are frequently employed, e.g., Support Vector Machine (SVM) [6] or Siamese and Triplet-Loss networks [7], [18].

The above classification techniques all apply to closed sets. As previously discussed, techniques that require a closed-set assumption are too restrictive for the problem considered here because they can not handle images of individuals that are not already in the classified set. It is, in general, possible to apply a number of traditional unsupervised learning approaches to open set problems [3], [19]. However, to be effective the datasets on which they operate must contain large numbers of images for each of the individuals. Unfortunately, that is not true for the animal identification problem that we consider here. For most of individuals in the dataset only a single image exists. The closed-set assumption has been extended to include an additional "unknown identity" class [9], however, this does not address the classification of multiple new unknown individuals. To address this issue, Stewart et al. [10] treat the animal ID problem as a continual curation problem. In their work, they allow the algorithm to ask for human assistance when a new image does not sufficiently match an existing individual in the already classified dataset.

The goal of our work is to solve the real-world automated animal identification problem, i.e., to develop an algorithm that identifies all individuals in an unlabeled African leopard image dataset without expert assistance. Our new algorithm uses existing segmentation and feature extraction techniques, however, the classification approaches discussed above are not applicable to unlabeled datasets. Therefore, we develop an adaptive clustering algorithm to group the images into classes of individual leopards.

B. Background on Hotspotter

We use the feature extraction technique from Hotspotter [12] to compute an $N \times N$ similarity score matrix for the N images in an African leopard image dataset. Hotspotter uses a variant of SIFT (scale-invariance feature transform) [20], [21] to identify all keypoints in an image, which are typically the locations of corners or edges. Each keypoint is associated with a descriptor, which is a vector of the histogram of oriented gradients in a neighborhood around the keypoint.

To compare two images (I_q and I_d from a set of images $\mathcal{I}$), for every descriptor ($\boldsymbol{d}_q$) in the first image (I_q), Hotspotter uses the Euclidean distance to find the two nearest neighbor descriptors ($\boldsymbol{d}_{\text{nn}}$ and $\boldsymbol{d}_{\text{nn2}}$) from all other images in the dataset. Then, Hotspotter calculates the similarity score, denoted $\delta(\boldsymbol{d}_q)$, for the descriptor $\boldsymbol{d}_q$ with the two nearest neighbor descriptors using

$$\delta(\boldsymbol{d}_q) = \left\|\boldsymbol{d}_{\text{nn2}} - \boldsymbol{d}_q\right\|^2 - \left\|\boldsymbol{d}_{\text{nn}} - \boldsymbol{d}_q\right\|^2, \tag{1}$$

where $\boldsymbol{d}_{\text{nn}}, \boldsymbol{d}_{\text{nn2}} \in (\mathcal{D}_{\text{all}} - \mathcal{D}_q)$. The variables $\mathcal{D}_{\text{all}}$ and $\mathcal{D}_q$ are sets of descriptors from all images in the dataset $\mathcal{I}$ and the image I_q, respectively. Finally, a similarity score, denoted $\Delta(I_q, I_d)$, between the two images I_q and I_d is calculated by summing the similarity scores for all qualified descriptor pairs. A qualified descriptor pair is defined as a descriptor pair where $\boldsymbol{d}_{\text{nn}}$ is from the second image I_d. That is,

$$\Delta(I_q, I_d) = \sum_{\boldsymbol{d}_q \in \mathcal{D}_q} \delta(\boldsymbol{d}_q), \text{ only if } \boldsymbol{d}_{\text{nn}} \in \mathcal{D}_d, \tag{2}$$

where $\mathcal{D}_d$ is a set of descriptors from the image I_d.

C. Background on Clustering

Our algorithm, presented in Section III, builds on the classic k-means clustering method [22]. The baseline clustering technique, against which we evaluate our approach, is presented

in Algorithm 1 and referred to as k-medoids++. It consists of k-medoids clustering [23] combined with the initialization used in k-means++ [24].

The input to a basic k-means algorithm includes the desired number of clusters, k, and a set of data points with known locations to be clustered. In our case, the points are images in the set $\mathcal{I}$ with unknown positions but known distances between each other, so that there is no easy way to compute a cluster mean. This is why we use k-medoids clustering instead, where a cluster medoid, denoted M_i, is defined as the data point I_x that has the minimum within-cluster variation in a given cluster, denoted C_i, i.e.,

$$M_i = \arg\min_{I_x \in C_i} \left(\sum_{I_y \in C_i} \left\| I_x, I_y \right\|^2 \right), \tag{3}$$

where I_y is a data point in the cluster C_i, and $\left\| I_x - I_y \right\|^2$ is the squared Euclidean distance between two data points I_x and I_y. In our work, where I_x and I_y are images, we use the similarity score given by (2) instead of the Euclidean distance.

We modify the k-medoids algorithm to add an initialization step that is analogous to that used in k-means++. Let the set of cluster medoids be denoted $\mathcal{M} = \{M_1, \ldots, M_i, \ldots, M_k\}$, where M_i is the i^{th} initial cluster medoid. The first initial medoid M_1 is chosen randomly with uniform probability. That is, the probability P_1 of any data point I_x being chosen as M_1 is

$$P_1 = \frac{1}{N}. \tag{4}$$

After which, each subsequent initial medoid M_i is randomly picked from the remaining data points with the probability proportional to the squared distance between a data point and the nearest medoid in $\{M_1, \ldots, M_{i-1}\}$. That is, the probability P_i of a data point I_x being chosen as M_i is

$$P_i = \frac{\left\| I_x, M_{o_x} \right\|^2}{\sum_{I_y \in \mathcal{I}} \left\| I_y, M_{o_y} \right\|^2}, \tag{5}$$

where the variables M_{o_x} and M_{o_y} are the nearest medoids in $\{M_1, \ldots, M_{i-1}\}$ for data points I_x and I_y, respectively.

Algorithm 1 A k-Medoids++ Clustering Algorithm

Given: Distance matrix of a set of data points $\mathcal{I}$
Number of clusters, k
Result: Clustering, $\mathcal{C} = \{C_1, \cdots, C_i, \cdots, C_k\}$

```
/*k-means++ initialization*/
Initialize a set of clustering medoids, M ← { }
Choose M_1 randomly from I using (4)
M ← M ∪ M_1, C_1 ← M_1
for i = 2 : k do
    Choose M_i randomly from (I − M) using (5)
    M ← M ∪ M_i, C_i ← M_i
/*k-medoids clustering*/
repeat
    for each I_x ∈ (I − M) do
        Determine the cluster with the nearest medoid, C_o
        C_o ← C_o ∪ I_x
    for each C_i ∈ C do
        Select M_i using (3)
until M does not change;
```

D. Background on Silhouette Score

The silhouette score is frequently used to evaluate the quality of a given clustering when dataset labels are not available. In Section III, we develop an extended silhouette score for our proposed algorithm to evaluate and improve the clustering performance. Here we review the standard definition.

To calculate the silhouette score of an image, the mean similarity score of the image with any cluster in a given clustering is computed first. Assume an image I_x is in an image dataset $\mathcal{I}$, and a cluster C_i is within a clustering $\mathcal{C}$ for the set $\mathcal{I}$. The mean similarity score of the image I_x with the cluster C_i, denoted $\bar{\Delta}_i$, is

$$\bar{\Delta}_i = \frac{1}{|C_i|} \left(\sum_{I_y \in C_i} \Delta(I_x, I_y)^2 \right), \tag{6}$$

for $I_x \notin C_i$. If $I_x \in C_i$ then I_x is left out of the mean calculation. Unfortunately, (6) is not defined if $|C_i| = 1$, and so the silhouette score is not applicable to single-item clusters.

The silhouette score measures the degree to which an item belongs to the cluster it has been assigned by comparing the difference between the mean similarity scores for its cluster with the nearest neighbor cluster [25]. The nearest neighbor cluster for an image I_x is defined as a cluster with the largest value of $\bar{\Delta}_i$, other than its own cluster. That is, the silhouette score of the image I_x, denoted s_x, is given by

$$s_x = \frac{\bar{\Delta}_o - \bar{\Delta}_{\text{nn}}}{\max\left\{\bar{\Delta}_o, \bar{\Delta}_{\text{nn}}\right\} + \epsilon}, \tag{7}$$

where $\bar{\Delta}_o$ and $\bar{\Delta}_{\text{nn}}$ are the mean similarity scores of the image I_x with its cluster C_o and its nearest neighbor cluster C_{nn}, respectively, and ϵ is required to avoid division by zero. Thus, the silhouette score is in the range $[-1, 1]$. A value of 1 indicates maximum confidence that the image belongs to its cluster while -1 indicates maximum confidence that it does not. Finally, the mean silhouette score over all images within a clustering $\mathcal{C}$, denoted $\bar{s}$, can be used for estimating the overall quality of the clustering, i.e.,

$$\bar{s} = \frac{1}{|\mathcal{I}|} \left(\sum_{I_x \in \mathcal{I}} s_x \right). \tag{8}$$

It is important to note that because the silhouette score is not defined for a single-item cluster, 0 is typically used as its silhouette score. This indicates that there is no confidence in whether the item should be by itself or included in a different cluster [25]. Unfortunately, it is very common for African leopard image datasets, resulting from camera traps, to include many individuals that are represented by only a single image. This means that using the standard silhouette score to evaluate clustering of such datasets will be misleading. To address this issue, we propose a novel method to calculate the silhouette score of an image in a single-item cluster in Section III.

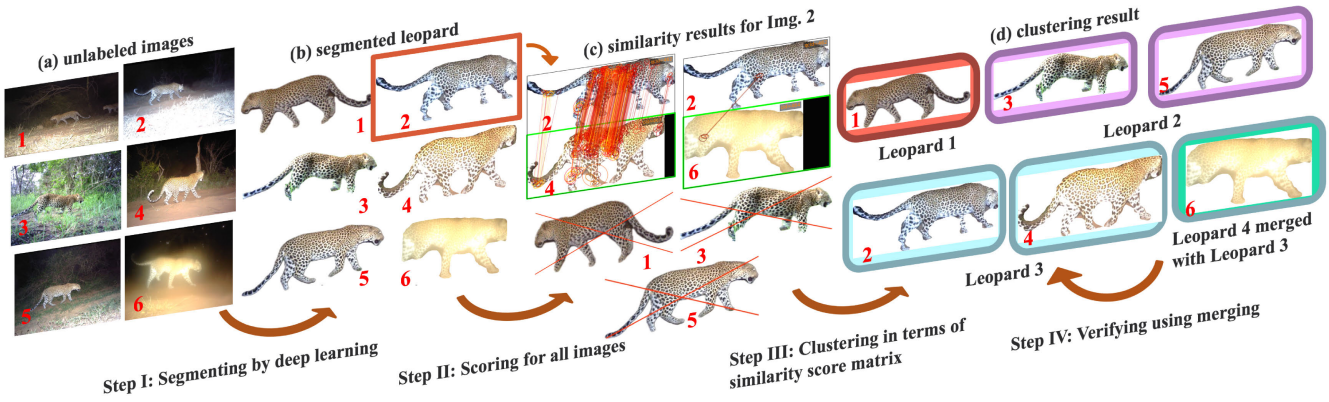

Fig. 1. This figure illustrates the flowchart of our automated leopard individual identification algorithm for a small subset of our image dataset. It contains four steps: segmenting by deep learning, scoring for all image pairs, clustering in terms of the similarity score matrix, and verifying a clustering by merging. In (a) six representative unlabeled original images, numbered Images 1 to 6, are shown. Applying segmentation to these images results in (b). After segmentation, the similarity scores between all images are computed. An example of this for Image 2 is displayed in (c). A line between two images indicates a qualified descriptor pair between the images. For example, Images 2 and 4 have a high similarity score because there are a large number of lines between them. Images with no qualified descriptor pairs are indicated with an "X", i.e., Images 1, 3, and 5, and result in a similarity score of 0. The algorithm then uses these similarity scores to perform clustering, the results of which are shown in (d). The internal colors around the leopard images represent the clusters assigned after initial clustering. Unfortunately, clusterings will typically have errors, so the algorithm applies a post-clustering verification step. The results of this step are shown in (d) using the external colors around the images to indicate the final clusters. That is, the final clustering results in 3 clusters, i.e., Leopard 1 containing Image 1, Leopard 2 including Images 3 and 5, and Leopard 3 having Images 2, 4, and 6.

III. Automated Individual Leopard Identification

A. Overview of the Automated Identification Algorithm

In this article, we design an effective algorithm to automatically identify animal individuals in an unlabeled African leopard image dataset.[1] The goal is to label all images' identities and determine the number of individuals in the dataset. Our algorithm consists of four parts: segmenting leopard objects from the background using deep learning, scoring similarity for image pairs using Hotspotter, clustering in terms of an $N \times N$ similarity score matrix, and verifying the initial clustering to generate an improved clustering. A flowchart is shown in Fig. 1. More specifically, we first use a standard mask region-based CNN (mask R-CNN) [26] model to segment leopard objects from the background in images. Then, we apply Hotspotter [12] to generate the $N \times N$ similarity score matrix for all images in the dataset, as described in the previous section. Next, we design a novel adaptive k-medoids++ clustering algorithm that uses silhouette score feedback to generate a clustering. In addition, we propose a new definition for the silhouette score of a single-item cluster to better represent the quality of a clustering. This allows our algorithm to ultimately converge to a more accurate clustering. Finally, we develop a post-clustering verification procedure to deal with the inherent limitations of medoid-based clustering approaches and generate an improved result.

In the following subsections, we will elaborate on these contributions.

B. Expanded Silhouette Method

Before discussing our new clustering algorithm, we first describe our expanded definition of the silhouette score that is used throughout our algorithm. As previously discussed,

[1]Code has been made available at: https://github.com/obaiga/Automatic-individual-animal-identification

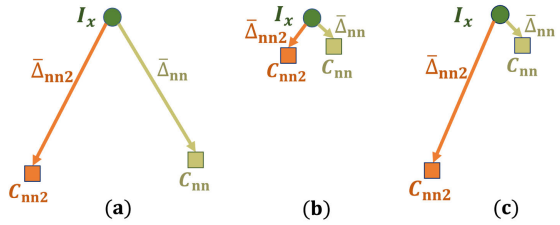

Fig. 2. This figure shows three typical relationships for an isolated image with its top-2 nearest neighbor clusters based on their mean similarity scores. A solid circle is used to represent an isolated image I_x and different colored squares correspond to its nearest and second nearest neighbor clusters, C_{nn} and C_{nn2}, respectively. The length of an arrow is proportional to the reciprocal of the mean similarity score of the image with a cluster, labeled $\bar{\Delta}$. In both (a) and (b), the value of s'_x is near 0 because the difference between $\bar{\Delta}_{nn}$ and $\bar{\Delta}_{nn2}$ is small, however, they represent very different cases. In particular, the image in (a) should stay isolated whereas the image in (b) may belong to either cluster. In case (c), s'_x is large and it is obvious that the isolated image should join its nearest neighbor cluster.

the standard definition of the silhouette score provides no information for single-image clusters, denoted C_{iso}, referred to as *isolated images*, which are common in our dataset. To address this issue, we present a novel approach to generate a reasonable value for the confidence that an image belongs to its single-image cluster. To do this, we first compute a value, denoted s'_x, using (7) as if an isolated image I_x is included in its nearest neighbor cluster C_{nn}, i.e.,

$$s'_x = \frac{\bar{\Delta}_{nn} - \bar{\Delta}_{nn2}}{\bar{\Delta}_{nn} + \epsilon}, \tag{9}$$

where $I_x \in C_{iso}$ and $\bar{\Delta}_{nn2}$ is the mean similarity score of the image with its second nearest neighbor cluster C_{nn2}.

It is important to note that s'_x is always in the range $[0, 1]$. This is true because for any image, the mean similarity score with its second nearest neighbor cluster $\bar{\Delta}_{nn2}$ can not be greater than the mean similarity score with its nearest neighbor cluster $\bar{\Delta}_{nn}$. Fig. 2 illustrates three typical cases that occur. In Fig. 2(a), it is clear that I_x should stay as an isolated cluster

and so one would like its silhouette score to reflect this, i.e., $s_x = 1$. In Fig. 2(b), it is not clear whether I_x should stay as an isolated cluster, or be merged with one of its nearby neighbors, therefore one would like a silhouette score value of $s_x = 0$. However, in both of these cases the value of $s'_x \approx 0$ and thus we cannot distinguish between these two situations. Therefore, we set $s_x = 0.5$ whenever s'_x is small, which our adaptive k-medoids++ clustering algorithm will use to distinguish between these two cases as discussed in the next subsection. In contrast, the case shown in Fig. 2(c) is unambiguous. It results in a "large" value of s'_x indicating a strong confidence that the image belongs to its nearest neighbor cluster, and thus s_x should be a large negative value. We describe the relationship between s'_x and s_x mathematically as

$$s_x = \begin{cases} -s'_x & \text{if } s'_x \geq T \text{ and } I_{\text{nn}} \in C_{\text{nn}} \\ 0.5 & \text{otherwise,} \end{cases} \tag{10}$$

where $I_x \in C_{\text{iso}}$, I_{nn} is its nearest neighbor image, and we define "large" using the following threshold

$$T = \frac{1}{\sum_{C_i \subset \mathcal{C}} |C_i|} \left(\sum_{C_i \subset \mathcal{C}} \sum_{I_y \in C_i} s_y \right) \quad \text{if } I_y \in C_{\text{niso}}, \tag{11}$$

where C_{niso} is a cluster having more than one image. The value of the threshold is the average silhouette score for all non-isolated images. Intuitively, we only want to accept isolated images into a cluster when it will make the overall clustering better. In addition, we add the condition $I_{\text{nn}} \in C_{\text{nn}}$ because the value of s'_x may be large due to $\bar{\Delta}_{\text{nn2}} \approx 0$ regardless of the value of $\bar{\Delta}_{\text{nn}}$. By including isolated images in our expanded definition of the silhouette score, we obtain a much more accurate measure for the quality of a clustering.

Next we propose a measure of the quality of a cluster C_i, denoted $\bar{s}_i$, that is the mean silhouette score of all images in the cluster, i.e.,

$$\bar{s}_i = \frac{1}{|C_i|} \left(\sum_{I_x \in C_i} s_x \right). \tag{12}$$

It is used as a feedback parameter in our novel adaptive k-medoids++ clustering algorithm that is discussed in the next subsection.

C. Adaptive k-Medoids++ Clustering

In general, the baseline k-medoids++ clustering algorithm, described in section II, must be executed multiple times to obtain a clustering that is satisfactory. Because the algorithm is always started with the same probability distribution, there is no learning from one execution to the next. To increase the likelihood that subsequent executions find better clusterings, we propose a technique to adaptively adjust the probability of an image being selected as a medoid seed. In particular, if a current cluster medoid results in a "good" cluster, its probability of being selected as a medoid in future iterations is increased. Likewise, if a medoid results in a "bad" cluster, its probability of being selected as a medoid in future iterations is decreased. We quantify what is considered a "good" or "bad" cluster using the value of the cluster's mean silhouette score. We define the current best clustering, denoted $\mathcal{C}^*$, as the clustering with the highest mean silhouette score, denoted $\bar{s}^*$. Our algorithm terminates when $\bar{s}^*$ does not improve for a specified number of iterations. The pseudocode is shown in Algorithm 2.

Algorithm 2 Adaptive k-Medoids++ Clustering

Given : Similarity score matrix for an image dataset $\mathcal{I}$
Number of clusters, k
Result: Best clustering, $\mathcal{C}^*(k)$

```
Best mean silhouette score of a clustering, s̄*(k) ← 0
Iteration, j ← 0
Weight factors, W(j) = {w_1(j), ··· , w_|I|(j)} ← {1, ··· , 1}
repeat
    j ← j + 1
    Generate M and C using Algorithm 1 with (13), (14), (16)
    Calculate mean silhouette score s̄ of a clustering C
    if s̄ > s̄*(k) then
        for each I_x ∈ I do
            if I_x isanyclustermedoidM_i then
                w_x(j) ← w_x(j − 1) · (s̄_i + 1)
            else
                w_x(j) ← w_x(j − 1)
        C*(k) ← C, s̄*(k) ← s̄
    else
        W(j) ← W(j − 1)
until C*(k) does not change for a specific iteration number;
```

To calculate the probability that an image will be chosen as an initial cluster medoid at the next iteration, denoted $(j+1)$, we introduce a weight factor for the image in (4) and (5), i.e., the probability $P_1(j+1)$ that an image I_x is selected as the first medoid M_1 is given by

$$P_1(j+1) = \frac{w_x(j)}{\sum_{I_y \in \mathcal{I}} w_y(j)}, \tag{13}$$

where $w_x(j)$ and $w_y(j)$ are the weight factors for the images I_x and I_y at iteration j, respectively. After which, the probability $P_i(j+1)$ of image I_x being chosen as a subsequent cluster medoid seed M_i is given by

$$P_i(j+1) = \frac{\frac{1}{\Delta(I_x, M_{ox})^2 + \epsilon} \cdot w_x(j)}{\sum_{I_y \in \mathcal{I}} \left(\frac{1}{\Delta(I_y, M_{oy})^2 + \epsilon} \cdot w_y(j) \right)}. \tag{14}$$

We make the probability inversely proportional to the similarity score in order to make it more likely that the medoids are well distributed.

For an image I_x that is the medoid for a cluster C_i, its weight factor is based on the quality of C_i in the current best clustering. Specifically, the weight factor is computed as the mean silhouette score of the cluster $\bar{s}_i$ plus 1. (This keeps its value positive because the range of the silhouette score is $[-1, 1]$.) For example, if $\bar{s}_i$ is -1, the image will not be selected as a cluster medoid. When $\bar{s}_i$ is 1, the weight factor increases the probability of the image being chosen as a cluster medoid at the next iteration by a factor of two.

If an iteration results in a new best clustering, the weights of images serving as medoids are adjusted to reflect their cluster quality. Images that do not serve as medoids retain their weight factor from the previous iteration. That is, for a current best

clustering $\mathcal{C}^*$ of the dataset $\mathcal{I}$, at an iteration j, the weight factor for an image I_x is given by

$$w_x(j) = \begin{cases} w_x(j-1) \cdot \left(\bar{s}_i + 1\right) & \text{if } I_x = M_i, \quad M_i \in \mathcal{M}^* \\ w_x(j-1) & \text{otherwise,} \end{cases} \tag{15}$$

where $\mathcal{M}^*$ is the set of cluster medoids for the current best clustering $\mathcal{C}^*$, M_i is the medoid for a cluster C_i, and $w_x(0) = 1$. The adjustment of the weight factors in this manner also addresses the issue presented previously in Fig. 2(a) and (b), i.e., how to determine if isolated medoids should be merged or stay isolated. By setting $\bar{s}_i$ to 0.5, it becomes more likely for these medoids to become seed medoids. Once this occurs, the medoid in Fig. 2(a) will remain isolated, whereas the medoid in Fig. 2(b) will be clustered with its nearest neighbors.

It is important to note that our adaptive k-medoids++ clustering algorithm applies to the similarity scores of images. That is, instead of selecting a medoid with the smallest distance to its cluster members as in (3), we use the image with the maximum similarity score among all the cluster members, i.e.,

$$M_i = \arg\max_{I_x \in C_i} \left(\sum_{I_y \in C_i} \Delta(I_x, I_y)^2 \right). \tag{16}$$

Because the similarity score of an image with itself is infinity, we define it to be a large value given by the squared sum of its similarity scores with all other images in the dataset $\mathcal{I}$, i.e.,

$$\Delta(I_x, I_x) = \left(\sum_{I_y \in \mathcal{I}, I_y \neq I_x} \Delta(I_x, I_y) \right)^2. \tag{17}$$

D. Determining the Best Clustering

Our adaptive k-medoids++ algorithm, given a cluster number k, uses the mean silhouette score of a clustering to find the best clustering $\mathcal{C}^*(k)$, i.e., a clustering with the highest mean silhouette score, $\bar{s}^*(k)$. Ideally, a k closer to the actual class number results in an $\bar{s}^*(k)$ closer to 1, which corresponds to a more accurate clustering. In contrast, a k farther from the actual class number typically results in a lower value of $\bar{s}^*(k)$. This correlation motivates us to use a modified ternary search [27] to identify the best k^* that results in the best clustering $\mathcal{C}^*$, in a computationally efficient manner as compared to an exhaustive search. The pseudocode is shown in Algorithm 3.

Specifically, the ternary search splits a given range of k into three equal sub-intervals. It then retains the range that includes the k with a better $\bar{s}^*(k)$ and repeats the process until the range cannot be split. Because Algorithm 2 is probabilistic, different executions with the same value of k may result in different best clusterings, so that only the best clustering with the highest value of $\bar{s}^*(k)$ is retained, shown in Line 14 – 18.

Like all k-medoids clustering algorithms, the initial choice of cluster medoids may prevent clusters from being combined. This is an inherent problem of all such algorithms and we present a post-clustering verification procedure to address this issue in the next subsection.

Algorithm 3 Determine the Best Clustering

Given : Similarity score matrix for an image dataset $\mathcal{I}$
Result: Best number of clusters, k^*
Best clustering, $\mathcal{C}^*$

1 /*Ternary search*/
2 Best mean silhouette score of a clustering, $\bar{s}^* \leftarrow 0$
3 Left and right thresholds for k values, $k_l \leftarrow 1$, $k_r \leftarrow |\mathcal{I}|$
4 **repeat**
5 Divide k range into thirds, $k_{m1} \leftarrow k_l + \text{round}(\frac{k_r - k_l}{3})$, $k_{m2} \leftarrow k_r - \text{round}(\frac{k_r - k_l}{3})$
6 Generate $\mathcal{C}^*_{m1}$ and $\bar{s}^*_{m1}$ using Algorithm 2 with k_{m1}
7 Generate $\mathcal{C}^*_{m2}$ and $\bar{s}^*_{m2}$ using Algorithm 2 with k_{m2}
8 **if** $\bar{s}^*_{m1} < \bar{s}^*_{m2}$ **then**
9 $k_l \leftarrow k_{m1}$
10 **else if** $\bar{s}^*_{m1} > \bar{s}^*_{m2}$ **then**
11 $k_r \leftarrow k_{m2}$
12 **else**
13 $k_l \leftarrow k_{m1}$, $k_r \leftarrow k_{m2}$
14 /*Retain current best clustering with highest $\bar{s}^*$ */
15 **if** $\bar{s}^*_{m1} > \bar{s}^*$ **then**
16 $k^* \leftarrow k_{m1}$, $\mathcal{C}^* \leftarrow \mathcal{C}^*_{m1}$, $\bar{s}^* \leftarrow \bar{s}^*_{m1}$
17 **if** $\bar{s}^*_{m2} > \bar{s}^*$ **then**
18 $k^* \leftarrow k_{m2}$, $\mathcal{C}^* \leftarrow \mathcal{C}^*_{m2}$, $\bar{s}^* \leftarrow \bar{s}^*_{m2}$
19 **until** $k_r - k_l \leq 1$;

E. Post-Clustering Verification

Our novel post-clustering verification procedure addresses two issues that are inherent limitations of medoid-based clustering algorithms that typically occur when there is a mismatch between the parameter k and the number of true clusters. The first issue is that there may be multiple clusters that should be a single cluster, which occurs when k is too large. This is due to the algorithm randomly choosing multiple images from the same true cluster as medoid seeds. The second issue is images that are assigned to clusters where they do not belong, referred to as *outlier images*, which occurs when k is too small. It is due to clustering algorithms being forced to assign any non-medoid image to a cluster, resulting in images that are far from their assigned clusters. Therefore, the goal of our post-clustering verification procedure is to recognize when these two issues occur and then potentially merge similar clusters and reassign outlier images. To achieve this goal, we employ the silhouette score, which reflects both the confidence to which an image belongs to its own cluster and the qualities of individual clusters in a clustering.

First, given the best clustering $\mathcal{C}^*$, we determine similar clusters that should be merged. Recall that for an image I_x, the nearest neighbor image is the most similar image outside its own cluster, while the nearest neighbor cluster is the next best-fitting cluster other than its own. Analogously, for a cluster C_i, a *companion cluster*, if one exists, satisfies two properties. It is the nearest neighbor cluster for the majority of the images in C_i. It is also the cluster that contains the nearest neighbor images for the majority of the images in C_i. A cluster C_i is merged with its companion cluster C_j if the resulting mean silhouette score $\bar{s}_{ij}$ of the merged cluster C_{ij} is better than the average silhouette score before merging, i.e.,

$$C_{ij} = C_i \cup C_j$$

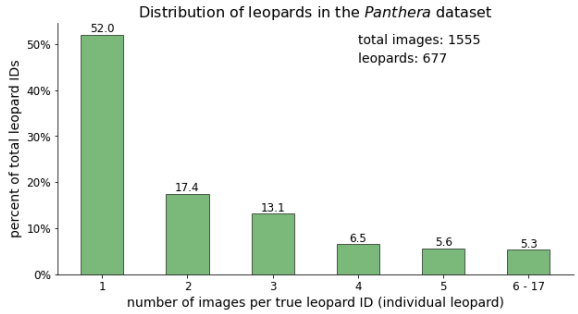

Fig. 3. The number of images of an individual leopard in the *Panthera* dataset varies significantly, as illustrated in this distribution. Over half of the leopards are represented by only a single image, which complicates the animal ID problem.

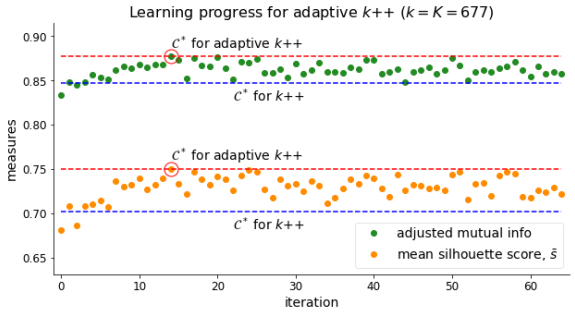

Fig. 4. This figure plots the adjusted mutual information score (regarded as accuracy) and mean silhouette score of a clustering $\bar{s}$ during the learning process from a typical execution of adaptive k++. These values are compared to those for the best clustering $\mathcal{C}^*$ using the baseline k++, shown in the blue dashed line, while the best clustering C^* using adaptive k++ is shown in the red dashed line. The accuracy for adaptive k++ quickly increases within the first 15 iterations to obtain much better clusterings than the best from k++. This is due to updating the selection probabilities of medoid seeds for adaptive k++. In addition, this figure indicates that the mean silhouette score $\bar{s}$ is a useful estimate for the quality of a clustering.

$$\text{if} \quad \bar{s}_{ij} > \frac{1}{|C_i| + |C_j|}\left(\sum_{I_x \in C_i} s_x + \sum_{I_y \in C_j} s_y\right)$$
$$\text{and} \sum_{I_x \in C_i} [\![C_{\text{nn}} = C_j]\!] > \frac{|C_i|}{2}$$
$$\text{and} \sum_{I_x \in C_i} [\![I_{\text{nn}} \in C_j]\!] > \frac{|C_i|}{2}, \tag{18}$$

where $[\![Q]\!]$ is the Iverson bracket function, i.e., $[\![Q]\!] = 1$ if the statement Q is true, otherwise $[\![Q]\!] = 0$. All clusters, except for isolated images, are considered for merging. The best clustering $\mathcal{C}^*$ is updated after merging all companion clusters.

Next, we need to determine whether an image is an outlier. Recall that an image with a negative silhouette score may not belong to its currently assigned cluster. This is our definition of an outlier image. We then determine whether an outlier image should be reassigned to its nearest neighbor cluster or become an isolated image, i.e., a cluster by itself. For this, we compute a value s'_x for an outlier image I_x using

$$s'_x = \frac{\bar{\Delta}_{\text{nn}} - \max(\bar{\Delta}_{\text{nn2}}, \bar{\Delta}_o)}{\bar{\Delta}_{\text{nn}}}, \tag{19}$$

where $I_x \in C_{\text{niso}}$ and $s_x < 0$. If an outlier image is an isolated image, s'_x is calculated using (9). The decision for determining whether an outlier image is moved to its nearest neighbor cluster or isolated is given by

$$I_x \in \begin{cases} C_{\text{nn}} & \text{if } s'_x \geq T \text{ and } I_{\text{nn}} \in C_{\text{nn}} \\ C_{\text{iso}} & \text{otherwise,} \end{cases} \tag{20}$$

where $s_x < 0$, T is calculated using (11), and I_{nn} is the nearest neighbor image outside of C_o with the highest similarity score for the image I_x. After evaluating all outlier images, the entire post-clustering verification procedure is repeatedly executed to improve the best clustering $\mathcal{C}^*$ and the corresponding cluster number k^* until the best mean silhouette score of the clustering $\bar{s}^*$ cannot be improved.

IV. Results

Our automated individual leopard identification algorithm is tested on a set of camera trap images of African leopards provided by *Panthera* [11], an organization dedicated to the conservation of wild cats. The images were collected over several years from multiple camera traps placed throughout the leopards' habitat. The dataset contains 1555 images (N) of 677 leopards (K), with the distribution of individual leopards shown in Fig. 3. *Panthera* scientists provided the ground truth by manually clustering the 1555 images into 677 animal IDs corresponding to the right or left side of an individual animal. (Leopards have distinct spot patterns on each side of the body.) Several reasons cause the ratio of K/N to be relatively large in this small dataset. These reasons include the seclusive behavior of African leopards, their wide-ranging habitat, and the low density of camera traps. In addition, the camera traps only take one image when activated.

The first step of our automated identification algorithm employs the open-source Python implementation of mask R-CNN [26] to segment leopard bodies in images and then uses the Hotspotter application [12] to compute similarity scores for all segmented image pairs. Next, the algorithm executes our novel adaptive k-medoids++ clustering technique that uses a modified ternary search [27] to determine the best clustering $\mathcal{C}^*$ and the corresponding cluster number k^*. Finally, the algorithm implements our new post-clustering verification procedure to further improve the clustering accuracy and update $\mathcal{C}^*$ and k^*.

We first illustrate the improvement in the performance of our adaptive k-medoids++ clustering technique (referred to as adaptive k++) when compared to the traditional k-medoids++ clustering algorithm (referred to as k++) that we use as a baseline.[2] The execution of adaptive k++ is terminated after 50 consecutive iterations without improvement in the best clustering $\mathcal{C}^*$, defined as the clustering with the highest mean silhouette score $\bar{s}^*$, or after 200 iterations, whichever comes first. To gain insight into how adaptive k++ arrives at better solutions, in Fig. 4 we present the learning curve, i.e., accuracy as a function of iterations, from a typical execution using the true number of leopards in the dataset, $k = 677$. The accuracy value is measured by the adjusted mutual information score [29], which is used to evaluate the

[2]We also compared our approach to other unsupervised techniques, e.g., spectral clustering [28], with comparable results in Appendix.

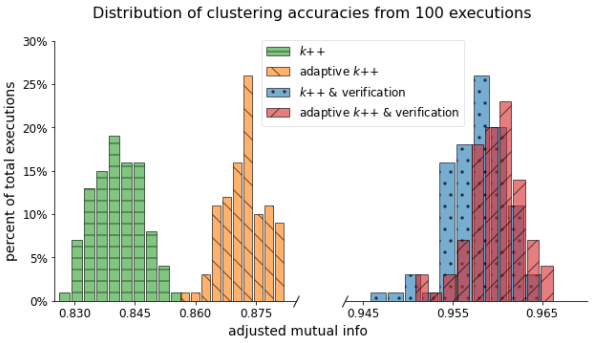

Fig. 5. This figure shows the distribution of the clustering accuracies for the best clusterings from 100 random executions of adaptive k++ and k++. Both approaches were executed with $k = K = 677$. The baseline k++ has an average accuracy of 0.841. By updating the selection probabilities of medoid seeds, adaptive k++ is able to improve the accuracy, on average, by 3.7% to 0.872. Or, using verification directly on k++ improves the accuracy, on average, by 13.8% from 0.841 to 0.957. When combining adaptive k++ with verification, the average accuracy reaches 0.960, i.e., a 14.1% improvement over the baseline k++ value of 0.841. Note that after verification, the cluster number k typically changes depending on the number of reassigned outlier images and merged companion clusters.

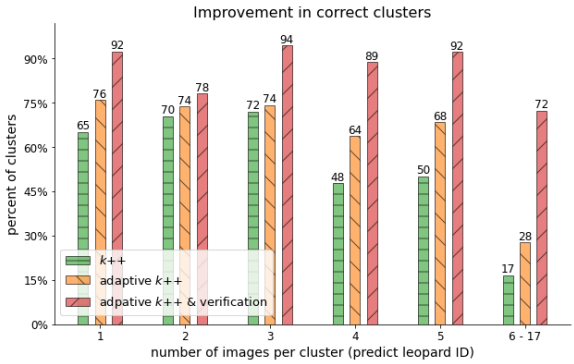

Fig. 6. This figure shows correct cluster percentages as a function of cluster sizes in the best clustering for adaptive k++ and the baseline k++. The best clustering is from a typical execution with $k = K = 677$. On average, adaptive k++ identifies more correct clusters (71.5%) than the baseline k++ (62.3%). This is because adaptive k++ updates the selection probabilities of cluster medoid seeds in terms of their individual cluster qualities. When followed by verification, adaptive k++ identifies 88.5% of the total number of true clusters. Verification is particularly effective at improving performance for large clusters, increasing the number of correct clusters from 17% to 72% for clusters containing more than 5 images. This is due to the verification procedure repeatedly merging companion clusters and identifying isolated images until the clustering cannot be improved.

agreement between a generated clustering and the ground truth clustering for the dataset. We also show the curve of the mean silhouette score $\bar{s}$ to illustrate its effectiveness as a proxy for accuracy. In this case, our best clustering accuracy is 0.031 higher than that of the baseline k++ and it occurs after only 15 iterations. On average, the best clustering is identified in 50 iterations so that the algorithm terminates after 100 iterations. Because adaptive k++ adjusts the selection probabilities of medoid seeds, better medoids are more likely to be selected in subsequent iterations and there is rapid improvement after only a few iterations.

We now compare the average accuracy of adaptive k++ with the baseline k++ over 100 executions. Because adaptive k++ performs a maximum of 200 iterations, we compare the result of one execution of it to the best clustering $\mathcal{C}^*$ obtained from 200 random executions of k++. (In other words, 20,000 executions of k++ are compared to 100 executions of adaptive k++.) The results of this comparison are illustrated in Fig. 5 where we show the distribution of clustering accuracy for adaptive k++. Both approaches were executed with $k = K = 677$. A comparison of the two distributions indicates that our adaptive k++'s average accuracy is 0.032 higher than that of k++. We next explore the additional improvement that can be obtained from our post-clustering verification procedure.

We apply verification to the best clusterings generated by adaptive k++ and the baseline k++ from the above 100 executions. The results are shown in Fig. 5 where we display the distributions of improved clustering accuracy. A comparison of the two distributions before and after verification for both clustering approaches demonstrates that verification effectively improves the average clustering accuracy by 0.088 and 0.116 for adaptive k++ and k++, respectively. It is important to note that after verification, the cluster number k typically changes due to reassigning outlier images and merging companion clusters. On average, the updated cluster number k for adaptive k++ and k++ changes to 700 and 694, respectively. Several reasons cause an updated cluster number k to be typically greater than the actual leopard number K. These reasons include identifying low-quality images as isolated images and separating the images of a single animal into multiple clusters.

To gain further insight into the effect of adaptive k++ and verification on improving the clustering performance, we investigated the proportion of *correct clusters* as a function of cluster size. We define a correct cluster as a cluster that is comprised of all of the images of a single leopard and no other leopard images. The results are shown in Fig. 6 where we compare the number of correct clusters in the best clusterings between adaptive k++ and the baseline k++ from a typical execution with $k = K = 677$. This figure shows that adaptive k++ is able to identify more correct clusters than the baseline k++ regardless of the cluster size. For the case of isolated images, adaptive k++ identifies 11% more correct clusters than the baseline k++. This is because adaptive k++ increases the probabilities of isolated images being selected as medoids by using our expanded definition of the silhouette score. For larger cluster sizes, adaptive k++ is able to reduce the probabilities of multiple images from the same true cluster being selected as cluster medoid seeds. However, this becomes increasingly more difficult for larger cluster sizes. For example, for clusters with more than 5 images, adaptive k++ results in a significant improvement in correct clusters, but it still represents only 28% of the true clusters with that cluster size. This issue is addressed by the verification procedure.

The impact of employing verification on the best clusterings is also shown in Fig. 6. A comparison of the correct cluster percentages before and after verification illustrates that verification is very effective in improving the number of correct clusters for all cluster sizes. We now discuss the mechanism by which the improvement is achieved, as a function of cluster size. For the case of isolated images, the combination of adaptive k++ with verification results in 92% correct clusters,

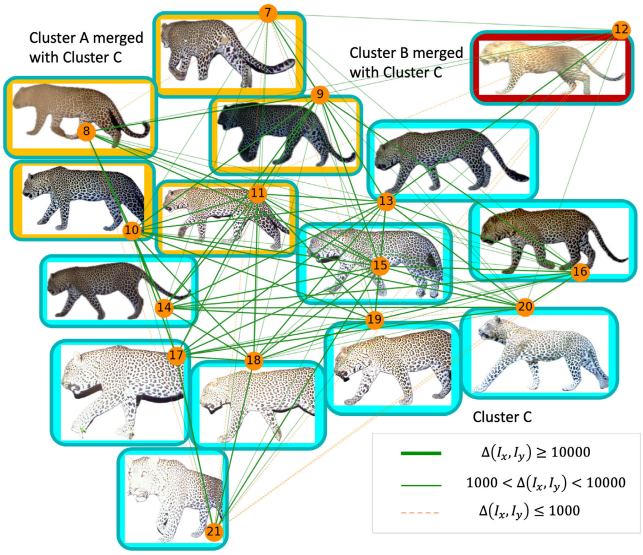

Fig. 7. This figure illustrates a case where the verification procedure merges all images from an individual leopard to a correct cluster, that initially were separated into multiple clusters by adaptive k++. The graph shows a leopard represented by 15 images (7 to 21). These images are initially clustered by adaptive k++ into three clusters (images in the same initial clusters have the same interior border colors), i.e., Cluster A includes Images 7 to 11, Cluster B has Image 12, and Cluster C contains Images 13 to 21, because Images 10, 12, and 15 were selected as cluster medoids. After verification, these images are merged into a single cluster, C, as the correct cluster (images in the same cluster have the same exterior border color).

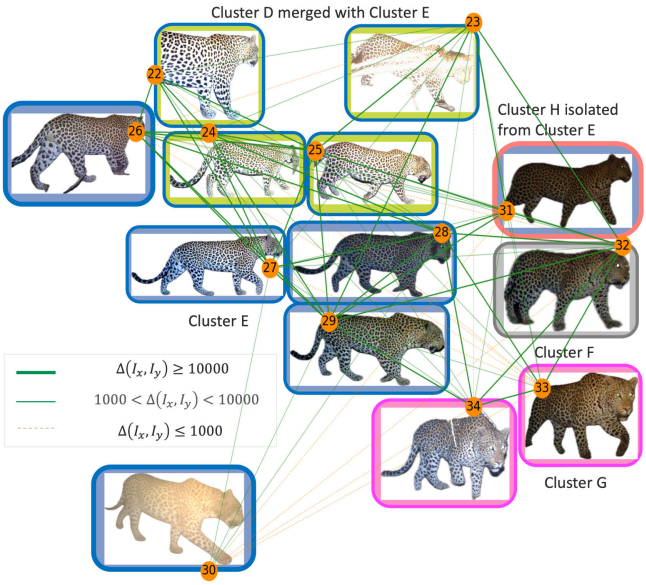

Fig. 8. This figure illustrates a case where the verification procedure is not able to merge all images of a leopard into a single cluster. The graph shows a leopard represented by 13 images (22 to 34), where images in the same initial clusters have the same interior border color. That is, Cluster D includes Images 22 to 25, Cluster E contains Images 26 to 31, Cluster F has isolated Image 32, and Cluster G involves Images 33 and 34. After verification, Cluster D is correctly merged with Cluster E, however, Images 31 to 34 are separated into three clusters, F, G, and H (images in the same clusters have the same exterior border colors). This occurs because these three clusters each have images that have high similarity scores with images from two other clusters. Therefore, verification could not determine which clusters should be merged.

compared to just 65% for the baseline k++. This is achieved by identifying isolated images that are incorrectly assigned to clusters as outlier images and reassigning them to new clusters by themselves. The impact of repeatedly merging companion clusters and identifying isolated images becomes greater as the cluster size increases. In particular, for clusters consisting of two images, there is a moderate increase in correct clusters, whereas for clusters greater than 5 images, there is a dramatic improvement over the baseline k++, so that 72% of the clusters are correctly identified. On average, the verification procedure is repeated 4 times until the clustering cannot be improved.

Next, we illustrate why large clusters are difficult for medoid-based clustering approaches to identify correctly and how verification can arrive at better solutions. Fig. 7 shows an individual leopard represented by 15 images (labeled 7 to 21), where the image layout is plotted by a force-directed graph drawing algorithm [30] and the edges represent the similarity scores between image pairs. Despite strong similarity scores, these images are initially clustered into three clusters (labeled A, B, and C), where images belonging to the same clusters have the same interior border colors. This occurred because Images 10, 12, and 15 were selected as cluster medoids by adaptive k++. This illustrates why it is difficult for adaptive k++ to identify large clusters, i.e., it is likely that they will contain multiple medoids. Fortunately, the verification procedure is able to merge these three clusters into a single cluster, i.e., after verification, Cluster B is a correct cluster, which is indicated by all images having the same exterior border color. Specifically, verification regards Cluster A as a companion for Cluster C and merges them together. Image 21 is identified as an outlier and reassigned to Cluster C. Currently, our novel verification procedure is very successful in merging clusters when they are similar and distinct from other clusters.

However, sometimes verification cannot correctly cluster all images from an individual leopard into a single cluster. One such case is illustrated in Fig. 8, which shows a leopard represented by 13 images (labeled 22 to 34). After verification, Cluster D is correctly merged with Cluster E, however, Images 31 to 34 are incorrectly separated into three clusters (labeled F, G, and H). This happens because Images 31 to 34 have high similarity scores with more than one cluster. For example, Image 31 becomes an isolated image (Cluster H) because verification cannot determine if it belongs to Cluster E or F. This is not surprising, because both Images 31 and 32 should belong to Cluster E. Likewise, Image 32 is similar to both Clusters E and G and so stays isolated (Cluster F). Cluster G is not merged with Cluster E, because even though Image 34 is very similar to Image 29, the nearest neighbor for Image 33 is Image 32.

Finally, we demonstrate the effectiveness of the modified ternary search that employs the mean silhouette score of a clustering $\bar{s}$ to identify the best clustering $\mathcal{C}^*$ and the corresponding cluster number k^*. Recall that Fig. 4 indicates $\bar{s}$ is a useful proxy to evaluate the performance of a clustering. To determine $\mathcal{C}^*$ and k^*, we present the ternary search curve, i.e., the best silhouette score $\bar{s}^*(k)$ as a function of the cluster number k. The result is illustrated in Fig. 9, where we also plot the accuracy of the resulting clusterings as measured by the adjusted mutual information score. The goal is to identify the best clustering C^* with the largest value of $\bar{s}^*$ and

TABLE I
ALGORITHM ACCURACY

Approach	Predicted Clusters (Images)						$K = 677$
	correct		partially correct		incorrect		predicted k
k++	554 (1001)		220 (476)		19 (78)		793
adaptive k++	566 (1093)	+	145 (363)	+	28 (99)	=	739
adaptive k++ & verification	**617 (1303)**		**91 (218)**		**10 (34)**		**718**

correct cluster: consists of ALL the images of a single leopard and no other leopard images
partially correct cluster: consists of SOME images from a single leopard and no other leopard images
incorrect cluster: NOT a correct or partially correct cluster
(images): total number of images in a corresponding cluster type

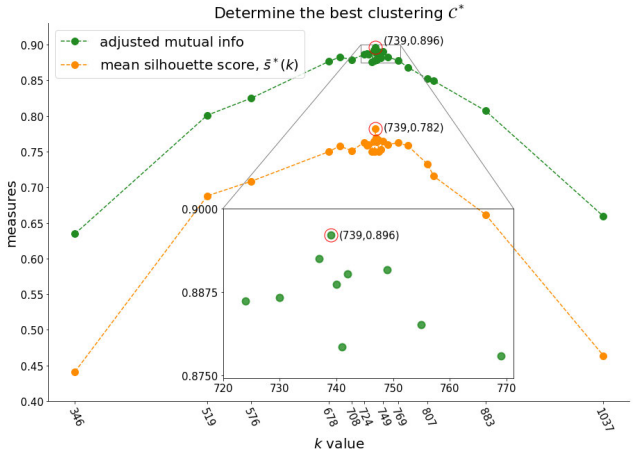

Fig. 9. This plot shows the results of the modified ternary search, where the adjusted mutual information score (regarded as accuracy) and best mean silhouette score $\bar{s}^*(k)$ as functions of the cluster number k. The modified ternary search identified the best clustering with the highest $\bar{s}^*(k)$ after executing adaptive k++ on the selected values of k. The accuracy of the best clustering $\mathcal{C}^*$ is 0.896 and the corresponding cluster number k^* is 739.

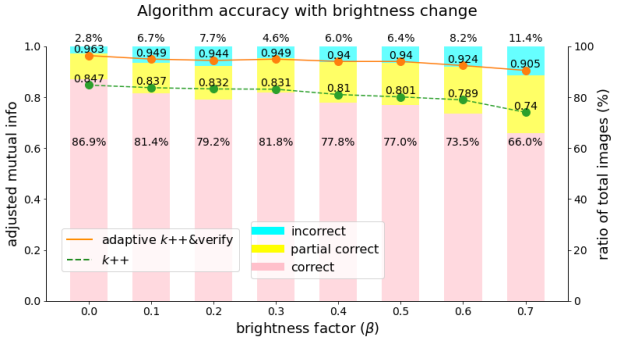

Fig. 10. This figure shows the accuracy (measured by adjusted mutual information score) for the best clusterings of the baseline k++ and adaptive k++ with verification under different values of brightness factor β. Both algorithms were executed with $k = K = 677$. When the illumination level changes up to 70%, our adaptive k++ with verification still gets a good accuracy, 0.905, compared to the baseline k++, 0.740. In addition, this figure displays the ratio of images in the three types of clusters (correct, partially correct, incorrect) in the best clusterings for adaptive k++ with verification under different values of β. When $\beta = 0.5$, our adaptive k++ achieves 77% images belonging to correct clusters while only 6.4% images are assigned to incorrect clusters.

its corresponding value of k^* in a computationally efficient manner. The final best clustering results in an $\bar{s}^* = 0.782$ and a $k^* = 739$ with a clustering accuracy of 0.896. After verification, the accuracy of the best clustering is improved to 0.958, and the updated cluster number becomes 718.

To illustrate another method for evaluating a clustering, we classify all clusters in a clustering into three different types, i.e., correct, *partially correct*, and *incorrect*. Recall that a correct cluster has all of the images from an individual leopard and no other leopard images. Similarly, we define a partially correct cluster as a cluster with some (but not all) of the images of a particular leopard and no images of any other leopards. If a cluster is neither a correct nor partially correct cluster, then it is defined as incorrect. The results of applying this evaluation method are shown in Table I where we compare the number of different cluster types in the best clustering $\mathcal{C}^*$ obtained from the baseline algorithm and our approach. Specifically, after verification, the best clustering for adaptive k++ obtains a significant improvement over the baseline algorithm, i.e., 85.9% correct clusters as compared to 69.8% for the baseline k++.

Table I also reflects the improvement in the number of partially correct clusters, reducing them from 27.7% to 12.7% of the total number of clusters. However, there are still 91 clusters that are only partially correct. This is because it is difficult for verification to decide which cluster an image belongs to if the image is similar to more than one cluster. Fig. 8 illustrates an example of this where verification separates images from an individual leopard into several partially correct clusters. In addition, in the best clustering, adaptive k++ followed by verification reduces the number of incorrect clusters to 1.4%. It is unlikely that one can reduce this number to zero, because these clusters contain images that are either of poor quality or only contain partial leopards.

We performed further experiments on the robustness of our algorithm to image quality, i.e., under- and over- exposure of images. Specifically, we randomly selected 1/3 of the dataset images to be underexposed, 1/3 to be overexposed, and 1/3 unchanged. For overexposed images, every pixel value is multiplied by $1+\beta$; for underexposed images, every pixel value is multiplied by $1-\beta$, where the brightness factor β is varied from 0.1 to 0.7. The results are shown in Fig. 10, where the performance of the baseline k++ and adaptive k++ with verification as a function of the brightness factor are displayed. In terms of the adjusted mutual information score, the accuracy of baseline k++ drops 12.6% from the original 0.847 to 0.740 when the brightness factor increases to 0.7. Our adaptive k++ with verification is more robust to illumination changes, and it still achieves 0.905 accuracy, only dropping 6.0% from 0.963. In addition, in terms of incorrectly identified images, our algorithm is able to maintain a reasonably small number of such images up to a 50% change ($\beta = 0.5$) in

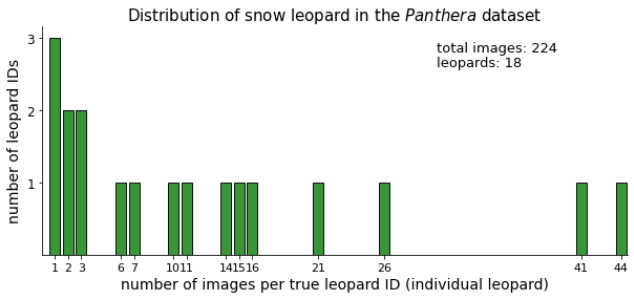

Fig. 11. This distribution illustrates the significant variation in the number of images of an individual snow leopard in the *Panthera* dataset. Two clusters have over 40 images each.

TABLE II
ALGORITHM ACCURACY

Approach	Images ($N = 224$, $K = 18$) correct	partially correct	incorrect
k++	10	205	9
adaptive k++ & verification	**23**	**201**	**0**

correct images: total number of images in correct clusters
partially correct images: total number of images in partially correct clusters
incorrect images: total number of images in incorrect clusters

the illumination levels for 2/3 of the images. One reason that adaptive k++ with verification prevents making errors is because it is able to identify the extremely low quality images as outliers and isolates them, instead of attributing them to an incorrect cluster.

Finally, we evaluated how our algorithm generalizes to camera trap images of other animals with distinct spot patterns, i.e., snow leopards, also provided by *Panthera* [11]. The distribution of the snow leopard dataset, which is shown in Fig. 11, is quite different from the African leopard dataset. This dataset is even smaller, i.e., $N = 224$, and also a smaller ratio of K/N, where the number of snow leopards, $K = 18$. The results are shown in Table II, where we compare the number of images in three different cluster types (correct, partially correct, incorrect) in the best clusterings obtained from k++ and adaptive k++ with verification, as determined by our expanded silhouette score. Our adaptive k++ with verification does not result in any incorrectly clustered images and has more than double the number of completely correct images as compared to the baseline k++. The reason for the relatively large number of partially correct images as compared to the African leopard dataset is due to the radically different distributions of numbers of individuals in a cluster. In particular, for the snow leopard dataset, 132 of the 224 images belong to only 4 individuals, with two clusters having over 40 images each, whereas for the African leopard dataset, over half of the individuals belong to isolated clusters. Therefore the situation that is illustrated in Fig. 8 occurs more frequently for the snow leopard dataset.

V. CONCLUSION

In this paper, we propose an effective algorithm to automatically identify individual animals in a dataset of unlabeled camera-trap images of African leopards, provided by *Panthera*. This work is currently done by experts and is time consuming and error prone. Our algorithm consists of a modified ternary search that uses a novel adaptive k-medoids++ clustering algorithm to determine the best clustering and the corresponding cluster number. Our adaptive k++ introduces a weight factor for each image to adjust the probability of the image being chosen as a cluster medoid seed during iterations. The weight of an image that is a cluster medoid is adaptively adjusted to reflect its cluster quality, measured by an expanded definition of the silhouette score. Our silhouette score is an effective proxy to evaluate the performance of clustering and the quality of a single cluster. In addition, to further improve the clustering performance, we propose a post-clustering verification procedure to potentially reassign outlier images and merge companion clusters. Future work will address the effects of multiple medoid seeds in large clusters by combining images that are similar to more than one cluster.

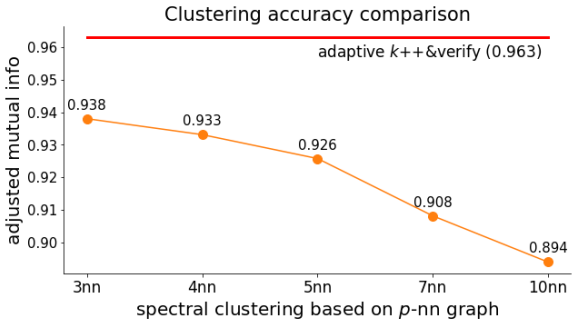

Fig. 12. This figure plots the clustering accuracies (measured by adjusted mutual information score) for the best clusterings of adaptive k++ with verification and p-nn spectral for different values of p. Both approaches were executed with $k = K = 677$. The accuracy of our adaptive k++ with verification, 0.963, is higher than the overall of p-nn spectral, e.g., the best result of p-nn spectral ($p = 4$), 0.938. Note that to display the representative result of p-nn spectral, we used an optimal number of eigenvectors, 100, under a range of values.

APPENDIX
COMPARISON TO SPECTRAL CLUSTERING

To further validate the effectiveness of our adaptive k-medoids++ clustering technique, we have selected an additional unsupervised technique to evaluate and compare to our approach. The technique we selected is spectral clustering based on p nearest neighbor graph (p-nn spectral) [28] because it (1) also uses a similarity matrix, (2) is fundamentally different than k++, and (3) has a reputation for better performance on our type of dataset. Unfortunately, p-nn spectral requires three parameters: (1) the number of clusters (k), (2) the number of nearest neighbors (p), and (3) the number of eigenvectors, which makes the algorithm more difficult to apply. However, we selected a range of parameters and showed a representative result of variation as a function of p, using the true value of k, and an optimal number of eigenvectors, 100. The comparison results are shown in Fig. 12, which plots the clustering accuracies of the best clusterings for our adaptive k++ with verification and p-nn spectral under different values of p. This figure indicates the accuracy of our approach is higher than the overall p-nn spectral. Once again, it is important to note that this result is unachievable in practice because it requires an oracle that provides the optimal values for p, k, and the best number of eigenvectors.

ACKNOWLEDGMENT

The authors would like to thank *Panthera* for providing the camera trap data set.

REFERENCES

[1] M. S. Norouzzadeh et al., "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 25, pp. E5716–E5725, Jun. 2018.

[2] D. Tuia et al., "Perspectives in machine learning for wildlife conservation," *Nature Commun.*, vol. 13, no. 1, pp. 1–15, 2022.

[3] M. Vidal, N. Wolf, B. Rosenberg, B. P. Harris, and A. Mathis, "Perspectives on individual animal identification from biology and computer vision," *Integrative Comparative Biol.*, vol. 61, no. 3, pp. 900–916, Oct. 2021.

[4] S. Schneider, G. W. Taylor, S. Linquist, and S. C. Kremer, "Past, present and future approaches using computer vision for animal re-identification from camera trap data," *Methods Ecol. Evol.*, vol. 10, no. 4, pp. 461–470, 2019.

[5] S. Kumar and S. K. Singh, "Visual animal biometrics: Survey," *IET Biometrics*, vol. 6, no. 3, pp. 139–156, May 2017.

[6] E. Nepovinnykh, T. Eerola, H. Kälviäinen, and G. Radchenko, "Identification of saimaa ringed seal individuals using transfer learning," in *Proc. Int. Conf. Adv. Concepts Intell. Vis. Syst.* Cham, Switzerland: Springer, 2018, pp. 211–222.

[7] E. Nepovinnykh, T. Eerola, and H. Kälviäinen, "Siamese network based pelage pattern matching for ringed seal re-identification," in *Proc. IEEE Winter Appl. Comput. Vis. Workshops (WACVW)*, Mar. 2020, pp. 25–34.

[8] S. Li, J. Li, H. Tang, R. Qian, and W. Lin, "ATRW: A benchmark for amur tiger re-identification in the wild," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 2590–2598.

[9] P. Chen et al., "A study on giant panda recognition based on images of a large proportion of captive pandas," *Ecol. Evol.*, vol. 10, no. 7, pp. 3561–3573, Apr. 2020.

[10] C. V. Stewart, J. R. Parham, J. Holmberg, and T. Y. Berger-Wolf, "The animal id problem: Continual curation," 2106, *arXiv:2106.10377*.

[11] Panthera. (2019). *An Image Dataset of Wild African Leopards*. [Online]. Available: https://panthera.org/

[12] J. P. Crall, C. V. Stewart, T. Y. Berger-Wolf, D. I. Rubenstein, and S. R. Sundaresan, "HotSpotter—Patterned species instance recognition," in *Proc. IEEE Workshop Appl. Comput. Vis. (WACV)*, Jan. 2013, pp. 230–237.

[13] S. Nayeri, M. Sargolzaei, and D. Tulpan, "A review of traditional and machine learning methods applied to animal breeding," *Animal Health Res. Rev.*, vol. 20, no. 1, pp. 31–46, Jun. 2019.

[14] D. T. Bolger, T. A. Morrison, B. Vance, D. Lee, and H. Farid, "A computer-assisted system for photographic mark–recapture analysis," *Methods Ecol. Evol.*, vol. 3, no. 5, pp. 813–822, Oct. 2012.

[15] T. Chehrsimin et al., "Automatic individual identification of saimaa ringed seals," *IET Comput. Vis.*, vol. 12, no. 2, pp. 146–152, Mar. 2018.

[16] A. Miguel, J. S. Beard, C. Bales-Heisterkamp, and R. Bayrakcismith, "Sorting camera trap images," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2017, pp. 249–253.

[17] J. Duyck, C. Finn, A. Hutcheon, P. Vera, J. Salas, and S. Ravela, "Sloop: A pattern retrieval engine for individual animal identification," *Pattern Recognit.*, vol. 48, no. 4, pp. 1059–1073, Apr. 2015.

[18] S. Schneider, G. W. Taylor, and S. C. Kremer, "Similarity learning networks for animal individual re-identification–beyond the capabilities of a human observer," in *Proc. IEEE Winter Appl. Comput. Vis. Workshops (WACVW)*, Mar. 2020, pp. 44–52.

[19] C. Geng, S. Huang, and S. Chen, "Recent advances in open set recognition: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3614–3631, Oct. 2021.

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[21] M. Perd'och, O. Chum, and J. Matas, "Efficient representation of local geometry for large scale object retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 9–16.

[22] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A *k*-means clustering algorithm," *J. Roy. Stat. Soc. Ser. C, Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.

[23] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, Mar. 2009.

[24] D. Arthur and S. Vassilvitskii, "*k*-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2007, pp. 1027–1035.

[25] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, Feb. 2020.

[27] M. S. Bajwa, A. P. Agarwal, and S. Manchanda, "Ternary search algorithm: Improvement of binary search," in *Proc. 2nd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2015, pp. 1723–1725.

[28] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[29] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Dec. 2010.

[30] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw., Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, 1991.

Cheng Guo (Student Member, IEEE) received the M.S. degree in electrical engineering and automation from the University of Jinan, Jinan, Shandong, China, in 2015. She is currently pursuing the Ph.D. degree in computer engineering with Colorado State University, Fort Collins, CO, USA.

Her research interests include image processing.

Agnieszka Miguel (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Florida Atlantic University, Boca Raton, FL, USA, in 1994 and 1996, respectively, and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, WA, USA, in 2001. She is currently a Professor and the Chair of Electrical and Computer Engineering with Seattle University, Seattle, WA, USA. Her research interests include data compression, image processing, and engineering education.

Anthony A. Maciejewski (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from The Ohio State University, Columbus, OH, USA, in 1982, 1984, and 1987, respectively.

From 1988 to 2001, he was a Professor in electrical and computer engineering with Purdue University, West Lafayette, IN, USA. He is currently a Professor in electrical and computer engineering with Colorado State University, Fort Collins, CO, USA. His research interests include robotics, high-performance computing, and engineering education.